StyleGAN as an AI Deconvolution Operator for Large Eddy Simulations of Turbulent Plasma Equations in BOUT++

Project: FARSCAPE III

Jony Castagna – UKRI-STFC Hartree Centre

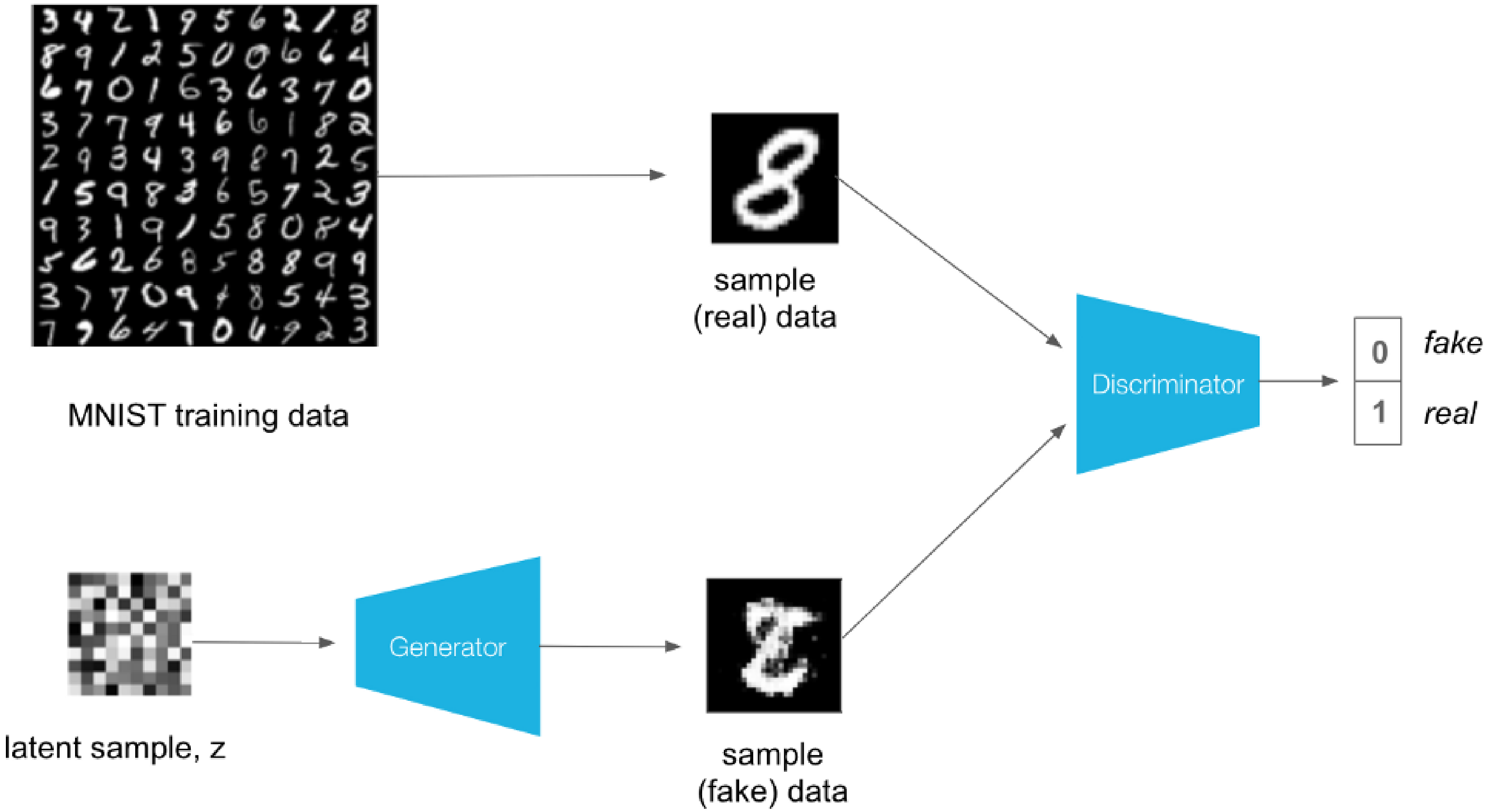
Francesca Schiavello – UKRI-STFC Hartree Centre

Lorenzo Zanisi – UKAEA

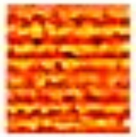
Science and Technology Facilities Council
Hartree Centre

UK Atomic Energy Authority

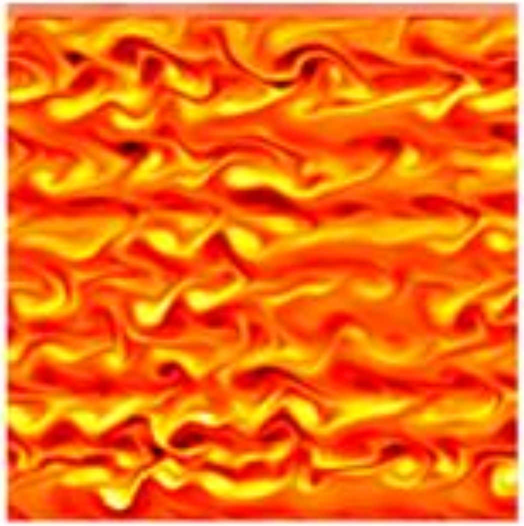
Generative Adversarial Networks (GANs)



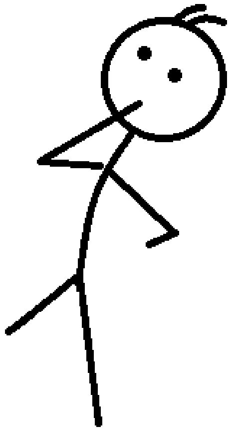
Idea: Can I train a GAN to reconstruct the DNS fields from the internal fields seen as LES fields?



