Reconstruction of fine grid CFD solution from a coarse grid one (Machine Learning 2022 Course)

Maksimilian Pavlov Semyon Polyansky Dmitry Belousov Olga Gorbunova Egor Rumiantsev

Abstract

Certain problems, for instance, timestepping of the Navier-Stokes equation, can be very computationally expensive to solve with desirable accuracy. Normally this is ameliorated by using e.g. multigrid methods, which are, however, unusable with commercial solvers. Another alternative is to employ the method of Galerkin and, upon isolating the modes of some snapshot matrix of fine-grid solutions, compute further on coarse grid with projection onto said modes, providing enhancement of approximation and not impairing usability with commercial solvers. For this task two methods are salient: SINDy (sparse identification of nonlinear dynamics), which is in essence regression on functional basis, and employment of generative networks and autoencoders. Study of these is the goal.

Github repo: https://github.com/Genndoso/Reconstruction-of-fine-grid-CFD-solution-from-a-coarse-grid-one
Video presentation: https://cutt.ly/iSPtUwa

1. Introduction

The first method under consideration (aside from the Galerkin approximation that is used purely as a proven benchmark to test against) entails constructing a library of candidate functions of the input data (solution on a coarse grid with interpolation on intermediate fine-grid points) and doing regularised (to promote sparsity) regression on those with the goal of reconstructing a fine-grid solution. An alike method was successfully used to capture nonlinear dynamics in data and even reconstruct parameterised PDEs, so the method appears feasible. The second method under consideration entails collecting solutions on coarse and fine grids (this time without the strict need to interpolate the coarse-grid one) and training a GAN to reconstruct the latter from the former. Of particular interest is examining if it would be enough to supply individual time snapshots instead of snapshot matrices. These two are planned to be tested on a number of problems concerned with nonlinear dynamics against each other and a benchmark Galerkin approximation.

1.1. Data

We have considered three problems which we will now state in the language of magnetohydrodynamics, as they are all of MHD or amagnetic hydrodynamic nature. It must be stated that in general magnetohydrodynamical problems are more amenable to grid coarsening due to the magnetic field's influence smoothing out the instabilities (comparative instabilities w/ and w/o B in Figure 1), although problems that correspond to situations of e.g. intersecting characteristics (like problem of Orszag and Tang) still provide a good target for our purposes.

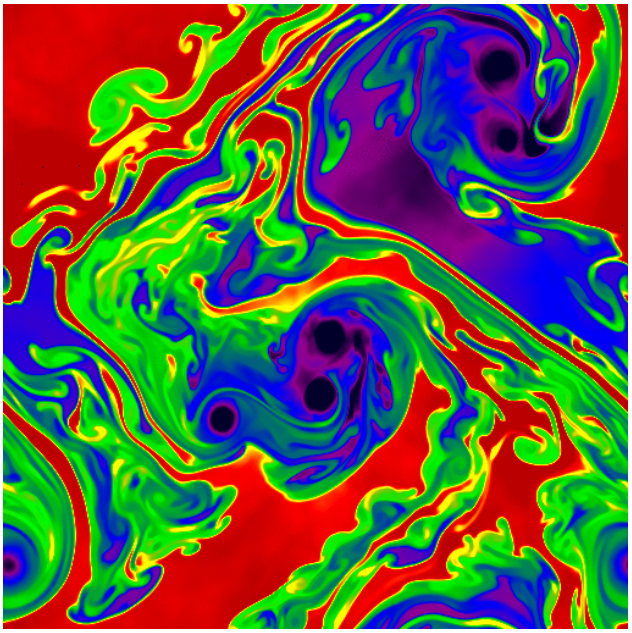

Figure 1. Kelvin-Helmholtz Uninhibited Instability

We firstly state the ideal magnetohydrodynamical laws in whose framework the problems are stated (Batchelor, 1967), with the customary usage of bold variables to indicate vector quantities:

$$\rho_t = -\nabla \cdot (\rho \boldsymbol{v}) \tag{1}$$

$$(\rho \boldsymbol{v})_t = -\nabla \cdot [\rho \boldsymbol{v}\boldsymbol{v}^T + (p + \frac{1}{2}\|\boldsymbol{B}\|)\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^T] \tag{2}$$

$$\boldsymbol{B}_t = \nabla \times (\boldsymbol{v} \times \boldsymbol{B}) \tag{3}$$

$$e_t = -\nabla \cdot [(\frac{\gamma}{\gamma - 1}p + \frac{1}{2}\rho\|\boldsymbol{v}\|)\boldsymbol{v} - (\boldsymbol{v} \times \boldsymbol{B}) \times \boldsymbol{B}] \tag{4}$$

With e being the internal energy, the magnetic field being purely solenoidal as per the law of Gauss (this condition is secured by means of initial condition and ususally preserved, but has to be occasionally enforced e.g. in Orszag-Tang problem by way of Leray projection onto solenoidal space), γ being specific heat ratio, all norms being 2-norms, and internal energy being coupled wtih pressure by way of the following:

$$e = \frac{1}{2}\rho\|\boldsymbol{v}\| + \frac{1}{2}\|\boldsymbol{B}\| + \frac{p}{\gamma - 1} \tag{5}$$

We can now proceed to formulate the problems that were considered, as upon setting the magnetic flux density to identically zero regular hydrodynamic formulation is obtained.

