StyleGAN as an AI deconvolution operator for large eddy simulations of turbulent plasma equations in BOUT++

Special Collection: Papers from the 4th International Conference on Data-Driven Plasma Science

J. Castagna ; F. Schiavello ; L. Zanisi ; J. Williams

Phys. Plasmas 31, 033902 (2024)
https://doi.org/10.1063/5.0189945
StyleGAN as an AI deconvolution operator for large eddy simulations of turbulent plasma equations in BOUT++

Cite as: Phys. Plasmas 31, 033902 (2024); doi: 10.1063/5.0189945
Submitted: 1 December 2023 · Accepted: 21 February 2024 ·
Published Online: 21 March 2024

J. Castagna,1,a) F. Schiavello,1 L. Zanisi,2 and J. Williams1

AFFILIATIONS

1Hartree Centre, UKRI-STFC, Sci-Tech Daresbury Keckwick Lane, Warrington WA4 4AD, United Kingdom
2Advanced Computing, CCFE, Culham Science Centre, Abingdon OX14 3DB, United Kingdom

Note: This paper is part of the Special Collection: Papers from the 4th International Conference on Data-Driven Plasma Science.

a)Author to whom correspondence should be addressed: jony.castagna@stfc.ac.uk

ABSTRACT

We present the use of StyleGAN, a face-synthesis generative adversarial network (GAN) developed by NVidia, as a deconvolution operator for large eddy simulation (LES) of plasma turbulence. The overall methodology, named style eddy simulation, has been integrated into the BOUT++ solver and tested on the original and modified Hasegawa–Wakatani models using different mesh sizes, 2562 and 5122, and different values of the adiabaticity parameter \( \gamma \) and background density gradient \( \kappa \). Using a LES resolution of \( 32 \times 32 \) and \( 64 \times 64 \), i.e., \( 64 \times \) smaller resolution than the corresponding direct numerical simulation (DNS), results show convergence toward the ground truth as we tighten the reconstruction tolerance, and an algorithm complexity \( O(N \log N) \) is compared to the \( O(N^2) \) of BOUT++. Finally, the trained GAN can be used to create valid initial conditions for a faster DNS by avoiding to start from nonphysical initial perturbations.

© 2024 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/). https://doi.org/10.1063/5.0189945

I. INTRODUCTION

Direct numerical simulation (DNS) of turbulent plasma is a very useful tool for understanding the complex evolution of edge turbulence occurring in tokamak devices. This is due to its lack of ad hoc models and, hence, high fidelity in the results. However, DNS is computationally expensive, due to fine space-temporal resolutions required. On the other hand, large eddy simulations (LESs) are much faster, being run on a coarser resolution altogether, but an ad hoc modeling of the fine scales is required. This is notoriously very challenging in plasma physics as the so-called sub-grid scale model must not only be able to capture the dissipation occurring during the direct cascade of energy but also complex patterns, like zonal flows, occurring as a reorganization of the turbulent regime, and the strong back-scattering of energy from the small scales, which leads to an inverse cascade. In fact, even for simplified cases, like the modified Hasagawa–Wakatani (HW) system, obtaining the correct particle flux and energy spectra can be extremely challenging, where common sub-grid scale models, like the Smagorinsky model, easily fail in doing so. Some other models have been proposed and compared by Leddy in their thesis work, but none of these have been able to capture all the examined features (energy spectra, particle flux, and enstrophy). The key contribution of this work is to provide a fast and accurate AI-driven methodology to compute these terms, while also maintaining the robustness of classical solvers.

Across the range of AI approaches, deep learning has recently captured a huge attention in the scientific community. In particular, generative adversarial networks (GANs) are neural networks composed of two competing sections: a generator, which generates samples from random noise, and a discriminator, trained to recognize whether a given sample is an output of the generator or if it comes from the real dataset. Both are trained at the same time, until they achieve a so-called Nash equilibrium, where discriminator and generator are in a state in which neither can improve further. In such conditions, the discriminator may not be able to distinguish the difference between real and synthetic data, depending on the equilibrium score achieved. Following our previous work on Navier–Stokes equations, we present these results on the use of StyleGAN, an NVidia GAN for face synthesis, as a deconvolution operator for large eddy simulation (LES) of plasma turbulence. The manuscript shows how StyleGAN is capable of reconstructing 2D direct numerical simulation (DNS) images of the...
density, electrostatic potential, and vorticity fields. The time integration of these quantities occurs via the BOUT++ numerical solver after calculating the correct sub-grid scale quantities from the reconstructed DNS fields.