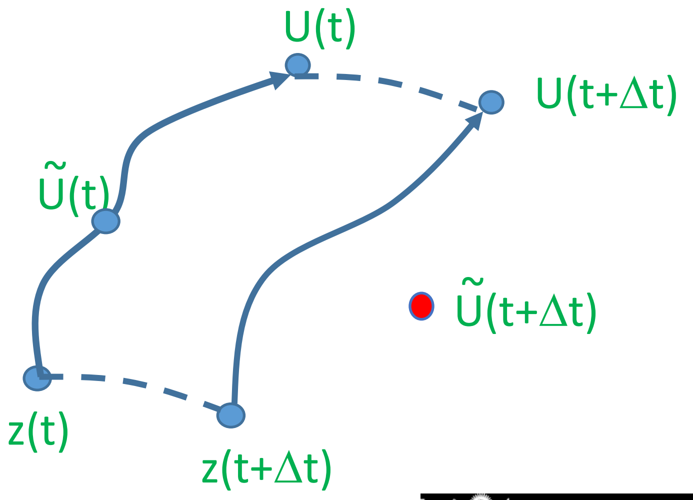
LES field



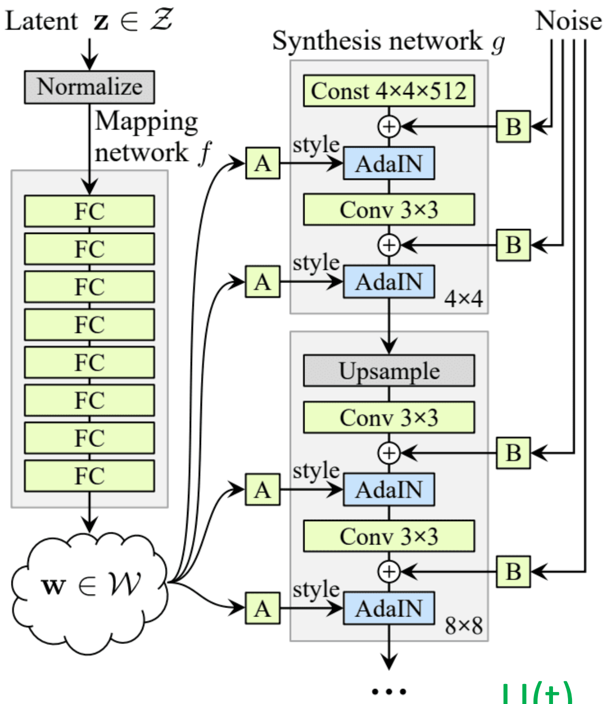
DNS field



Potentially two instantaneous of the same mHW problem can be obtained, **U(t)** and **U(t+Δt)** but there is no guarantee that the internal layers are representation of the same filtered mHW problem, **$\tilde{U}(t)$** and **$\tilde{U}(t+\Delta t)$** !

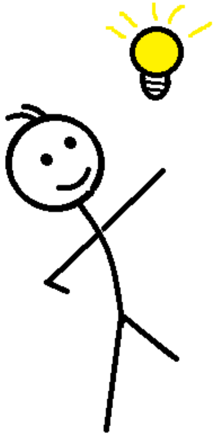


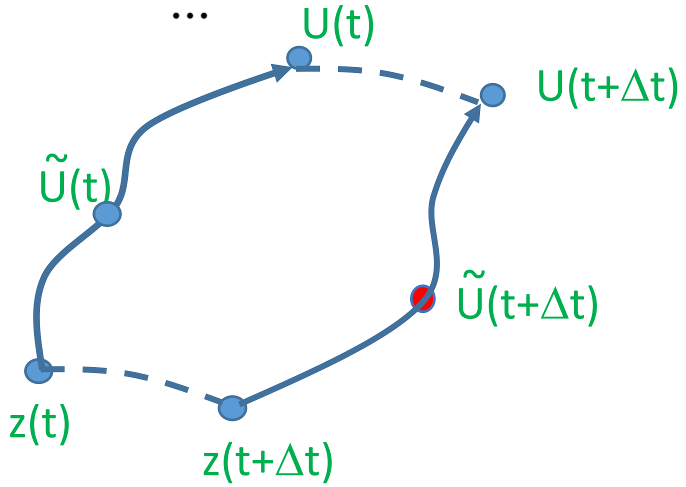
Idea: I need a more “flexible GAN”: StyleGAN!



Our generator thinks of an image as a collection of “styles”, where each style controls the effects at a particular scale

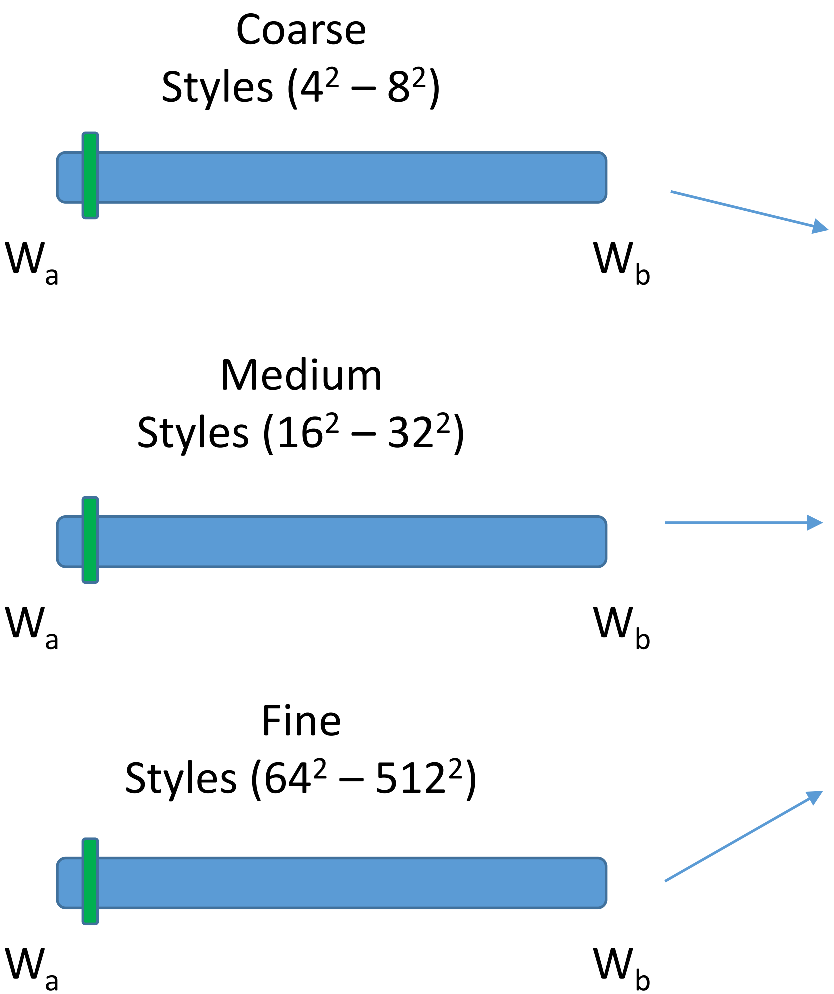
- Coarse styles ⟶ pose, hair, face shape
- Middle styles ⟶ facial features, eyes
- Fine styles ⟶ color scheme