1.1.1. Problem 1. Karman vortex street.

It is known that when an obstacle is inserted in a stream of liquid (the classical example is a cylindre), violent vortex shedding will be observed, leading to a formation of the so-called Karman vortex street in the object's wake. This is a classical problem in CFD, and we consider it in absence of any magnetic fields, however curious their effect may be. The geometry is given on Figure 2. The discretisation frequencies are 32 and 16 elements across geometry's diameter for fine and coarse grid respectively. The walls impose zero-velocity no-slip condition, and the pressure at the outflow is constant and zero. Furthermore,

$$p = 0; \Gamma_1 = [2.2; y]; y \in [0; 0.41]$$

$$u(x, y, t) = [1.5 \cdot \frac{4y(0.41 - y)}{0.41^2}, 0]; \Gamma_2 = [0; y]; y \in [0; 0.41]$$

$$u = [0; 0]; \Gamma_3 = \partial\Omega \backslash (\Gamma_1 \cup \Gamma_2)$$

Of the parameters we take:

$$\rho = 1$$

$$\mu = [0.001, 0.00116, \cdots, 0.005]$$

1250 timesteps from $t = 0$ to $t = 1.25$ were considered.

The particular interest of this model is the ability to reconstruct vortices that may perhaps be smoothed out to a great degree. Solutions for the problem were obtained with the method of incremental pressure correction due to Goda (Goda, 1979), and the coarse-grid solution is interpolated with cubic splines as a pre-processing step. This interpolation step is performed for all problems and therefore not stated in other subsections.

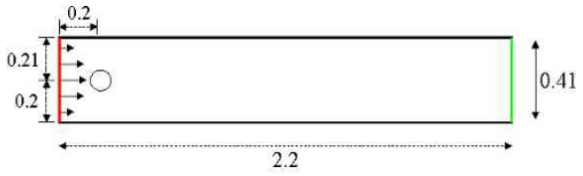

Figure 2. Geometry on which solution was obtained

1.1.2. Problem 2. Orszag-Tang vortex

The problem of Orszag and Tang is one of the benchmarks in computational magnetohydrodynamics and corresponds to the onset of supersonic turbulence (Orszag & Tang, 1979). Both lone shocks and shock-shock interactions are present in the system's evolution, thereby offering rather rich dynamics. The problem was solved on square domain $\Omega = [-0.5, 0.5]^2$ and time interval $[0; 0.8]$ with 400 elements in either direction for fine and 200 - for coarse mesh. The parameter was chosen to be the specific heat ratio, varied between known values. Boundary conditions were employed: periodic on both axes, initial conditions specified below.

$$\rho_0 = \frac{25}{36\pi} \tag{6}$$

$$p_0 = \frac{5}{12\pi} \tag{7}$$

$$\boldsymbol{V}_0 = [-\sin(2\pi y) \;\; \sin(2\pi x)]^T \tag{8}$$

$$\beta_0 = \frac{1}{\sqrt{4\pi}} \tag{9}$$

$$\boldsymbol{B}_0 = [-\beta_0 \sin(2\pi y) \;\; \beta_0 \sin(4\pi x))]^T \tag{10}$$

The solution was carried out by means of the Roe solver, as it provides the best accuracy among all Riemann solvers. At each step additionally the solenoidal constraint had to be enforced because Godunov-type methods can violate it. To this end we invoke the Helmholtz decomposition theorem, which in MHD has the form of the so-called Leray projection onto the solenoidal component.

1.1.3. Problem 3. Kelvin-Helmholtz instability in presence of magnetic field

This is also one of the known benchmarks in CMHD for reasons of the magnetic field's smoothing effect on instabil-

ities (Frank et al., 1996). As observed by Chandrasekhar, any magnetic field not perfectly normal to the direction of streaming has the effect equivalent to surface tension, which is known to inhibit instability development [chandra-instability]. Of especial interest is also the qualitative property of the problem that allows to judge the numerical diffusion effect of the scheme and discretisation, for it contributes significantly to suppression of instability. This therefore poses an additional problem of reconstructing possibly-suppressed instability from smoothed coarse grid.

The problem was solved on square domain $\Omega = [-0.5, 0.5]^2$ and time interval $[0; 5]$ with 256 and 128 elements in the direction of streaming and its orthogonal complement for fine and twice less than that - for coarse mesh. Boundary conditions are periodic everywhere. The parameter was taken to be the viscosity of the fluid in the centre. Additionally:

$$V_x^0 = \{-0.5; |y| < 0.25; 0.5; |y| \geq 0.25\} \tag{11}$$

$$\rho_0 = \{1; |y| < 0.25; 2; |y| \geq 0.25\} \tag{12}$$

$$p_0 = 2.5 \tag{13}$$

$$\gamma = 1.4 \tag{14}$$

$$\boldsymbol{B}_x^0 = \frac{1}{2\sqrt{4\pi}} \tag{15}$$

The instability is further seeded by a random velocity perturbation with amplitude no greater than 0.01.