Approximate deconvolution methods (ADMs) for LES were introduced in 1999 by Stolz et al. using an approximate inversion of the unfiltered velocity expanded in a Taylor series. Other formulations have been proposed, with different levels of complexity but failed to capture fine details in the structures. Neural networks (NNs) have been recently proposed as ADM methods, and tests on turbulent channel flows have shown good agreement with DNS ground truth data. A more complex architecture, such as GANs, has been proposed for the same turbulent channel flow, but the time integration relies on a recurrent neural network (RNN). Good results were obtained for steady conditions but, in general, training RNN on such tasks is difficult to succeed due to the high compression of the information embedded into the latent space. A LES-GAN based model has also been proposed, where a cycleGAN has been used for three different tasks: (i) to recover the original flow field from filtered data; (ii) to reconstruct full-resolution fields, using partially measured data from the DNS of turbulent channel flows; and (iii) to generate a DNS-resolution flow field from LES data. Again, good agreement with DNS data have been found to prove the capability of GANs, in general, to learn the complex highly non-linear correlations embedded into the NS equations, but no applications to plasma turbulence have been carried out so far using this methodology.

In StyleGAN, each block of the generators would produce an image of a given resolution. As in painting, each image has a content and a style. A common example in generative AI literature is van Gogh’s style applied to the nocturne sky, which gives the famous “Starry Night” canvas painting. In the same way, the style of each generator block can be changed according to the latent space. In the case of turbulence, the contents are the turbulence structures, and the style can be linked to the Reynolds number (and other characteristic numbers), with fine vortices for high Reynolds and larger blobs for lower values. Coarse, medium, and fine details can be changed via linear interpolation between the different styles, as shown in Fig. 1 (Multimedia view). The first use of StyleGAN for turbulent flow appears in the work of Gupta et al., where it has been used for weather forecast simulations achieving an order of magnitude larger high-fidelity images with accurate energy spectra. However, the use of the different styles was not investigated. In the field of plasma turbulence, Greg et al. proposed a novel machine-learning-based LES technique based on a potential-surrogate correction. However, to the best of the authors’ knowledge, there are no applications of GANs to plasma turbulence.

In this paper, we propose to test the style eddy simulation (StylES) algorithm for plasma turbulent flows using the Hasegawa–Wakatani and modified Hasegawa–Wakatani 2D turbulence models (hereafter, HW and mHW, respectively). We integrated the methodology into the BOUT++ solver to investigate the phenomena occurring in the divertor region of Tokamak reactors. The work presented and investigated here is in 2D scenarios, but there is no theoretical limit to extend this work to other 3D turbulence scenarios. The 3D models are outside of scope for this paper but are in the pipeline for the potential future work. Instead, in this paper, we will present the main novelties of our current work, which can be described as the use of StylES for plasma turbulent simulations and potential speedup for large systems; its validity when interpolating between different latent spaces, which is required to match the BOUT++ LES field; the use of the GAN generator for initial conditions of DNS; and an improvement of the StylES algorithm itself to allow better and faster convergence toward ground truth.

We tested the HW and mHW models on two different resolutions, 256 and 512, and compared the results with the ground truth DNS obtained using the BOUT++ solver. The time integration occurs after a match between the LES fields produced by the solver and the GAN-LES fields controlled by a tolerance $\varepsilon_{REC}$. Results show convergence toward the ground truth DNS as $\varepsilon_{REC} \rightarrow 0$, and all the main properties of the flow are recovered. Moreover, for the StylES algorithm, we observe a complexity of $\mathcal{O}(N \log N)$, where $N$ is the number of points per geometrical dimension, compared to the $\mathcal{O}(N^2)$ of BOUT++, i.e., for larger systems, we obtain larger speedup. However, as optimization of the current solver has not yet been carried out, benchmarking of the different tests is outside of this manuscript’s scope and will be described in the successive work. The primary intention here is to prove the validity of the methodology and the integration with a common solver used in the community of plasma turbulence simulation.

The paper continues as follows: Sec. II presents the theoretical background and the StylES methodology. Section II B explains how training data for the GAN has been produced, and Sec. III shows the results on the different tests and scenarios with current limitations and validity. Conclusion and future work are presented in Sec. IV. The code used to generate the results is available on the GitHub BOUT++ and StylES repositories, both need to be downloaded and linked following the instructions on StylES repository README file.