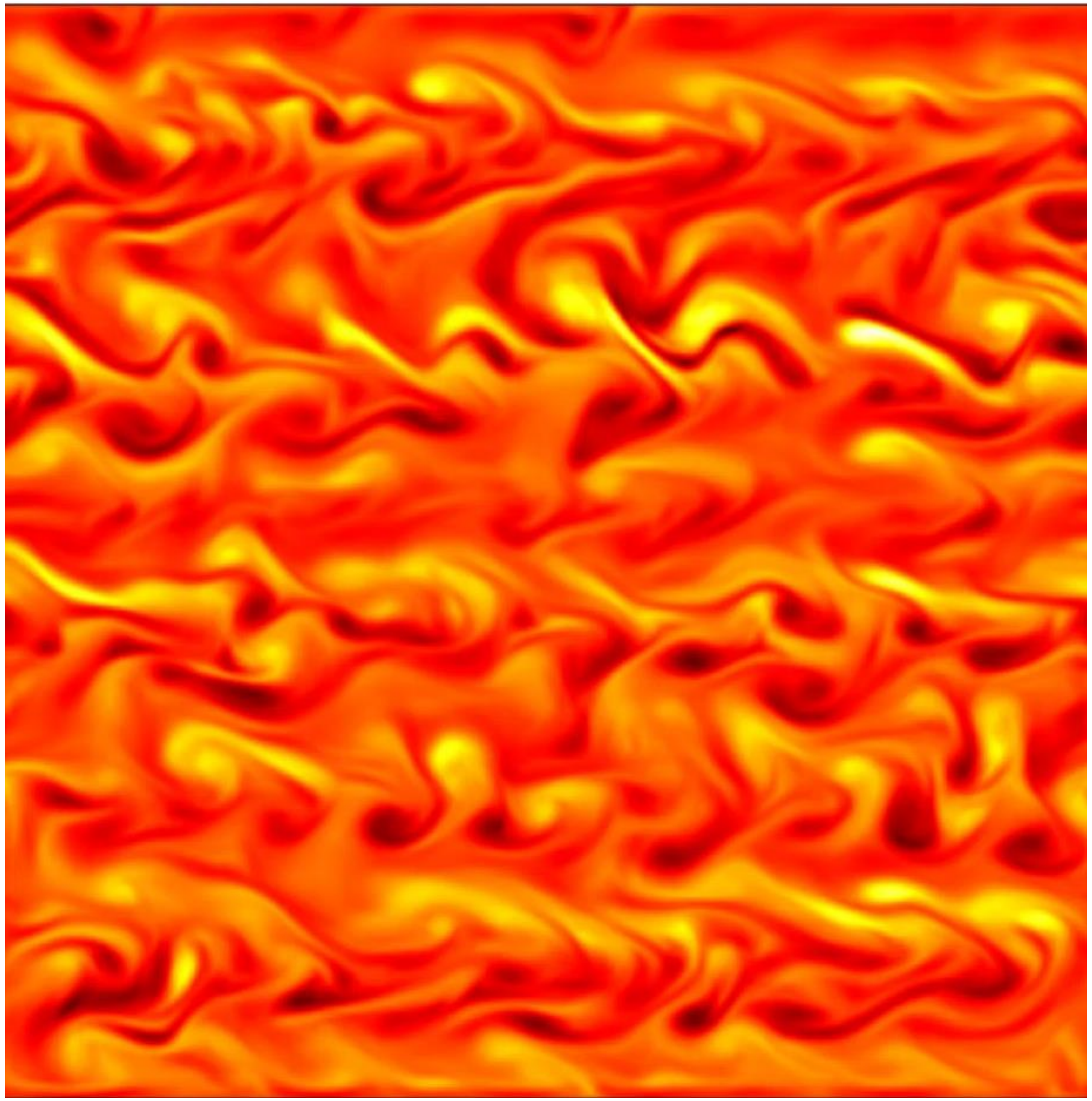




Each layer (style) can be adjusted without interfering with the other levels!

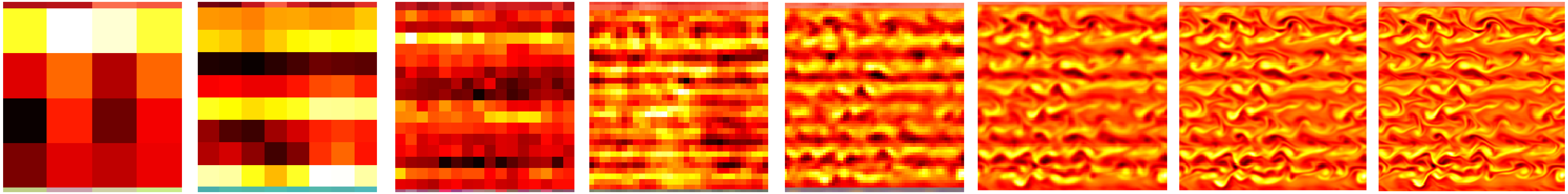
Latent space interpolation





Style Eddy Simulation (StylES)

Different layers of the StylES generator



4x4 | 8x8 | 16x16 | 32x32 | 64x64 | 128x128 | 256x256 | 512x512

Different layer can be "thought" as different filtered LES fields!

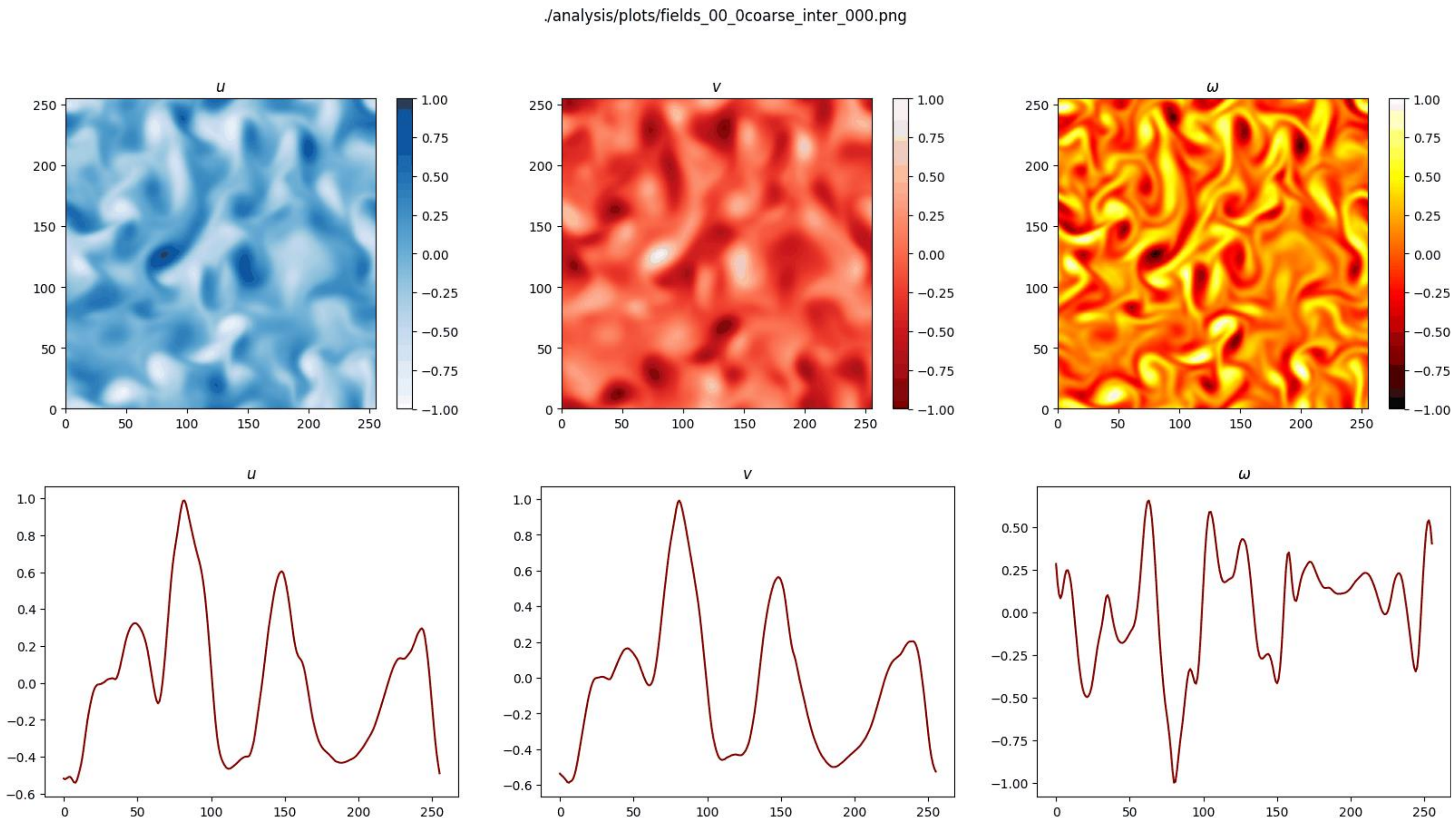
We can use StyleGAN for deconvolution of a LES field and find corresponding DNS field