Solution was carried out by means of the Harten-Lax-van Leer Discontinuities solver, picked for positive conservation of pressure and density. The solution appears to be inhibited quite quickly and is therefore taken as the easiest one to reconstruct.

2. Related work

In their work (P. Pant, 2021) created a Deep learning neural network (DL-ROM) which are capable of non-linear projections to reduced order states in tasks of fluid dynamics. Consequently, the reduced order state was used to efficiently predict future time steps of the simulation using 3D Autoencoder and 3D U-Net based architectures. They got DL-ROM yields great reconstruction results for this simulation by maintaining low values of MSE over a span of 20 iterations. State of the art model performance showed highly accurate of the higher order state, by minimizing the reconstruction error with the ground truth simulation of the future time step.

(Kigure, 2021) developed Pix-to-Pix generative adversarial network (GAN), specifically video-to-video translation network, to computational fluid dynamics (CFD) simulations. The architecture of GANs in this research is a combination of the image-to-image translation networks (the so-called "pix2pix") and Long Short-Term Memory (LSTM). The obtained results shows that deep learning models can be implemented in order to reduce computational costs of simulation for rough approximation.

(Chakir & Maday, 2009) developed a Galerkin-type method specifically for relaxing the demands of solving parametric PDEs by means of projecting coarse-grid approximations that can be substantially cheaper (especially in higher dimensions) on a subspace spanned by a certain number of fine-grid solutions obtained previously.

3. Algorithms and Models

3.1. NIRB

Github repo: https://github.com/Genndoso/Reconstruction-of-fine-grid-CFD-solution-from-a-coarse-grid-one

The baseline algorithm hinges on the ideas of multigrid solution and Galerkin projection. The classical multigrid is best introduced with an easily analysed smoothing iterative method, such as the method of Jacobi. Taking the method of Jacobi (or, indeed, any matrix-splitting method which takes $A = M - N$ with the iteration matrix $G = M^{-1}N$) for some toy problem of kind $u'' = f$, we have for error $e_i = Ge_{i-1}$, and for the eigenvectors of G (therefore also of A) harmonics with harmonics for corresponding eigenvalues (in fact, cosines). Expanding the error in these eigenbasis, we will see that each component shall have its own convergence rate, determined by the corresponding cosine eigenvalue, which are unequal and obviously closer to unity for lower-frequency components. This can also be understood as smoothing in the sense of averaging having less impact on lower-frequency errors. In this vein, multigrid is introduced as a method that takes advantage of the fact that some components (that are of high frequency) are damped quickly; it can, in fact, be possible to make a careful enough choice of an iterative method which ensures that the entire upper half frequency range is quickly damped. Once in a few iterations the high-frequency components are suitably damped, we therefore are left with a smoothed problem with a smoothed error. Since only errors of long wavelength now remain, we may recast the problem on a coarser grid without injuring the faithfulness of the error. But that would bring some of the previously low-frequency components into the high-frequency range! Consider, for instance, a grid with n points. By the theorem of Kotelnikov, the upper frequency bound is $\frac{n}{2}$, with the rapid-damping cutoff being at best $\frac{n}{4}$. If we now take a grid with half as many points, both these values are also cut in half, and a full quarter of the total frequency range is made available for quick smoothing. Even this simple method offers a good convergence booster, yet it possesses a fatal flaw that renders it scarcely usable for many engineering

tasks. That flaw is its intrusivity. Indeed, we need to apply different operators, alternate between grids and even at times solve a restated version of the problem. This is not difficult if one possesses full freedom, but such freedom is often not afforded by the commercial solvers. We are therefore compelled to also seek non-invasive methods.

Chakir and Maday propose one such method in the following way: intuitively describing, after a number of iterations is made on the fine grid, further computations are all done on the coarse grid with the demand that the residuals of approximation of the fine-grid solution by the coarse-grid one be made orthogonal to the space of the computed fine-grid snapshot (or some subspace thereof)(Chakir & Maday, 2009)(CHAKIR et al., 2013)(Chakir et al., 2018). That is achieved by first obtaining an orthogonal basis of either the entire aforementioned space or some subspace of it and then projecting the coarse-grid solutions onto that basis. More formally, the algorithm can be described in the following way 1:

One sees the coefficient regularisation procedure in the algorithm presented. It can be omitted for ease of &computation and replaced with a simple projection onto the basis obtained, but it will yield generally inferior approximations. Firstly, we note that we can improve the simple procedure without regularisation by ensuring it returns for parameters for which we possess fine-grid solutions exactly those solutions. e also note that the more solutions we compute, the greater the degree to which they are linearly dependent, i.e. the worse the condition number of the matrix corresponding to a transform which sends the M aforementioned solutions to their fine-grid values precisely. To achieve the stated goal and remedy the poor conditioning we employ a linear regularised regression problem which is stated in the following way, intuitively understood as seeking an operator that best fits coarse-grid projective coefficients onto the known fine-grid ones, with some regularisation introduced to combat poor matrix conditioning:

$$\alpha_i^j =< u_h^i, \phi_j > \tag{16}$$

$$\beta_i^j =< I_H^h u_H^i, \phi_j > \tag{17}$$

$$\psi_i = \arg\min_{\psi_i}(\|\boldsymbol{\beta}\psi_i - \alpha_i\| + \lambda\|\psi_i\|) \tag{18}$$

Equation (18) may be solved by means of a descent algorithm or, provided that the matrix β is small, directly by means of the normal equation:

$$\psi_i = (\beta^\dagger\beta + \lambda I)^{-1}\beta_i\alpha_i \tag{19}$$

Regardless of the way by which (18) is solved, we obtain the rectified projection:

$$\tilde{\beta}_i^j = \sum_j \psi_{i,j} < I_H^h u_H^i, \phi_j > \tag{20}$$

And then

$$\hat{u}_h^i = \psi_{j,k} < I_H^h u_H^i, \phi_k > \phi_j \tag{21}$$

This is precisely the method described in the algorithm. It must also be noted that a number of improvements can and have been introduced by us, but they are relatively minor and pertain to obtaining the data, a part of workflow which does not strictly enter in the algorithm outlined in 1 and is treated there as a black box.

The principal improvements that are applicable here (since the others were devised for multiparametric problems) are substitution of SVD for randomised SVD whenever it is used and early stopping of fine-grid solution by checking the newest SV added with the Frequent Directions sketching algorithm. It must be noted, however, that the latter improvement's time-saving effect was purely confined to this method and the next, since it is generally a good idea to give NNs more data, and upon testing on simpler problems we have found that this time-saving effect can be very substantial, reducing e.g. the necessary amount of Darcian flow problem evaluations on fine grid to merely two. Below we outline the employed RSVD and FREDE algorithms (Ji et al., 2016)(Ghashami et al., 2015).

3.2. SINDy

Github repo: https://github.com/Genndoso/Reconstruction-of-fine-grid-CFD-solution-from-a-coarse-grid-one

The name of the subsection is a bit misleading: in fact this is not the SINDy method proposed by Brunton and Kutz, but rather a modification based on the same general principles (S.B). We firstly motivate our choice to approach the problem before us from a functional-basis-regression standpoint by listing considerations that support such an arrangement. Firstly we consider the coarse-grid deviation from fine-grid as a result of missing interior points. The question of enhancement is therefore one of interpolation, which is manifestly reducible to a special case of the method under consideration. Secondly, we consider the error arising from the eminent smoothing properties of coarsening: suppose we have a problem of fluid movement past a cylindre. Suppose that we choose cells so large the cylindre lies wholly inside one of them. Of course, the flow would reduce to simple flow of Poiseuille which is radically different from the desired model. Rectification of imperfections of this kind, namely loss of resolution upon coarsening, we may pose as a problem of solving some differential equation inside the large cells. This would be easily understandable as precisely a functional-basis regression, perhaps onto some orthogonal function basis.

Now armed with this motivation, we outline the algorithm.

SINDy can be understood as abusing the Galerkin projection by piling on an indeterminate number of very distinct

functions and projecting onto them, with the difference that SINDy necessarily demands parsimony of representation.

We approximate the dynamic system under study (which we here understand as evolution in the space of discretisation steps, or, alternatively, as application of an indeterminate best-fit interpolation operator) in the way:

$$\mathbf{X}_h = \mathbf{\Theta}(\mathbf{X_H})\mathbf{\Xi}$$

Where $\mathbf{\Xi}$ contains coefficients, and $\mathbf{\Theta}(\mathbf{X_H})$ is a library of functions, nominally symbolic, but perfectly working with grid-evaluated.

We can therefore compose each column of the coefficient matrix by seeking a sparsest possible solution in the following way:

$$\xi_k = \arg\min_{\xi_k^{'}} \|\mathbf{X_h}_k - \mathbf{\Theta}(\mathbf{X_H})\xi_k^{'}\|_2 + \\ +\lambda\|\xi_k^{'}\|_1 \tag{22}$$

This may be solved by a number of methods. LASSO, though common in machine learning for the purpose, was not chosen by us because it does not enforce sparsity strictly enough. Instead, two algorithms were tested: Sequential Thresholded Least Squares (STLS) and Sparse Relaxed Regularised Regression (SR3) (Champion et al., 2020). Refer to the appendix. We must note that we eventually discarded SR3 for reason of its reliance on nonstandard libraries for implementation of the proximal operator (sigpy, ODL, etc.); STLS was found to perform well, and SR3 in many cases gives identical results with the exception of things that are useful for SINDy's purpose by design but which we are not terribly likely to encounter in our application, like trimming corrupted data.