II. METHODOLOGY

Here, we give a quick resume of the StylES algorithm to reconstruct the fluid flow from a LES field at a given time $t_0$. More details can be found in the work of Castagna and Schiavello.
A. Theoretical background

LES is obtained by filtering the model equations according to a given filtering operation\textsuperscript{22} defined as

$$\tilde{U}(x, t) = \int F(r, x) U(x - r, t) \, dr = F \ast U(x, t), \quad (1)$$

where $U$ is a generic vector field,\textsuperscript{7} represents a filtered variable, and $F$ is the filter function satisfying the normalization condition given as follows:

$$\int F(r, x) \, dr = 1. \quad (2)$$

When applied to the HW equations, we obtain the following filtered equations (for a homogeneous filter):

$$\frac{\partial \tilde{\eta}}{\partial t} + \{\tilde{\zeta}, \tilde{\eta}\} = \tilde{z} (\tilde{\phi} - \tilde{n}) - \mu_\xi \nabla^4 \tilde{\eta},$$

$$\frac{\partial \tilde{n}}{\partial t} + \{\tilde{\phi}, \tilde{n}\} = \tilde{z} (\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \tilde{\phi}}{\partial y} - \mu_n \nabla^4 \tilde{n}, \quad (3)$$

or expanding the Poisson brackets terms

$$\frac{\partial \tilde{\eta}}{\partial t} + \frac{\partial \tilde{\phi}}{\partial y} \frac{\partial \tilde{\eta}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x} \frac{\partial \tilde{\eta}}{\partial y} = \tilde{z} (\tilde{\phi} - \tilde{n}) - \mu_\xi \nabla^4 \tilde{\eta},$$

$$\frac{\partial \tilde{n}}{\partial t} + \frac{\partial \tilde{\phi}}{\partial y} \frac{\partial \tilde{n}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x} \frac{\partial \tilde{n}}{\partial y} = \tilde{z} (\tilde{\phi} - \tilde{n}) - \kappa \frac{\partial \tilde{\phi}}{\partial y} - \mu_n \nabla^4 \tilde{n}$$

$$- D_{\phi, \phi} \frac{\partial \tilde{\phi}}{\partial y} + D_{\phi, \phi}, \quad (4)$$

where $n$ is the perturbed particle density, $\phi$ is the electrostatic potential, and $\zeta$ is the vorticity field. Moreover, $\mu_\xi = \mu_\nu$ are the hyperviscosity diffusion coefficients, usually added for numerical stability, while $z$ and $\kappa$ are the adiabaticity parameter and the background gradient density, respectively. Finally,

$$\frac{\partial \phi}{\partial y} \frac{\partial \tilde{\eta}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x} \frac{\partial \tilde{\eta}}{\partial y} = D_{\phi, \phi} \frac{\partial \tilde{\phi}}{\partial y}$$

$$\frac{\partial \phi}{\partial y} \frac{\partial \tilde{n}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x} \frac{\partial \tilde{n}}{\partial y} = D_{\phi, \phi}, \quad (5)$$

are the closure models for our filtered system.

The filtered mHW equations are very similar, but the following terms are replaced:

$$\tilde{z} (\tilde{\phi} - \tilde{n}) \rightarrow \tilde{z} (\tilde{\phi} - \tilde{n}),$$

$$\tilde{z} (\tilde{\phi} - \tilde{n}) \rightarrow \tilde{z} (\tilde{\phi} - \tilde{n}). \quad (6)$$

where

$$f = f - \frac{1}{L_y} \int f \, dy, \quad (7)$$

i.e., the average values along the normal direction are used to adjust the adiabatic term and allow the formation of zonal flows.\textsuperscript{21}

The Stiles idea is to reconstruct the $n$, $\phi$, and $\zeta$ terms at DNS resolution from their corresponding $\tilde{n}$, $\tilde{\phi}$, and $\tilde{\zeta}$, find the nonlinear terms $\{\}$, and then solve the filtered Eq. (4). Note that only $n$ and $\phi$ could be reconstructed, as $\zeta$ is obtained as the Laplacian of $\phi$. However, the GAN does not guarantee smooth second-order derivatives, and some small high-frequency oscillations could be captured and enhanced during time integration. Therefore, we reconstruct all three fields.

The full methodology relies on the following three main assumptions:

(1) The existence and uniqueness of the solution. This has been recently proven for the HW system of equations,\textsuperscript{23} and it is essential to guarantee the unique correlation between LES fields and DNS fields when an invertible filter is used. If this condition is not satisfied, the problem becomes ill-posed.

(2) The smoothness and continuity of the GAN manifold. It can be proven that under certain conditions,\textsuperscript{24,25} the generated StyleGAN images from a latent space manifold $\mathcal{Z}$ map to a smooth manifold $\mathcal{M}$. This condition is essential to guarantee the convergence during the latent space search.

(3) The linear interpolation on the latent space always produces a valid DNS field. In the author’s knowledge, this has not yet been proven, however, from our experiments, we have found this to always be true and as such we raise this to the level of conjecture.

These three conditions have been confirmed \textit{a posteriori} by the observation of the search in the latent space, which converges in a fast manner toward the correct solution when searching for the correct field at a later time step, $t + \Delta t$. This search in the latent space is fruitful for $\Delta t$ up to the first 10k steps. Afterward, we observe scenarios where this search can become very slow, and current investigation is under way to understand the root of this undesirable effect.