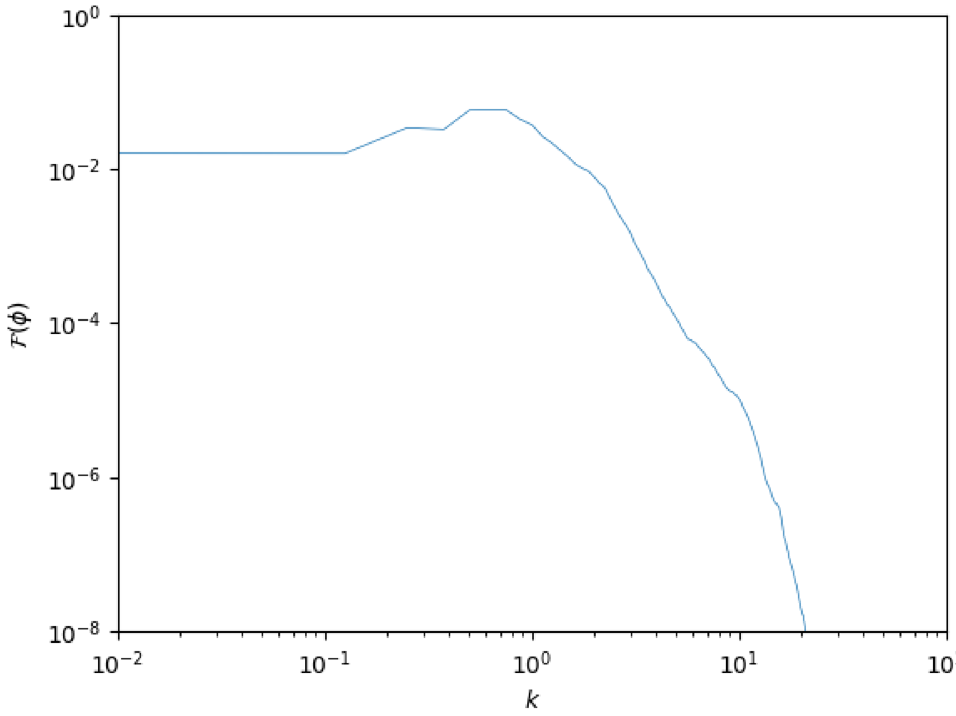
We do not need a RNN!

Main assumptions of the StylES

- Existence and unicity of the unfiltered and filtered equations solution
- A linear interpolation between 2 latent spaces W^+ produces always a valid DNS field
- Continuity and smoothness of the GAN manifold -> research in the latent space always converges! The problem it may take very long…

Interpolation and Energy spectra





A linear interpolation between two latent spaces gives always a DNS spectrum!

Integration with BOUT++

Filtered form of HW equations

$$\frac{\partial \tilde{\zeta}}{\partial t} + \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\zeta}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\zeta}}{\partial y} = \alpha(\tilde{\phi} - \tilde{n}) - \mu_\omega \nabla^4 \tilde{\zeta} + D_{\phi_y \zeta_x} + D_{\phi_x \zeta_y}$$

$$\frac{\partial \tilde{n}}{\partial t} + \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{n}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{n}}{\partial y} = \alpha(\tilde{\phi} - \tilde{n}) - k\frac{\partial \tilde{\phi}}{\partial y} - \mu_n \nabla^4 \tilde{n} + D_{\phi_y n_x} + D_{\phi_x n_y}$$

$\tilde{n} \quad \tilde{\phi} \quad \tilde{\zeta}$

are the LES fields to be passed to StylEGAN running on GPU via TensorFlow

where:

$$\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}} - \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\zeta}}{\partial x} = D_{\phi_y \zeta_x}$$

$$\widetilde{\frac{\partial \phi}{\partial x}\frac{\partial \zeta}{\partial y}} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\zeta}}{\partial y} = D_{\phi_x \zeta_y}$$

$$\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial n}{\partial x}} - \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{n}}{\partial x} = D_{\phi_y n_x}$$

$$\widetilde{\frac{\partial \phi}{\partial x}\frac{\partial n}{\partial y}} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{n}}{\partial y} = D_{\phi_x n_y}$$

LES size fields to be passed back to BOUT++

Integration with BOUT++

```
rLES = findLESTerms(n, phi, vort, pModule, pFindLESTerms);
int N_LES = n.getNz();
int cont=0;
for(int i=2; i<n.getNx()-2; i++)   // we assume 2 guards cells in x-direction
  for(int j=0; j<1; j++)
    for(int k=0; k<n.getNz(); k++){
      Dpyvx(i,j,k) = rLES[cont + 0*N_LES*N_LES];
      Dpxvy(i,j,k) = rLES[cont + 1*N_LES*N_LES];
      Dpynx(i,j,k) = rLES[cont + 2*N_LES*N_LES];
      Dpxny(i,j,k) = rLES[cont + 3*N_LES*N_LES];
      cont = cont+1;
    }

ddt(n) = -Dn*Delp4(n) + Dpyvx + Dpxvy;
ddt(vort) = -Dvort*Delp4(vort) + Dpynx + Dpxny;
```

call a function with an Embedded Python call

pass back 1D numpy array to BOUT++

add sub-grid scale terms

hw.cxx file in Hasegawa-wakatani example

https://github.com/farscape-project/BOUT-dev.git

branch: bout_with_StylES

Issues (I)

...but:

$$\nabla^2 \phi = \zeta$$
$$\nabla \cdot (\nabla \phi) = \zeta$$
$$\nabla \cdot (-\mathbf{E}) = \zeta$$

but for the quasi-neutrality condition:

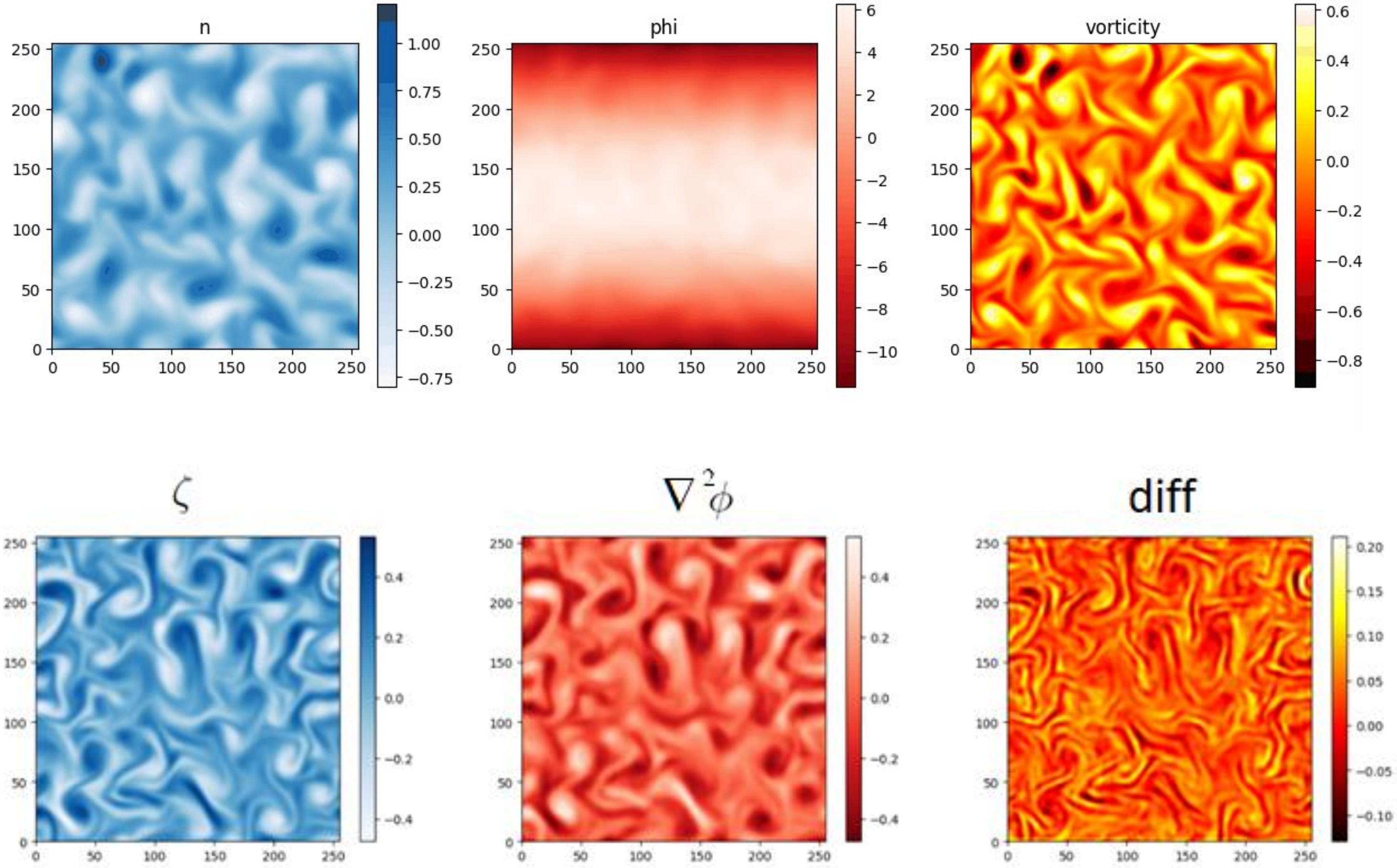
$$\nabla \cdot \mathbf{E} = 0$$ (over the full domain!)

from which follows:

$$\int_{\mathcal{D}} \zeta = \int_{\mathcal{D}} \phi = 0$$

and for mass conservation:

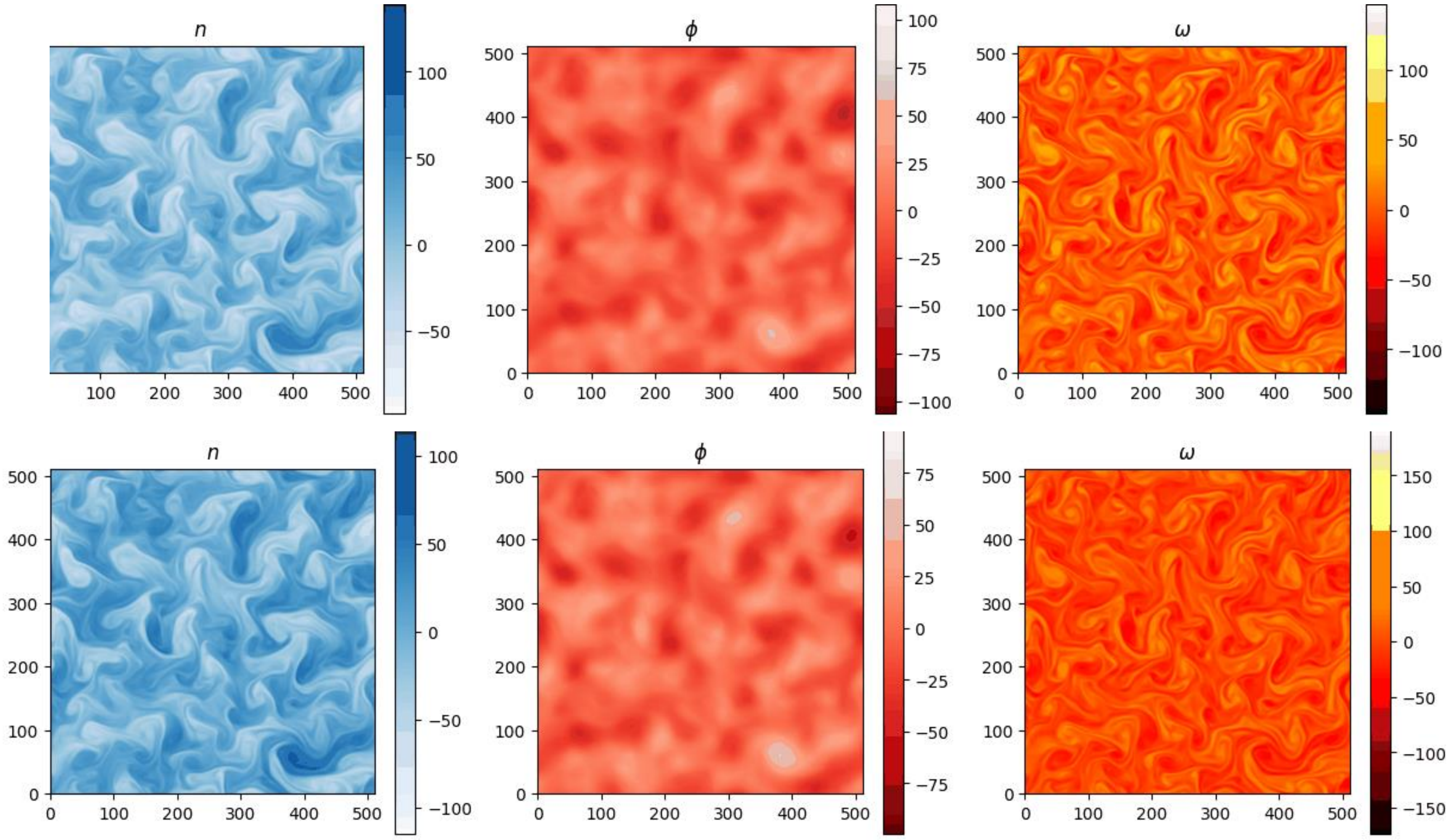
$$\int_{\mathcal{D}} n = 0$$



Issues (I): solved!

Without correction a net flow appears…

$$\zeta' = \zeta - \bar{\zeta}$$
$$\phi' = \phi - \bar{\phi}$$
$$n' = n - \bar{n}$$



We can now use StyleGAN to generate an initial condition for the simulation!