3.3. Autoencoder

Github repo: https://github.com/Genndoso/Reconstruction-of-fine-grid-CFD-solution-from-a-coarse-grid-one
Autoencoderis a special architecture of artificial neural networks that provides unsupervised learning using the backpropagation method. (Liou et al., 2014) The simplest autoencoder architecture is a feed-forward network, without feedback, most similar to a perceptron and containing an input layer, an intermediate layer, and an output layer.(Goodfellow et al., 2016) Unlike a perceptron, the output layer of an autoencoder must contain as many neurons as the input layer. The main principle of the standard autoencoder network is to get a response at the output layer that is closest to the input one. On Figure 3 example of the typical autoencoder is presented.

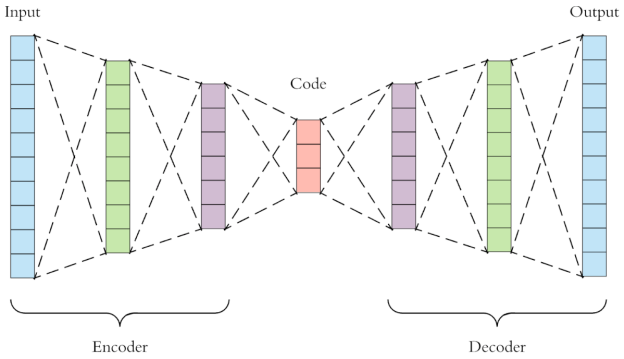

Figure 3. Example of autoencoder architecture

In our case we feed results obtained on the coarse mesh as an input to the autoencoder and results of on the fine mesh as the output. This way we try to build operator which maps results from the coarse mesh to the fine one.

Structure of our encoder is the following: as an input and output we provide RGB pictures, which are represented as width*height*3 neurons. The whole net is a simple MLP (width*height*3 – 1536 – 153 – 70 – 153 – 1536 –width*height*3) structure with LeakyReLU activations and Dropouts on each layer. 100 epochs was used for training. Adam optimizer and learning rate scheduler was implemented. Training was based on MSE loss:

$$MSEloss = \sqrt{\sum_{i=1}^{n} y_{true} - y_{pred}} \tag{23}$$

3.4. Deep convolutional generative adversarial network (DCGAN)

Github repo: https://github.com/Genndoso/Reconstruction-of-fine-grid-CFD-solution-from-a-coarse-grid-one
Generative adversarial networks is one of the most popular models of machine learning for the last several years. Generative networks existed before GANs but the "adversarial" part actually adds a lot of value and new perspectives. The novelty comes from the fact that in this framework we train two models at the same time: the Generative and the discriminative ones. The Generative model tries to capture the data distribution, while the Discriminator model one estimates a probability that some content was generated by the Generative model, and did not come from the actual data. The Generative model then tries to fool the Discriminator model, by maximising the probability that it makes a mistake. This competition between these two models is what makes GANs so powerful (Ian J. Goodfellow, 2014) Deep convolutional GAN (DCGAN) was for the given task of fine grid reconstruction. DCGAN can generate realistic synthesis images from vectors in the latent space. (P. Isola, 2017) proposed the network learning the mapping from an

input image to an output image to enable the translation between two images.

To construct GAN from to construct fine grid from low resolution coarse grid we tried to implement U-net architecture for generator part and feed-forward convolution network (CNN) for discriminator part. Example of U-net architecture is presented on Figure 4.

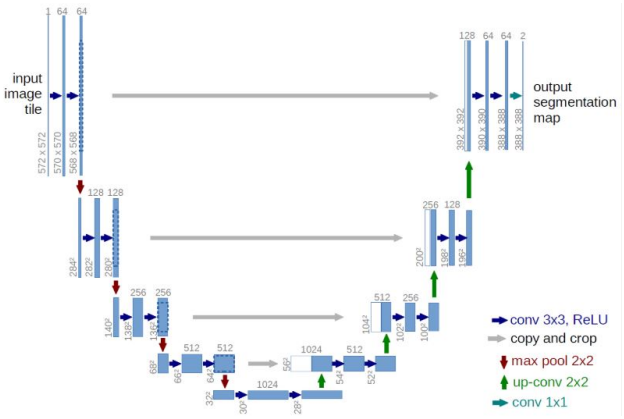

Figure 4. Example of U-net architecture

Discriminator part of Generative adversarial network consist in 4 convolution layers with batch normalization and LeakyReLU as activation function for all layers except the last. Detailed architecture of Discriminator is presented on Table 1.