B. Dataset

Taking Leddy’s thesis as a reference,\textsuperscript{6} the simulation size of the system is $L_x = L_y = 50.176$, while the parameters $\alpha$, $\kappa$, $\nu_\zeta$, and $\nu_\phi$ are defined according to \textit{Table I}. The different values of hyper-viscosity are necessary to have a match between the energy spectrum smallest scales and the mesh resolution (see Fig. 2) and avoid a null filter from DNS to LES. For the mHW, fully periodic boundary conditions are used in both directions in order to simplify the GAN construction which should, otherwise, consider the non-periodicity during the different convolutional layers.

<table>
<thead>
<tr>
<th>Type</th>
<th>$N$</th>
<th>$\alpha$</th>
<th>$\kappa$</th>
<th>$\nu_\zeta = \nu_\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW</td>
<td>$256^2$</td>
<td>1</td>
<td>0.5</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>HW</td>
<td>$512^2$</td>
<td>1</td>
<td>1</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>mHW</td>
<td>$256^2$</td>
<td>1</td>
<td>1</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>mHW</td>
<td>$512^2$</td>
<td>1</td>
<td>1</td>
<td>$10^{-4}$</td>
</tr>
</tbody>
</table>

\textit{Table I}: Physical properties defined for each test case.
The simulation is initialized with an unphysical perturbation (see Fig. 4.2 of Leddy’s thesis), which then evolves to reach a statistically steady state (Fig. 3). Training data are then gathered once the flow is statistically steady. A comparison of the energy spectra vs time is presented in Fig. 3: it can be noted that the evolution of initial nonphysical artificial initial perturbations.

The energy spectra follow the power law $k^{-3}$ typical of the energy cascade in 2D turbulence.

The simulation is initialized with an unphysical perturbation (see Fig. 4.2 of Leddy’s thesis, which then evolves to reach a statistically steady state (Fig. 3). Training data are then gathered once the flow is statistically steady. A comparison of the energy spectra vs time is presented in Fig. 3: it can be noted that the evolution of initial nonphysical artificial initial perturbations is different for the test cases. The different dataset have been collected in Fig. 3; it can be noted that the evolution of initial nonphysical perturbations.

![Energy spectra for the different test cases. As expected, the slope in the energy spectra follows the power low $k^{-3}$ typical of the energy cascade in 2D turbulence.](image)

**Fig. 2.** Energy spectra for the different test cases. As expected, the slope in the energy spectra follows the power low $k^{-3}$ typical of the energy cascade in 2D turbulence.

**Fig. 3.** Energy vs time for each test case showing the different evolution of the artificial initial perturbations.

The neural networks used in our study are shown in Fig. 4. As our GAN [Fig. 4(a)], we use the StyleGAN architecture. As in our previous work, we use a variant called MSG-StyleGAN (where MSG refers to multi-scale gradients), since this is more robust during training compared to StyleGAN. This works by converting the feature map at each layer of the generator to a three channel image ($n$, $\phi$, and $\zeta$), which is then passed to the corresponding layer in the discriminator. This will allow the flow of gradients from the discriminator to the generator at multiple scales, facilitating the learning of the dataset features.

The GAN loss function used is the Wasserstein loss, which improves training stability. The Wasserstein loss is defined as follows. Let us define the generator output from a random latent variable, $z$, as $G(z)$. The discriminator, $D$, learns to differentiate between real ($x$) and fake images [$G(z)$]. The generator, $G$, learns to produce an image closer to the distribution of the training data based on the discriminator output. The loss functions are

\[ \mathcal{L}_G = \log(1 + \exp(-D(G(z)))) \]  
\[ \mathcal{L}_D = \log(1 + \exp(D(G(z)))) + \log(1 + \exp(-D(x))) + \gamma R_1, \]

where $\mathcal{L}_G$ is the generator loss, $\mathcal{L}_D$ is the discriminator loss, and $\gamma R_1$ is a gradient regularization on the discriminator.

We trained the generator and discriminator of StyleGAN with the Adam optimizer. The learning rates were $7 \times 10^{-4}$ for the generator and $3 \times 10^{-3}$ for the discriminator (a ratio of four between the two values). This is referred to as the two timescale-update rule (TTUR), which improves the discriminator’s convergence as its weights can adapt to changes in patterns produced by the generator quicker than the generator can change the patterns produced. The training is stopped when a valid Nash equilibrium is achieved. In our case, when the energy spectra of the generated DNS fields are within a 5% difference from the dataset.