Issues (II)

$\tilde{n}$ $\tilde{\phi}$ $\tilde{\zeta}$ $\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}}$

unpack **7** 1D numpy arrays from BOUT++ (CPU, 64bit)

$\tilde{n}$ $\tilde{\phi}$ $\tilde{\zeta}$

TensorFlow arrays (GPU, 32bit)

n ϕ ζ

reconstruct DNS fields via search in the latent space (GPU)

$$\frac{\partial n}{\partial x}, \frac{\partial n}{\partial y}, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \zeta}{\partial x}, \frac{\partial \zeta}{\partial y}$$

find derivatives
2nd order central scheme
(GPU)

$$\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}, \frac{\partial \phi}{\partial x}\frac{\partial \zeta}{\partial y}, \frac{\partial \phi}{\partial y}\frac{\partial n}{\partial x}, \frac{\partial \phi}{\partial x}\frac{\partial n}{\partial y}$$

product as derivatives (GPU)

$$\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}} \quad \widetilde{\frac{\partial \phi}{\partial x}\frac{\partial \zeta}{\partial y}} \quad \widetilde{\frac{\partial \phi}{\partial y}\frac{\partial n}{\partial x}} \quad \widetilde{\frac{\partial \phi}{\partial x}\frac{\partial n}{\partial y}}$$

$$\widetilde{\frac{\partial \phi}{\partial y}} \; \widetilde{\frac{\partial \zeta}{\partial x}} \quad \widetilde{\frac{\partial \phi}{\partial x}} \; \widetilde{\frac{\partial \zeta}{\partial y}} \quad \widetilde{\frac{\partial \phi}{\partial y}} \; \widetilde{\frac{\partial n}{\partial x}} \quad \widetilde{\frac{\partial \phi}{\partial x}} \; \widetilde{\frac{\partial n}{\partial y}}$$

filter all quantities (GPU)

$$\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}} - \widetilde{\frac{\partial \phi}{\partial y}}\,\widetilde{\frac{\partial \zeta}{\partial x}} = D_{\phi_y \zeta_x}$$

...

find sub-grid scale terms (GPU)

$D_{\phi_y \zeta_x}$, $D_{\phi_x \zeta_y}$, $D_{\phi_y n_x}$, $D_{\phi_x n_y}$

pack to **4** 1D numpy arrays for BOUT++
(CPU, cast to 64bit)

Issues (II): Arakawa Discretization

$$\frac{\partial\phi}{\partial y}\frac{\partial\zeta}{\partial x},\ \frac{\partial\phi}{\partial x}\frac{\partial\zeta}{\partial y},\ \frac{\partial\phi}{\partial y}\frac{\partial n}{\partial x},\ \frac{\partial\phi}{\partial x}\frac{\partial n}{\partial y}$$

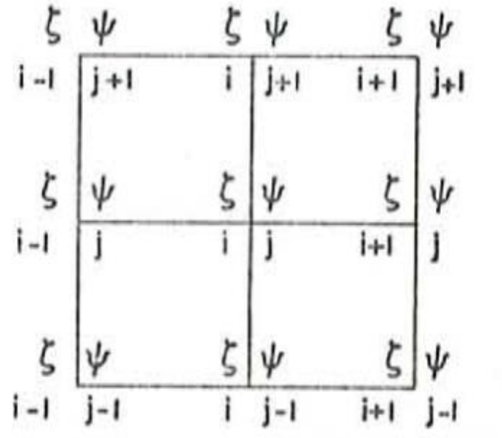
$$\widetilde{\frac{\partial\phi}{\partial y}\frac{\partial\zeta}{\partial x}}\quad \widetilde{\frac{\partial\phi}{\partial x}\frac{\partial\zeta}{\partial y}}\quad \widetilde{\frac{\partial\phi}{\partial y}\frac{\partial n}{\partial x}}\quad \widetilde{\frac{\partial\phi}{\partial x}\frac{\partial n}{\partial y}}$$

$$\widetilde{\frac{\partial\phi}{\partial y}}\ \widetilde{\frac{\partial\zeta}{\partial x}}\ \widetilde{\frac{\partial\phi}{\partial x}}\ \widetilde{\frac{\partial\zeta}{\partial y}}\ \widetilde{\frac{\partial\phi}{\partial y}}\ \widetilde{\frac{\partial n}{\partial x}}\ \widetilde{\frac{\partial\phi}{\partial x}}\ \widetilde{\frac{\partial n}{\partial y}}$$

$$\widetilde{\frac{\partial\phi}{\partial y}\frac{\partial\zeta}{\partial x}} - \widetilde{\frac{\partial\phi}{\partial y}}\,\widetilde{\frac{\partial\zeta}{\partial x}} = D_{\phi_y\zeta_x}$$

...

produces oscillations which violate conservation laws



$$J = \alpha J_1 + \gamma J_2 + \beta J_3, \qquad \alpha + \gamma + \beta = 1,$$

$$\begin{aligned} J_4 &= \tfrac{1}{2}(J_1 + J_2),\\ J_5 &= \tfrac{1}{2}(J_2 + J_3),\\ J_6 &= \tfrac{1}{2}(J_3 + J_1),\\ J_7 &= \tfrac{1}{3}(J_1 + J_2 + J_3). \end{aligned}$$

A. Arakawa and V. Lamb, 1977 (Methods in Computational Physics)

Issues (II): solved!

- For the Hasegawa-Wakatani systems

$$\frac{\partial \xi}{\partial t} = -\{\phi, \xi\}$$

$$\frac{\partial \xi}{\partial t} = -\left(\frac{\partial \phi}{\partial y}\frac{\partial \xi}{\partial x} - \frac{\partial \phi}{\partial x}\frac{\partial \xi}{\partial y}\right)$$

$$\frac{\partial \tilde{\xi}}{\partial t} = -\left(\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \xi}{\partial x}} - \widetilde{\frac{\partial \phi}{\partial x}\frac{\partial \xi}{\partial y}}\right) \quad \text{from linearity property of filter } G$$

$$\frac{\partial \tilde{\xi}}{\partial t} = -\left(\frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\xi}}{\partial x} + D_{\phi_y \xi_x} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\xi}}{\partial y} - D_{\phi_x \xi_y}\right) \quad \text{where } D = \widetilde{f g} - \tilde{f}\tilde{g}$$

$$\frac{\partial \tilde{\xi}}{\partial t} = -\left[\underbrace{\frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\xi}}{\partial x} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\xi}}{\partial y}}_{\{\tilde{\phi}, \tilde{\xi}\}} + \left(D_{\phi_y \xi_x} - D_{\phi_x \xi_y}\right)\right]$$

$$\frac{\partial \tilde{\xi}}{\partial t} = -\{\tilde{\phi}, \tilde{\xi}\} - \left(D_{\phi_y \xi_x} - D_{\phi_x \xi_y}\right)$$

but:

$$D_{\phi_y \xi_x} = \widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \xi}{\partial x}} - \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\xi}}{\partial x} \; ; \; D_{\phi_x \xi_y} = \widetilde{\frac{\partial \phi}{\partial x}\frac{\partial \xi}{\partial y}} - \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\xi}}{\partial y}$$

then:

$$D_{\phi_y \xi_x} - D_{\phi_x \xi_y} = \widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \xi}{\partial x}} - \frac{\partial \tilde{\phi}}{\partial y}\frac{\partial \tilde{\xi}}{\partial x} - \widetilde{\frac{\partial \phi}{\partial x}\frac{\partial \xi}{\partial y}} + \frac{\partial \tilde{\phi}}{\partial x}\frac{\partial \tilde{\xi}}{\partial y}$$

$$\widetilde{\{\phi, \xi\}} \qquad -\{\tilde{\phi}, \tilde{\xi}\}$$

Finally

$$\frac{\partial \tilde{\xi}}{\partial t} = -\cancel{\{\tilde{\phi}, \tilde{\xi}\}} - \widetilde{\{\phi, \xi\}} + \cancel{\{\tilde{\phi}, \tilde{\xi}\}} = -\widetilde{\{\phi, \xi\}} \quad \text{as expected!}$$

Calculate the Poisson bracket term and then filter it!!

$$-\overbrace{\cancel{\{\tilde{\phi}, \tilde{\xi}\}}}^{\text{CPU}} - \overbrace{\widetilde{\{\phi, \xi\}} + \cancel{\{\tilde{\phi}, \tilde{\xi}\}}}^{\text{GPU}} = -\overbrace{\widetilde{\{\phi, \xi\}}}^{\text{GPU}}$$

BOUT++ does not need to calculate the Poisson bracket term anymore!