Table 1. Architecture of the DCGAN discriminator

LAYER	OUT FEATURES
CONVOLUTION	64
CONVOLUTION	128
CONVOLUTION	256
CONVOLUTION	512
CONVOLUTION	1

Architecture of generator is presented on Table 2. The presented architecture was chosen by the several iterations of different architecture in order to find that one which are able fully work with image translation task. During optimal architecture searching main attention was paid on the amount of convolution layers and amount of filters on each layer of generator. Down layers has LeakyReLU activation function, while up layers after bottleneck have ReLU activation function. Moreover, all hidden convolution layers have batch normalization to preserve problem called internal covariate shift. Final convolution has Tanh activation.

Table 2. Architecture of the DCGAN generator

LAYER	OUT FEATURES
CONVOLUTION	64
CONVOLUTION	128
CONVOLUTION	256
CONVOLUTION	512
CONVOLUTION	512
CONVOLUTION	512
CONVOLUTION	512
CONVOLUTION BOTTLENECK	512
CONVOLUTION TRANSPOSE	1024
CONVOLUTION TRANSPOSE	512
CONVOLUTION TRANSPOSE	512
CONVOLUTION TRANSPOSE	512
CONVOLUTION TRANSPOSE	256
CONVOLUTION TRANSPOSE	128
CONVOLUTION TRANSPOSE	64
CONVOLUTION TRANSPOSE	3

Both datasets was trained on similar architectures. For training it was used batch size equal to 12 because of lack of computational power. Google Colab, unfortunately, doesn't provide significant computation power on GPU's. Input images and target images were preliminary normalized. Initial weight initialization was applied by model itself. For discriminator training L1 loss was used:

$$L1loss = \sum_{i=1}^{n} |y_{true} - y_{pred}| \tag{24}$$

For generator training binary cross entropy loss was implemented:

$$CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{\mathrm{N}_c} y_i \log(\hat{y}_i) \tag{25}$$

To find common patterns and obtain not so accurate but acceptable results there were used 100 epochs. For generator and discriminator Adam optimizer was chosen.

4. Experiments and Results

The obtained results are presented on Table 3. The quality of utilized algorithms was based on:

$$\|\tilde{\mathbf{X}}_H\| = \frac{\|\tilde{\mathbf{X}}_H - \mathbf{X}_h\|_1}{\|\mathbf{X}_h\|_1} \tag{26}$$

Table 3. Obtained results

ALGORITHM	CYLINDRIC DATASET	ORSZAG-TANG DATASET	KELVIN-HELMHOLTZ DATASET
SINDY	0.0001	0.001	7E-5
NIRB	0.178	0.262	0.005
NIRB-R	0.0005	0.007	0.0001
DCGAN	0.0442	0.0734	0.0115
AUTOENCODER	0.047	0.073	-

5. Conclusion

We conclude with the following: the unrectified NIRB method shows itself to be unstalbe and performs very poorly on large bases as the condition number deteriorates, the rectified NIRB achieves stellar performance, being able to approximate fine-grid solutions with good precision, the SINDy method performs about the same with SR3 and STLS and outperforms all other methods by far. These three methods, however, have serious drawbacks. Firstly, expensive computation is required to use them on high-resolution datasets, and the time can get very burdensome (it has been measured that the time it took for NIRB to evaluate an approximation to the Karman vortex street on a basis of 75 vectors was well in excess of two hours). Additionally, they present steep demands regarding the RAM of the machine and thus cannot be run on Colab for even somewhat serious data; a powerful remote machine was required, and the time was still quite long. On the other hand, neural network-based approaches do not demand consistent performance of a large quantity of explicit linear algebraic operations on sometimes very extensive snapshot matrices and provide a workable alternative, although their relative precision was not found to be impressive. Additionally, once a net has attained the final weights, it may be used to inexpensively predict however many new solutions, whereas the other methods will be obliged to perform regular costly steps for the new targets.

References

Batchelor, G. K. *An introduction to fluid dynamics*. Cambridge University Press, 1967.

Chakir, R. and Maday, Y. A two-grid finite-element/reduced basis scheme for the approximation of the solution of parameter dependent PDE. In *9e Colloque national en calcul des structures*, Giens, France, May 2009. CSMA. URL https://hal.archives-ouvertes.fr/hal-01420726.

CHAKIR, R., Joly, P., Maday, Y., and Parnaudeau, P. A Non intrusive reduced basis method : application to computational fluid dynamics. In *2nd ECCOMAS Young Investigators Conference (YIC 2013)*, Bordeaux, France, September 2013. URL https://hal.archives-ouvertes.fr/hal-00855906.

Chakir, R., Maday, Y., and Parnaudeau, P. A non-intrusive reduced basis approach for parametrized heat transfer problems. *Journal of Computational Physics*, 376, 10 2018. doi: 10.1016/j.jcp.2018.10.001.

Champion, K., Zheng, P., Aravkin, A., Brunton, S., and Kutz, J. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, 8:169259–169271, 01 2020. doi: 10.1109/ACCESS.2020.3023625.

Frank, A., Jones, T. W., Ryu, D., and Gaalaas, J. B. The Magnetohydrodynamic Kelvin-Helmholtz Instability: A Two-dimensional Numerical Study. , 460:777, April 1996. doi: 10.1086/177009.

