Imperial College London

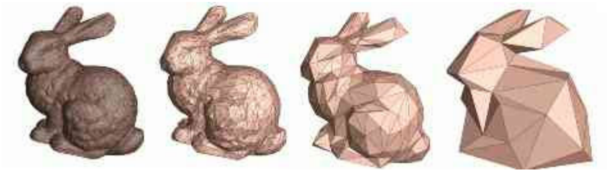
AMCG

Reduced-Order Modelling

Reduced-Order Modelling

Reduced-Order Modelling (ROM) or Model Order Reduction:

approximating a high-dimensional system with a lower-dimensional system whilst maintaining reasonable accuracy and gaining a significant computational speed up.



- ROM has been around for a long time
- it is one type of surrogate modelling
- "discretise-then-reduce" rather than "reduce-then-discretise" Sartori, . . . , Rozza 2016
- projection-based ROM (reduced basis) vs non-intrusive ROM
- data-driven
- digital twin (Girolami, Willcox, Schilders, Omer, . . .)
- parametric systems, time-dependent systems, parametric time-dependent systems

Reduced-Order Modelling

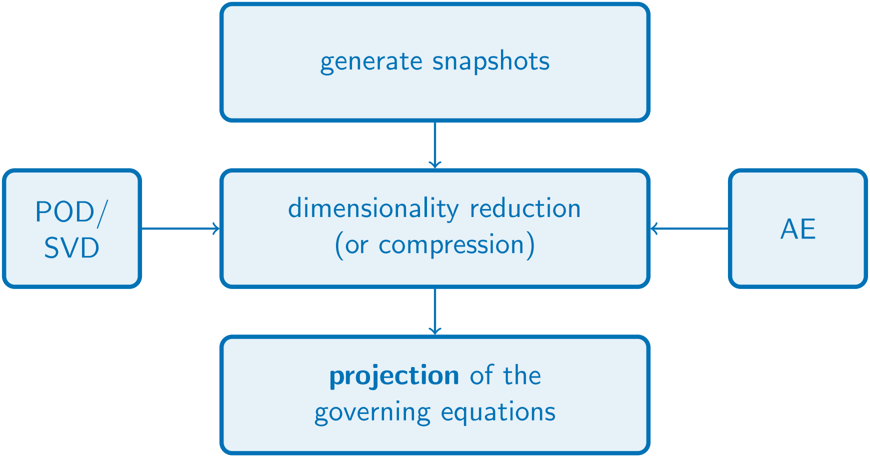
Efficiency is gained by an offline / online scheme.

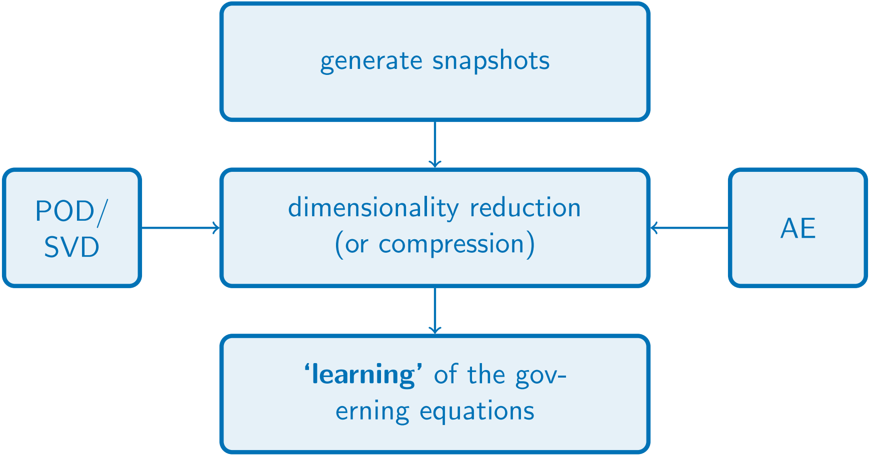
- Offline stage:
  - computationally expensive
  - run a series of forward models which are representative of the behaviour of the system
  - obtain basis functions (often POD / SVD based)
  - obtain building blocks which will be used to predict the evoultion of the system or to describe the parameter-dependence of the problem
- Online stage:
  - rapid to run
  - predict in time (train in time)
  - predict for unseen parameters (train for a set of parameter values)