Issues (II): solved!

skip Poisson brackets on BOUT++!

$\tilde{n} \quad \tilde{\phi} \quad \tilde{\zeta} \quad \cancel{\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}}}$

unpack **3** 1D numpy arrays from BOUT++ (CPU, 64bit)

$\tilde{n} \quad \tilde{\phi} \quad \tilde{\zeta}$

TensorFlow arrays (GPU, 32bit)

$n \quad \phi \quad \zeta$

reconstruct DNS fields via search in the latent space (GPU)

$$\frac{\partial n}{\partial x}, \frac{\partial n}{\partial y}, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \zeta}{\partial x}, \frac{\partial \zeta}{\partial y}$$

find derivatives
2^{nd} order central scheme
(GPU)

$$\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}, \frac{\partial \phi}{\partial x}\frac{\partial \zeta}{\partial y}, \frac{\partial \phi}{\partial y}\frac{\partial n}{\partial x}, \frac{\partial \phi}{\partial x}\frac{\partial n}{\partial y}$$

product as derivatives (GPU)

$$\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x} \quad \frac{\partial \phi}{\partial x}\frac{\partial \zeta}{\partial y} \quad \frac{\partial \phi}{\partial y}\frac{\partial n}{\partial x} \quad \frac{\partial \phi}{\partial x}\frac{\partial n}{\partial y}$$

find DNS Poisson brackets (GPU)

$$\widetilde{\frac{\partial \phi}{\partial y}\frac{\partial \zeta}{\partial x}}$$

filter them (GPU)

pack to **4** 1D numpy arrays for BOUT++
(CPU, cast to 64bit)
and add into the convective (explicit term)

Moreover…

You don’t need to call StyleGAN if…

- $\|f(U) - \tilde{U}^{GAN}\| < \varepsilon$
- simulation time (simtime) is the same (not sure how pvode works…)

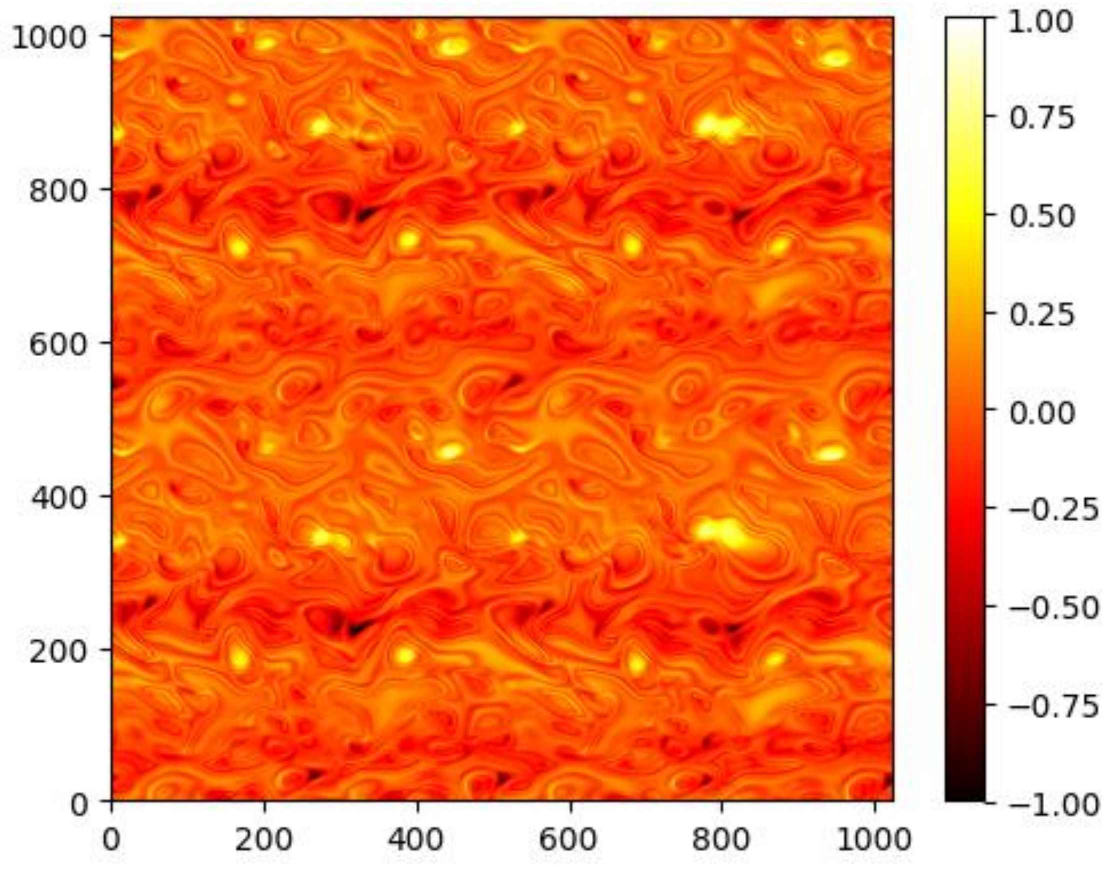
$$\left.\right\} \{\phi,\varpi\}(t+\Delta t) = \{\phi,\varpi\}(t)$$

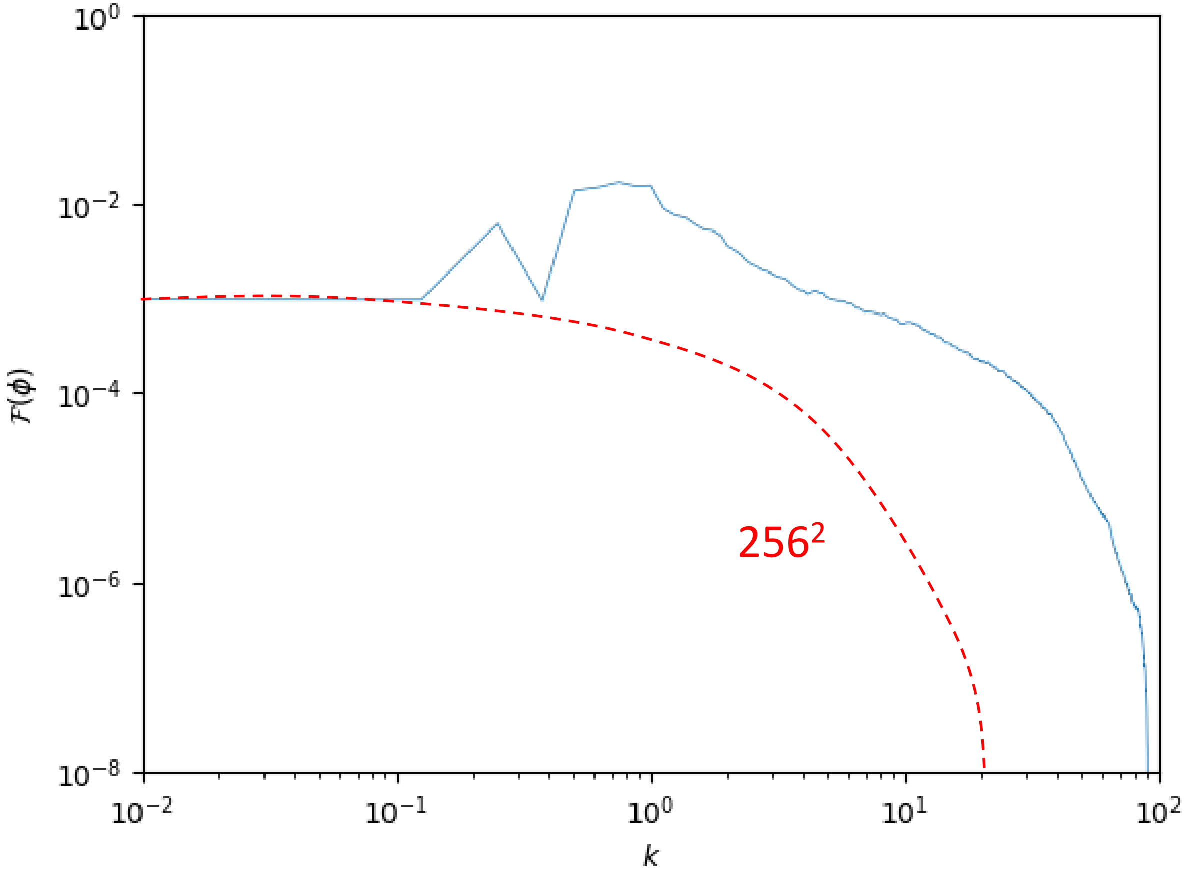
And:

- StyleGAN output are normalized → rescaling needed
- warm-up time (~100 steps) to fix mismatch $\nabla_\perp^2 \phi \neq \zeta$

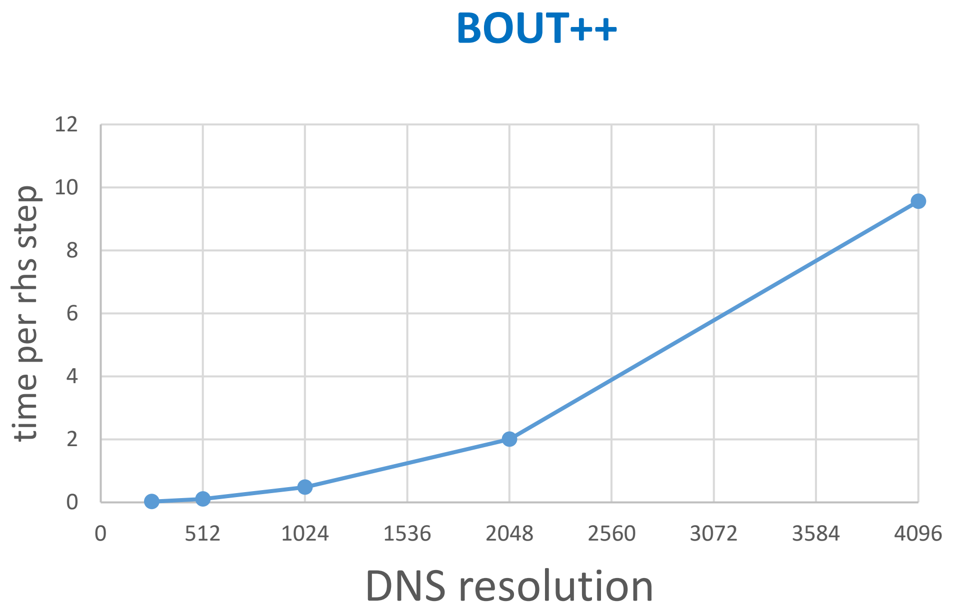
Running 1024 mHW

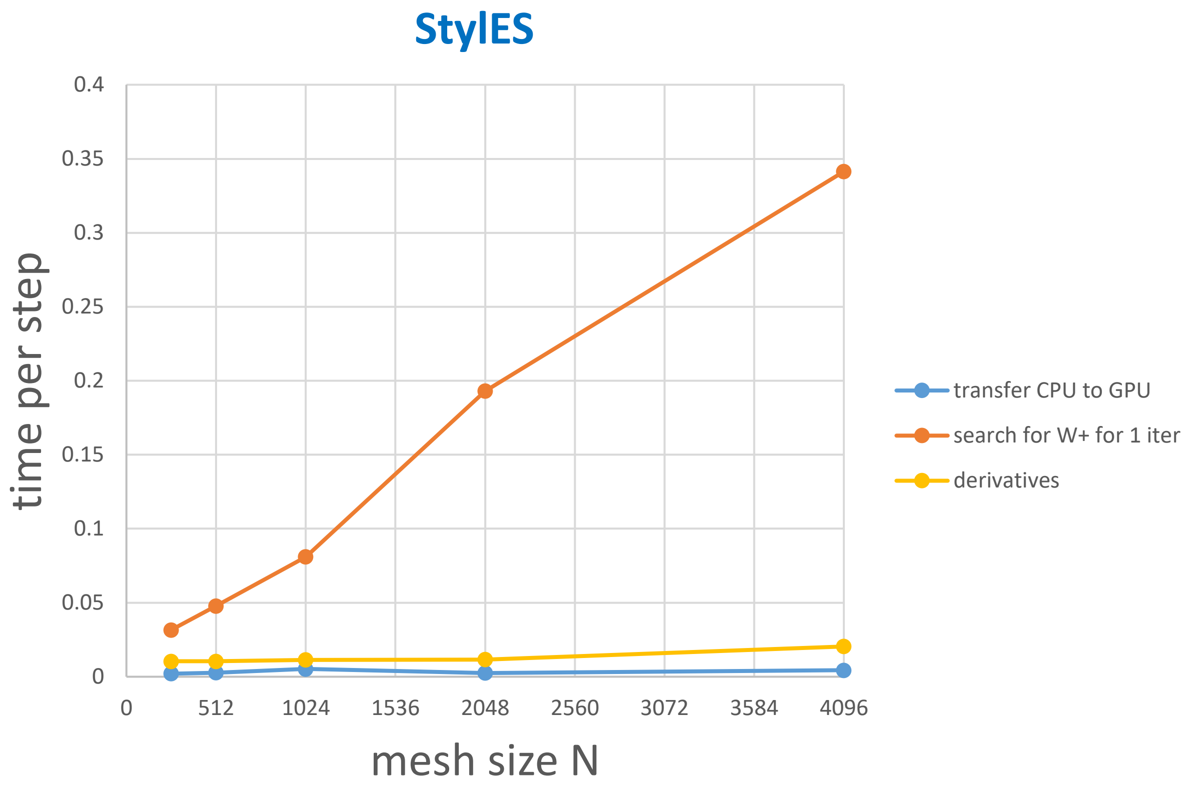
Same as N=256, but $\nu_{\varpi} = \nu_n = 10^{-6}$





Performance





StylES is ~ 10% of the DNS!

Resume (I)

- We introduced a novel surrogate model based on latest Generative Adversarial Networks (**GANs**) for turbulent flow simulations

- This allows to **avoid** the train of a **RNN** for a time integration

- We do **not** use physic constrains **yet**, as these are inherited via the filter operator

- Integration with **BOUT++** is completed and results are now being gathered from 256x256 to 1024x1024

Future work

- We need to optimize and parallelize the integration to multiGPU using LBANN
- Extension to full divertor geometry
- Extension to 3D-HW (as a series of 2D planes along z…)