Ghashami, M., Liberty, E., Phillips, J. M., and Woodruff, D. P. Frequent directions : Simple and deterministic matrix sketching. *CoRR*, abs/1501.01711, 2015. URL http://arxiv.org/abs/1501.01711.

Goda, K. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics*, 30(1):76–95, 1979. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(79)90088-3. URL https://www.sciencedirect.com/science/article/pii/0021999179900883.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Ian J. Goodfellow, j Pouget-Abadie, M. M. B. X. Generative adversarial networks. *Cornel university*, 2014.

Ji, H., Yu, W., and Li, Y. A rank revealing randomized singular value decomposition (R3SVD) algorithm for low-rank matrix approximations. *CoRR*, abs/1605.08134, 2016. URL http://arxiv.org/abs/1605.08134.

Kigure, H. Application of video-to-video translation networks to computational fluid dynamics. *Frontiers in Artificial intelligence*, 2021.

Liou, C.-Y., Cheng, W.-C., Liou, J.-W., and Liou, D.-R. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.

Orszag, S. A. and Tang, C. M. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, 90:129–143, January 1979. doi: 10.1017/S002211207900210X.

P. Isola, J. Zhu, T. Z. Image-to-image translation with conditional adversarial networks. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

P. Pant, R. Doshi, P. B. A. B. Deep learning for reduced order modeling and efficient temporal evolution of fluid simulations. *Archiv preprint: https://arxiv.org/abs/2107.04556*, 2021.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Semyon Polyansky (30% of work)

- Computation of the data
- Implementation of NIRB and SINDy algorithms, R3SVD, FREDE, STLS, SR3
- Numerical experiments with aforementioned models
- Preparing relevant sections of this report
- Preparing the SINDy, NIRB sections of this report

Maksimilian Pavlov (25% of work)

- Preparing the GitHub Repo
- Preparing the Section about GAN
- Preparing the Related Work section
- Coding and testing GAN on two datasets

Dmitry Belousov (15% of work)

- Testing different hyperparameters
- Preparing Presentation
- Preparing Video
- Preparing Appendix section
- Datasets preparation

Olga Gorbunova (10% of work)

- Preparing Presentation
- Data preparation
- Testing different hyperparameters

Egor Rumiantsev (20% of work)

- Autoencoder testing for different datasets
- Preparing the section about autoencoder

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: Basic template for GAN was taken, but weights initialization, training and architecture was changed. Other code is self-made completely

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   - [ ] Yes.
   - [x] No.
   - [ ] Not applicable.

   Students' comment: The utilized datasets are very heavy. The cylinder datasets only included in github repo and can be easily downloaded from the link

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: It is included only for ML models

10. A description of how experiments have been conducted is included.

    - [ ] Yes.
    - [x] No.
    - [ ] Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    - [ ] Yes.
    - [x] No.
    - [ ] Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

C. Algorithms

Algorithm 1 Non-Intrusive Reduced Basis Algorithm With Rectified Projective Coefficients

Input: parameterised snapshot matrix $\mathbf{X_h}$, size $L \times N$; parameterised interpolated coarse-grid snapshot matrix $\tilde{\mathbf{X}}_\mathbf{H} = \mathbf{I}_\mathbf{H}^\mathbf{h}\mathbf{X_H}$, size $L \times N$; regularisation parameter λ
Obtain orthonormal basis Φ by means of Gram-Schmidt or SVD or PCA: $\phi_i \in \mathbf{Span}(X_h); i = 1, \cdots, M; M \leq N; < \phi_i, \phi_j >= \delta_{ij}$
for $i = 1$ **to** N **do**
  for $j = 1$ **to** M **do**
    $\alpha_i^j =< u_h^i, \phi_j >$
    $\beta_i^j =< I_H^h u_H^i, \phi_j >$
  end for
end for
for $i = 1$ **to** N **do**
  $\psi_i = \arg\min_{\psi_i}(\|\boldsymbol{\beta}\psi_i - \alpha_i\| + \lambda\|\psi_i\|)$
  for $j = 1$ **to** M **do**
    $\tilde{\beta}_i^j = \sum_j \psi_{i,j} < I_H^h u_H^i, \phi_j >$
  end for
  for $j = 1$ **to** M **do**
    for $K = 1$ **to** M **do**
      $\hat{u}_h^i = \psi_{j,k} < I_H^h u_H^i, \phi_k > \phi_j$
    end for
  end for
end for

Algorithm 2 FREquent Directions skEtching

Input: l, $\mathbf{A} \in \mathbb{R}^{n \times d}$
$\mathbf{B} = 0^{l \times d}$
for $i = 1$ **to** n **do**
  $\mathbf{B}_l = a_i$
  $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = svd(B)$
  $\delta = \sigma_l^2$
  $\mathbf{B} = \sqrt{\mathbf{\Sigma}^2 - \delta \mathbf{I}_l} \cdot \mathbf{V}^T$
end for