While training StyleGAN, we also train the convolution/deconvolution operator [Fig. 4(b)] that allows us to move from fine to coarse grid fields, and vice versa, which is needed in the search of the latent space to find the correct LES–DNS pair of fields (described in Sec. II D). The inverse of the deconvolution operator is a multi-step convolutional filter, i.e., a series of Gaussian blur filters followed by convolutional layers to avoid aliasing effects, as shown in Fig. 4(b). The filter is trained to minimize the Euclidean distance between the filtered DNS and LES fields as

\[ \mathcal{L}_f = ||f \ast U^\text{GAN}_\text{DNS} - U^\text{GAN}_\text{LES}||_2^2. \]

The loss function for the filter in Eq. (10) is minimized using the Adam optimizer with a learning rate of $10^{-4}$.

As the final training value of $\mathcal{L}_f$ is never exactly zero, a correction term has to be added in the latent space search loss function $\mathcal{L}_{\text{REC}}$ to compensate for it, as described in Sec. II D.

**TABLE II. Computational effort and memory usage for each test case.**

<table>
<thead>
<tr>
<th>Type</th>
<th>N</th>
<th>$T_f$</th>
<th>$T_s$</th>
<th>$N_{runs}$</th>
<th>$N_s$</th>
<th>$C_{eff}$</th>
<th>Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW</td>
<td>256$^2$</td>
<td>501</td>
<td>1000</td>
<td>10</td>
<td>100</td>
<td>5000</td>
<td>7.4</td>
</tr>
<tr>
<td>HW</td>
<td>512$^2$</td>
<td>201</td>
<td>1000</td>
<td>4</td>
<td>100</td>
<td>5000</td>
<td>111</td>
</tr>
<tr>
<td>mHW</td>
<td>256$^2$</td>
<td>201</td>
<td>1000</td>
<td>4</td>
<td>100</td>
<td>5000</td>
<td>7.4</td>
</tr>
<tr>
<td>mHW</td>
<td>512$^2$</td>
<td>201</td>
<td>400</td>
<td>4</td>
<td>100</td>
<td>5000</td>
<td>111</td>
</tr>
</tbody>
</table>

The choice of $\Delta T_s$ is such that the variation in similarity structural index metric (SSIM) is always $\geq 10\%$. The last two columns show the overall computational effort $C_{eff}$ in node per hours, where 1 node of ScafellPike supercomputer at Hartree Center contains a two socket of 16-cores Intel Xeon Gold processors, and the disk space occupied by the data.
These operations and training of the filter are key to our methodology, as they allow us to accurately integrate the flow fields in time at lower resolutions without implicating a loss in accuracy traditionally seen in other coarse solvers.

D. The StylES algorithm

The deconvolution operator used in StyleGAN is essentially made of a series of upscaling layers to increase the resolution of the LES, i.e., we double the mesh resolution adding transposed convolutions. Other operations, like the adaptive instance normalization (AdaIN) and noise injection, allow us to control the style of each block, adding the missing reconstruction terms from the upscaling. For a given value of the intermediate latent space $W$ and for a given value of the input noises $N$, the operator is a single-valued function, which satisfies the required conditions of being fully invertible.

The StylES algorithm consists of reconstructing the DNS fields from the LES ones via a search in the latent space occurring only by linear interpolation between two or more extended latent spaces $W^+$ (this being the intermediate latent space $W$ replicated for each of the layers in the generator), i.e., $W = k_{LES} W' + (1 - k_{LES}) W''$ [see Fig. 4(c)]. Note that $W'$ and $W''$ are randomly generated, $k_{LES}$ is a vector of length $512 \times N_b$, where $N_b$ is the number of blocks of the generator and can be found as $N_b = \log_2(N_{DNS}) \times 2 - 2$, $N_{DNS}$ is the number of blocks of the generator.

### FIG. 4

Overview of neural network architectures used in this study. Panels show (a) the GAN model for generating DNS fields and (b) the convolution operator, which is inverted to perform deconvolution at inference time. (c) The latent space interpolation to find a field that satisfies (11). In (c), the BOUT++ operations are given in blue text, whereas the black text refers to steps of the StylES algorithm, which occurs on the GPU.
number of points in one direction for the DNS resolution, where the mesh grid is square. The interpolation occurs only for the first LES layers to make sure the deconvolution operator, and as such the reconstruction, is the same for all time steps within the algorithm. This is important to guarantee the uniqueness of the solutions. The loss function used during reconstruction is defined as follows:

$$\mathcal{L}_{REC} = \| U_{LES}^{GAN} - F \ast U_{DNS} \|^2 + \| F \ast U_{DNS} - F \ast U_{DNS} \|^2,$$

(11)

where $U$ represents the three fields $n$, $\phi$, and $\zeta$, i.e., the solution of the HW equations. The first term on the right hand side represents the Euclidean distance from the GAN–LES layer and the LES target, while the second term represents the Euclidean distance from the filtered DNS layer and the LES target, i.e., the filtered DNS field. For an ideal filter (see Sec. II C), i.e., a filter trained with a final filter loss of DNS layer and the LES target, i.e., the filtered DNS field. For an ideal filter, which is the ideal condition for a fully invertible filter. Note that the deconvolution–convolution pair is balanced and matched. As both quantities are positive when $\mathcal{L}_{REC} ightarrow 0$, we obtain

$$U_{LES}^{GAN} = F \ast U_{DNS}.$$

(12)

$$F \ast U_{DNS} = F \ast U_{DNS},$$

(13)

which is the ideal condition for a fully invertible filter. Note that $U_{DNS}$ is never known, this being the DNS field, while $F \ast U_{DNS}$ is our LES field at time $t + \Delta t$, obtained from the time integration of the filtered equations.