Algorithm 3 Sparse Relaxed Regularised Regression

Input: $\epsilon, \mathbf{W}_0$
$k = 0; err = 2\epsilon$
while $err > \epsilon$ **do**
  $k = k + 1$
  $\Xi^k = \arg\min_\Xi \frac{1}{2}\|\mathbf{X}_h - \mathbf{\Theta\Xi}\|^2 + \frac{1}{2v}\|\mathbf{\Xi} - \mathbf{W}^{k-1}\|^2$
  $\mathbf{W}^k = prox_{\lambda v R}\mathbf{\Xi}^k$
  $err = \frac{\|\mathbf{W}^k - \mathbf{W}^{k-1}\|}{v}$
end while
Note: $prox$ stands for the proximal operator.

Algorithm 4 Rank-Revealing Randomised Singular Value Decomposition With Power Scheme For Sampled Space Refinement

Input: l, $\mathbf{A} \in \mathbb{R}^{m \times n}$; sampling size t; oversampling number p; power number q; maximum iteration amount $maxit$; early-stopping nergy threshold τ
$\mathbf{\Omega} = Gaussian; \mathbf{\Omega} \in \mathbb{R}^{n \times (t+p)}$
$\mathbf{G}_0 = \mathbf{\Omega}$
$\mathbf{V}_L = \emptyset$
$\mathbf{U}_L = \emptyset$
$\mathbf{\Sigma}_L = \emptyset$
$k' = 0$
for $i = 0$ **to** $maxit$ **do**
  $\mathbf{Y}_i = \mathbf{AG}_i$
  $\mathbf{Q}_i = qr(\mathbf{Y}_i, 0)$
  for $j = 1$ **to** q **do**
    $\mathbf{Y}_i = \mathbf{A}^T\mathbf{Q}_i$
    $\mathbf{Y}_i = \mathbf{Y}_i - \mathbf{V}_L(\mathbf{V}_L^T\mathbf{Y}_i)$
    $\mathbf{Q}_i = qr(\mathbf{Y}_i, 0)$
    $\mathbf{Y}_i = \mathbf{AQ}_i$
    $\mathbf{Y}_i = \mathbf{Y}_i - \mathbf{V}_L(\mathbf{V}_L^T\mathbf{Y}_i)$
    $\mathbf{Q}_i = qr(\mathbf{Y}_i, 0)$
  end for
  $\mathbf{B}_i = (\mathbf{A}^T\mathbf{Q}_i)^T$
  $[\mathbf{U}_{\mathbf{B}_i}; \mathbf{\Sigma}_{\mathbf{B}_i}; \mathbf{V}_{\mathbf{B}_i}] = svd(\mathbf{B}_i, 0)$
  $\mathbf{U}_{\mathbf{B}_i} = \mathbf{Q}_i\mathbf{U}_{\mathbf{B}_i}$
  $\mathbf{V}_{\mathbf{B}_i} = qr(\mathbf{V}_{\mathbf{B}_i} - \mathbf{V}_L(\mathbf{V}_L^T\mathbf{V}_{\mathbf{B}_i}), 0)$
  $\mathbf{U}_L = [\mathbf{U}_L; \mathbf{U}_{\mathbf{B}_i}(:, 1:t)]$
  $\mathbf{\Sigma}_L = \begin{bmatrix} \mathbf{\Sigma}_L & 0 \\ 0 & \mathbf{U}_{\mathbf{\Sigma}_i}(1:t,1:t) \end{bmatrix}$
  $\mathbf{V}_L = [\mathbf{V}_L; \mathbf{V}_{\mathbf{B}_i}(:, 1:t)]$
  for $j = 1$ **to** t **do**
    $k' = i \times t + j$
    $\tilde{\phi}_{k'} = \frac{\sum_{i=1}^{k'} \sigma'^2_i}{\|\mathbf{A}\|_F^2}$
    if $\tilde{\phi}_{k'} \geq \tau$ **then**
      stop
    end if
  end for
  $\mathbf{G}_{i+1} = \mathbf{G}_i - \mathbf{V}_i(\mathbf{V}_i^T\mathbf{G}_i)$
end for
$[\mathbf{\Sigma}_L, Idx]$ = sort($\mathbf{\Sigma}_L$, 'descend')
$\mathbf{V}_L = \mathbf{V}_L(:, Idx)$
$\mathbf{U}_L = \mathbf{U}_L(:, Idx)$

Algorithm 5 Sequential Thresholded Least Squares

Input: $\mathbf{\Theta}, \mathbf{X}_h, \lambda, n, m$
$\Xi = \mathbf{\Theta}^{-1}\mathbf{X}_h$
for $i = 1$ **to** m **do**
  $\Xi_{jk} = 0; |\Xi_{jk}| < \lambda$
  for $j = 1$ **to** n **do**
    $\Xi_{kj} = \mathbf{\Theta}_k^{-1}\mathbf{X}_{hj}; |\Xi_{kj}| > 0$
  end for
end for