Some other caveats are listed here:

- During the training of StyleGAN, we use a randomized noise to avoid mode collapse (where the GAN fails to learn the full dataset distribution of images and repeatedly outputs only a small set of images), in line with the original StyleGAN work by Karras et al. This consists into injecting a different noise in every layer at every step of the training. However, during inference, the noise is fixed; otherwise, the search into the latent space will fail due to the continuous change of direction imposed by the different noises.

- The output of StyleGAN is forced to be normalized between $-1$ and $1$, i.e., $\tilde{U} = \tilde{U}$, where $\tilde{\cdot}$ indicates the following normalization operation:

$$\tilde{U} = 2 \frac{U - U_{min}}{U_{max} - U_{min}} - 1.$$

(14)

This condition, together with the conservation property of the LES filter $\ast k U = k F \ast U$, leads to

$$U_{DNS, min} = U_{LES, min},$$

$$U_{DNS, max} = U_{LES, max},$$

(15)

which can be used to rescale the LES field to the DNS one. It follows that the correct absolute value of the DNS field is obtained by rescaling the normalized one with same minimum and maximum values of the LES field.

### Algorithm 1: The Styles algorithm

**Require:** a trained StyleGAN based generator $\mathcal{G}(\cdot)$, an initial latent space $\mathcal{W} = 0.5\mathcal{W} + (1 - 0.5)\mathcal{W}$, which gives a DNS field $U_{DNS}(t_0)$ as a starting point; a trained filter $F$ from $U_{DNS}$ to $U_{LES}$, a scaling factor $K$ (we set $K = 10$ in all our simulations).

1. infer $U_{GAN}$ field and scale by $K$
2. subtract the average values to get $U_{DNS}(t_0) = K U_{DNS} - \text{mean}(K U_{DNS})$
3. take $U_{LES}(t_0) = F \ast U_{DNS}(t_0)$
4. while time steps $< \text{total time steps}$ do
5. find filtered nonlinear terms $U_{DNS}(t)U_{DNS}(t)$
6. solve filtered NS equations to find $U_{LES}(t + \Delta t)$
7. while $\mathcal{L}_{REC} > \epsilon_{REC}$ do
8. infer $U_{GAN}^{LES}(t + \Delta t)$ field, scale by $K$ and subtract the average values
9. find $\mathcal{L}_{REC}$
10. adjust $k_{LES}$ via backpropagation keeping $\mathcal{G}$ weights constant
11. end while
12. take $U_{DNS}(t + \Delta t) = U_{DNS}^{GAN}(t + \Delta t)$
13. increment time step
14. end while

- All three fields need to be null on average across the full domain. This is due to the quasi-neutrality condition, for which

$$\begin{align*}
\nabla \cdot E &= 0, \\
\n\nabla^2 \phi &= \zeta &= 0.
\end{align*}$$

(16)

The full Styles procedure is then summarized in Algorithm 1.

### E. Integration with BOUT++

The integration with BOUT++ occurs via a Python-embedded call to TensorFlow (>v2.10) from C++. However, rather than finding the $D_j$ terms of Eq. (5), we directly compute the Poisson terms on the left-hand side of Eq. (3) with finite differences in BOUT++ as

$$\{ \phi, \zeta \} = \frac{\partial \phi}{\partial y} \frac{\partial \zeta}{\partial y} - \frac{\partial \phi}{\partial x} \frac{\partial \zeta}{\partial x},$$

$$\{ \phi, \eta \} = \frac{\partial \phi}{\partial y} \frac{\partial \eta}{\partial x} - \frac{\partial \phi}{\partial x} \frac{\partial \eta}{\partial y}.$$  

(17)

This has a twofold advantage: it allows us to avoid the calculation of the Poisson bracket term on the CPU, saving computing resources, and it reduces the transfer CPU–GPU, saving on communication time. The computed terms of Eq. (17) are added in the explicit part of the BOUT++ solver to help convergence and stability. Moreover, these terms are calculated only when the difference between the LES fields is greater than the reconstruction tolerance $\epsilon_{rec}$ otherwise, they are kept as constants from the previous time step. Finally, correct discretization to allow conservation properties, like mass and energy, has to be taken into account when calculating the Poisson brackets terms.
BOUT++ uses the Arakawa discretization method and so does the StyleGAN algorithm.

The training of the MSG-StyleGAN occurs on 1 NVidia V100 GPU and takes 12 and 24 h for the 256 and 512 test cases, respectively.

Note that we train and infer in single precision float32 for speedup purposes, but BOUT++ runs in double precision float64. There will be a rounding error which, in our experiments, does not seem to affect the overall results. If double precision were to be used the timing would be theoretically doubled. Moreover, each channel is normalized by its minimum and maximum values. The reason is to improve the stability of the GAN during the training. This operation will break the correlation \( \nabla^2 \phi = \zeta \), which will be reinstated by BOUT++ at the LES solution. However, to avoid a long search in the latent space at the first time step to adjust to the correct potential field, only the Euclidean distance between \( n \) and \( \zeta \) is considered in Eq. (11). We will see from the results that the GAN potential field will follow the vorticity.

III. RESULTS

We first present the results on the mHW 256\(^2\) resolutions with details on the initial flow conditions, energy spectra, and impact of the reconstruction tolerance to the centerline trajectories of each field. We will then focus on the rest of the test cases. Note that the DNS results are obtained starting from a StyleGAN field. This proves that, apart from the initial adjustment in the electrostatic potential, the GAN can be used to generate a valid field for DNS. This can save considerable computational effort compared to an initial condition based on non-physical disturbances.

A. mHW 256\(^2\) resolution

An example of reconstruction from the LES 32\(^2\) vorticity field to the 256\(^2\) DNS field is given in Fig. 5. We then run the integrated BOUT++ with StylES solver for ten time units (~10k time steps) using the following different reconstruction tolerances \( \varepsilon_{\text{REC}} \): 1.0, 0.5, and 0.25. We then compared with the DNS ground truth the initial fields at time \( t = 0.01 \) and corresponding energy spectra, and the center plane trajectory of each field vs time to check the impact of the tolerance.

Figure 6 shows the differences at \( t = 0.01 \) between the inference of the trained StyleGAN and the actual DNS values when using \( \varepsilon_{\text{REC}} = 0.25 \). The main differences are in the potential fields. As previously mentioned, this is due to the lack of StyleGAN in correctly capturing the \( \nabla^2 \phi = \zeta \) correlation, and hence, a drastic change occurs at the first step. The mean square error (MSE) is 0.038, and the structural similarity index measure (SSIM) is 0.73. Figure 7(a) shows a comparison of the energy spectra at time \( t = 0.01 \), where small discrepancies can be observed due to the potential field. Figure 7(b) shows the spectra at \( t = 10 \), where the large scales begin to slightly deviate from the DNS compared to the spectra at \( t = 0.01 \) [Fig. 7(a)].

Figure 8 shows the trajectory of \( n, \phi, \) and \( \zeta \) for the first ten time units at different values of reconstruction in the mHW 256\(^2\) test case. As expected, the distance between the DNS trajectory and the reconstructed decreases as we tighten the tolerance: for \( \varepsilon_{\text{REC}} = 1 \), the solver never calls the StylES algorithm and then a flat (red) line is observed. As we reduce the tolerance \( \varepsilon_{\text{REC}} = 0.5 \), blue line), the algorithm kicks off around 5 \( \omega_d^{-1} \) and, after an initial adjustment, it tends to recover toward the DNS trajectory. Finally, with \( \varepsilon_{\text{REC}} = 0.25 \), the algorithm kicks at 1 \( \omega_d^{-1} \), and the recovery is very fast after 7 \( \omega_d^{-1} \).

Note that \( \varepsilon_{\text{REC}} \) can be seen as a stopping criterion for our algorithm in matching the LES fields and reconstructing the DNS field. The larger \( \varepsilon_{\text{REC}} \) is, the larger the allowed loss function is, as seen in Eq. (10). As noted above if \( \varepsilon_{\text{REC}} = 1 \), then the algorithm is rarely called, and the speed of the procedure matches the implicit LES for the given resolution. As the tolerance decreased, steps 7–11 in Algorithm 1 will take longer to complete, but this will ensure outputs that are increasingly closer to the ground truth DNS. As such, a value of \( \varepsilon_{\text{REC}} \) can be chosen as a compromise with regard to accuracy requirements and/or time constraints for running the algorithm.

**FIG. 5.** Reconstruction of the vorticity field \( \zeta \) from 32\(^2\) to 256\(^2\).
B. Other test cases

Now, we will demonstrate our methodology on the remaining test cases (Table I), using a reconstruction tolerance of 0.25 and highlighting the evolution of four key quantities of interest with time: the total energy $E$, the enstrophy $\zeta$, the particle radial flux $\Gamma_r$, and particle poloidal flux $\Gamma_p$. Note that we use the absolute value of the particle fluxes due to the log scale of the plot.

Figure 9(a) shows these values for the HW 256$^2$ test case: the energy and enstrophy computed by StylES show good agreement with the DNS data and stable solutions. The fluxes are more sensitive and show some oscillations with time in the StylES results. Similar fluctuations in radial and poloidal fluxes were reported by Leddy for several subgrid models. However, some larger differences are found at higher resolution [see Fig. 9(b)], with some marginal recovery later in time. Finally, Fig. 9(c) shows the discrepancies for the mHW 512$^2$, especially in the particle fluxes, most likely due to the wide difference in the initial electrostatic potential, also observable in the energy differences. Error accumulation and oscillations in Figs. 9(a) and 9(b) may be explained by the larger LES numerical discretization error and the
reconstruction error introduced by $\epsilon_{REC} \neq 0$. In the StylES framework, this may be reduced by a lower $\epsilon_{REC}$ tolerance. Alternatively, Sirigano et al. trained their model by solving the LES equations during training and evaluating against DNS.\textsuperscript{34} This gave improved agreement with filtered DNS for decaying isotropic (Navier–Stokes) turbulence, compared to a model trained on filtered DNS data.

C. Performance

The current version of the StylES algorithm has not yet been optimized, i.e., there are many improvements that can be implemented, like: (a) reducing the CPU-GPU communication by overlapping computation with communication; (b) avoiding repeated computations during Arakawa discretization; and (c) using optimized graphs for inference via NVidia TensorRT technology. This work is outside of the scope for the current paper and will be presented in a separate manuscript, for the time being, we only report that a similar speed between DNS and StylES is achieved in the $512^2$ test case, while for lower resolutions, we found the DNS faster and, hence, more convenient to run. However, we have carried out a scalability study to show how the performance will improve with the number of mesh points despite the lack of optimization. In fact, as it is enough to add only one StyleGAN block to double the mesh resolution, our methodology shows an algorithmic complexity of $O(N \log N)$, whereas BOUT++ has a complexity of $O(N^2)$. This can be clearly seen from Fig. 10 where the time per time step for BOUT++ and BOUT++ with StylES are presented. Note that the main contribution to the computational effort in StylES is due to the search into the latent space, which usually occurs within 5 iterations. Some improvements on the convergence speed are also part of the optimization processes mentioned above. Moreover, as most of the relevant plasma turbulence occurs in 3D configuration space, it is the author’s intention to move to a full 3D system. In principle, this would require a 3D GAN, to make sure proper structures of turbulence are learned. However, the number of layers and neurons would grow with the cube of the problem, dramatically limiting the finest resolution. A different approach, instead, would be to rely on a 2D GAN assuming that the turbulent effects are quasi-3D, i.e., filaments of turbulence do not bend on themselves or backward.

FIG. 9. Values of energy $E$ (black), enstrophy $\zeta$ (red), absolute radial flux $|F_r|$ (blue), and absolute poloidal flux $|F_p|$ (green) vs time. Panels show (a) HW $256^2$, (b) HW $512^2$, and (c) mHW $512^2$ test cases. The DNS values are the solid lines, while the StylES values are the dashed lines.

FIG. 10. Scalability of BOUT++ vs StylES with the number of points. Dashed lines indicate the theoretical $N^2$ and $N \log N$ scaling.
possible to reconstruct a 3D structure from a series of 2D independent planes. This would not only avoid the use of a very large NN but also speed up the LES calculation even more compared to the DNS.

IV. CONCLUSIONS

We presented a novel AI deconvolution operator for large eddy simulations of plasma turbulence using the latest state of the art generative adversarial network. We tested the methodology on the Hasegawa–Wakatani and modified Hasegawa–Wakatani equations, for different values of grid resolution and input parameters, and found that we converge toward the correct ground truth DNS dataset as we reduce the tolerance on the reconstruction procedure. However, some discrepancies are found in the higher resolution results, and further investigation has been currently carried out. Future works will be to test larger systems ($\geq1024^3$), optimize, and move to multi-GPU for 3D plasma turbulence simulations relevant to future fusion power plants. However, the main intention of this work is to create a surrogate model to accelerate plasma turbulence solvers, wherever non-linear terms appear and can be obtained via reconstruction. Moving to the Hasegawa–Wakatani 3D will be the next step via some hypotheses highlighted in the Performance section, followed by the upgrade to the Hermes-3 (Ref. 35) model. Then, expanding it to the full divertor geometry, we hope to provide a fast tool to accelerate simulations of edge turbulence.

ACKNOWLEDGMENTS

This work was funded by the FARSCAPE and ESCAPE projects, a collaboration led by UKAEA with the UKRI-STFC Hartree Centre.

AUTHOR DECLARATIONS
Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions


DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES