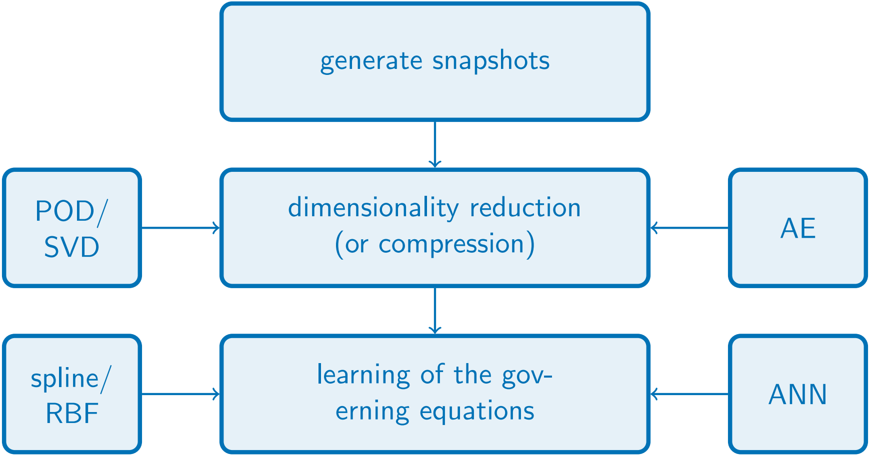
- Offline stage:
  (a) generate snapshots (solutions at different time levels / or solutions corresponding to different model parameter values)
  (b) find reduced basis, often done using Proper Orthogonal Decomposition which is an SVD-based method
  (c) project the original discretised system onto the low-dimensional space spanned by the basis functions found in (b)
- Online stage:
  (a) choose an arbitrary parameter value or time range for the prediction
  (b) assemble the reduced system corresponding to that parameter value
  (c) solve the reduced system
  (d) map back from reduced space to the physical space

- Offline stage:
  (a) generate snapshots (solutions at different time levels / or solutions corresponding to different model parameter values)
  (b) find reduced basis, often done using Proper Orthogonal Decomposition which is an SVD-based method
  (c) produce the reduced data that will be used in the training process
  (d) train a neural network
- Online stage:
  (a) choose an arbitrary parameter value or time range for the prediction
  (b) use the neural network to make predictions
  (c) map back from reduced space to the physical space

Reduced-Order Modelling
Projection-based: Offline stage




generate snapshots
POD/ SVD
dimensionality reduction (or compression)
AE
'learning' of the gov- erning equations



generate snapshots
POD/ SVD
dimensionality reduction (or compression)
AE
spline/ RBF
learning of the gov- erning equations
ANN

High-fidelity / high-resolution model (after discretisation)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \text{ where } \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{u}, \boldsymbol{b} \in \mathbb{R}^{N}.$$

Calculate snapshots matrix

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1), \boldsymbol{u}(\mu_2), \cdots, \boldsymbol{u}(\mu_{N^s})] \text{ or } \left[\boldsymbol{u}^{t_1}, \boldsymbol{u}^{t_2}, \cdots, \boldsymbol{u}^{t_{N^s}}\right], \ \boldsymbol{S} \in \mathbb{R}^{N\times N^s}.$$

Obtain POD basis functions by applying Singular Value Decomposition to the snapshots matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$ and defining the POD basis $\boldsymbol{R}$ to be the first $M \leqslant N^s \ll N$ columns of $\boldsymbol{U}$, i.e. $\boldsymbol{R} \in \mathbb{R}^{N\times M}$.

Project discretised system onto the low-dimensional space

$$\tilde{\boldsymbol{A}}\boldsymbol{\alpha} \equiv \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{b} \equiv \tilde{\boldsymbol{b}} \text{ where } \tilde{\boldsymbol{A}} \in \mathbb{R}^{M\times M}, \boldsymbol{\alpha}, \tilde{\boldsymbol{b}} \in \mathbb{R}^{M}.$$

High-fidelity / high-resolution model (after discretisation)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \text{ where } \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{u}, \boldsymbol{b} \in \mathbb{R}^{N}.$$

Calculate snapshots matrix

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1), \boldsymbol{u}(\mu_2), \cdots, \boldsymbol{u}(\mu_{N^s})] \text{ or } \left[\boldsymbol{u}^{t_1}, \boldsymbol{u}^{t_2}, \cdots, \boldsymbol{u}^{t_{N^s}}\right], \ \boldsymbol{S} \in \mathbb{R}^{N\times N^s}.$$

Obtain POD basis functions by applying Singular Value Decomposition to the snapshots matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and defining the POD basis $\boldsymbol{R}$ to be the first $M \leqslant N^s \ll N$ columns of $\boldsymbol{U}$, i.e. $\boldsymbol{R} \in \mathbb{R}^{N\times M}$.

Project discretised system onto the low-dimensional space

$$\tilde{\boldsymbol{A}}\boldsymbol{\alpha} \equiv \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{b} \equiv \tilde{\boldsymbol{b}} \text{ where } \tilde{\boldsymbol{A}} \in \mathbb{R}^{M\times M}, \boldsymbol{\alpha}, \tilde{\boldsymbol{b}} \in \mathbb{R}^{M}.$$

High-fidelity / high-resolution model (after discretisation)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \text{ where } \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{u}, \boldsymbol{b} \in \mathbb{R}^{N}.$$

Calculate snapshots matrix

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1), \boldsymbol{u}(\mu_2), \cdots, \boldsymbol{u}(\mu_{N^s})] \text{ or } \left[\boldsymbol{u}^{t_1}, \boldsymbol{u}^{t_2}, \cdots, \boldsymbol{u}^{t_{N^s}}\right], \ \boldsymbol{S} \in \mathbb{R}^{N\times N^s}.$$

Obtain POD basis functions by applying Singular Value Decomposition to the snapshots matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$ and defining the POD basis $\boldsymbol{R}$ to be the first $M \leqslant N^s \ll N$ columns of $\boldsymbol{U}$, i.e. $\boldsymbol{R} \in \mathbb{R}^{N\times M}$.

Project discretised system onto the low-dimensional space

$$\tilde{\boldsymbol{A}}\boldsymbol{\alpha} \equiv \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{b} \equiv \tilde{\boldsymbol{b}} \text{ where } \tilde{\boldsymbol{A}} \in \mathbb{R}^{M\times M}, \boldsymbol{\alpha}, \tilde{\boldsymbol{b}} \in \mathbb{R}^{M}.$$

High-fidelity / high-resolution model (after discretisation)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \text{ where } \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{u}, \boldsymbol{b} \in \mathbb{R}^{N}.$$

Calculate snapshots matrix

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1), \boldsymbol{u}(\mu_2), \cdots, \boldsymbol{u}(\mu_{N^s})] \text{ or } \left[\boldsymbol{u}^{t_1}, \boldsymbol{u}^{t_2}, \cdots, \boldsymbol{u}^{t_{N^s}}\right], \ \boldsymbol{S} \in \mathbb{R}^{N\times N^s}.$$

Obtain POD basis functions by applying Singular Value Decomposition to the snapshots matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$ and defining the POD basis $\boldsymbol{R}$ to be the first $M \leqslant N^s \ll N$ columns of $\boldsymbol{U}$, i.e. $\boldsymbol{R} \in \mathbb{R}^{N\times M}$.

Project discretised system onto the low-dimensional space

$$\tilde{\boldsymbol{A}}\boldsymbol{\alpha} \equiv \boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{b} \equiv \tilde{\boldsymbol{b}} \text{ where } \tilde{\boldsymbol{A}} \in \mathbb{R}^{M\times M}, \boldsymbol{\alpha}, \tilde{\boldsymbol{b}} \in \mathbb{R}^{M}.$$

High-fidelity / high-resolution model (after discretisation)

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \text{ where } \boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{u}, \boldsymbol{b} \in \mathbb{R}^{N}.$$

Calculate snapshots matrix

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1), \boldsymbol{u}(\mu_2), \cdots, \boldsymbol{u}(\mu_{N^s})] \text{ or } \left[\boldsymbol{u}^{t_1}, \boldsymbol{u}^{t_2}, \cdots, \boldsymbol{u}^{t_{N^s}}\right], \ \boldsymbol{S} \in \mathbb{R}^{N\times N^s}.$$

Obtain POD basis functions by applying Singular Value Decomposition to the snapshots matrix $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$ and defining the POD basis $\boldsymbol{R}$ to be the first $M \leqslant N^s \ll N$ columns of $\boldsymbol{U}$, i.e. $\boldsymbol{R} \in \mathbb{R}^{N\times M}$.

Learn evolution of the POD coefficients of the snapshots in time (or learn the parameter dependence of the POD coefficients of the snapshots) – for example, train a feed-forward NN with input-output pairs $\{(\boldsymbol{\alpha}^n, \boldsymbol{\alpha}^{n+1})\}_{k=1}^{N^s-1}$

Reduced-Order Modelling

Advantages:

Projection-based ROM is more closely related to the governing equations (and therefore physics) than NIROM.

To mitigate this, researchers are combining NIROM with

- physics-informed neural networks,
- data assimilation.

Projection-based ROM is more likely to result in a stable model than NIROM. To mitigate this, researchers are using adversarial techniques.

NIROM does not require the source code to be modified

- legacy code,
- complex codes,
- licensed software.

SVD: decomposes a matrix into three matrices, two orthogonal ones ($\boldsymbol{U}$, $\boldsymbol{V}$) and one with non-zeros only on the diagonal ($\boldsymbol{\Sigma}$).

POD: expands a solution in terms of coefficients and orthogonal basis functions. The basis functions are such that the representation error is minimised.

POD expansion for a temperature field $\boldsymbol{u}$:

$$\boldsymbol{u} = \sum_{i=1}^{M} \alpha_i \boldsymbol{\phi}_i \tag{1}$$

$$\text{or} \quad \boldsymbol{u} = \boldsymbol{R}\boldsymbol{\alpha} \tag{2}$$

where $\boldsymbol{\phi}_i$ is the ith column of $\boldsymbol{R}$, in other words, the ith POD basis function, and α_i is the ith entry of $\boldsymbol{\alpha}$, that is the ith POD coefficient.

For a particular projection Π, the representation error of the snapshots can be written as

$$\varepsilon = ||\boldsymbol{u} - \Pi(\boldsymbol{u})||^2 \equiv ||\boldsymbol{u} - \boldsymbol{R}\boldsymbol{R}^T\boldsymbol{u}||^2$$

where $\boldsymbol{R}^T$ maps from physical space to the reduced space and $\boldsymbol{R}^T$ maps from the reduced space to the physical space. We want to minimise this, and we know from theory that if the columns of $\boldsymbol{R}$ are the left singular vectors, the error will be minimised.

NB. POD is also known as principal component analysis, empirical orthogonal functions, the Hotelling transform, KarhunenLoève theorem.

Usually it is recommended to subtract the mean from the snapshots matrix:

POD expansion for a temperature field $\boldsymbol{u}$:

$$\boldsymbol{u} = \bar{\boldsymbol{u}} + \sum_{i=1}^{M} \tilde{\alpha}_i \tilde{\boldsymbol{\phi}}_i \tag{3}$$

$$\text{or} \quad \boldsymbol{u} = \bar{\boldsymbol{u}} + \tilde{\boldsymbol{R}}\tilde{\boldsymbol{\alpha}} \tag{4}$$

where the mean is given by $\bar{\boldsymbol{u}}$. The snapshots matrix will take the form

$$\boldsymbol{S} = [\boldsymbol{u}(\mu_1) - \bar{\boldsymbol{u}}, \boldsymbol{u}(\mu_2) - \bar{\boldsymbol{u}}, \cdots, \boldsymbol{u}(\mu_{N^s}) - \bar{\boldsymbol{u}}]$$

$$\text{or} \ \left[\boldsymbol{u}^{t_1} - \bar{\boldsymbol{u}}, \boldsymbol{u}^{t_2} - \bar{\boldsymbol{u}}, \cdots, \boldsymbol{u}^{t_{N^s}} - \bar{\boldsymbol{u}}\right].$$

It is often more computationally efficient to solve an eigenvalue problem than to apply singular value decomposition to a matrix.
Consider $\boldsymbol{S}^T\boldsymbol{S} \in \mathbb{R}^{M\times M}$,

$$\begin{aligned}
\boldsymbol{S}^T\boldsymbol{S} &= \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\right)^T\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\right) && (5)\\
&= \left(\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\right)\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\right) && (6)\\
&= \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T && (7)\\
&= \boldsymbol{V}\left(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\right)\boldsymbol{V}^T && (8)
\end{aligned}$$

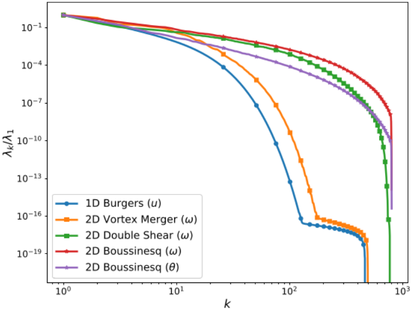
which is the eigendecomposition of the matrix $\boldsymbol{S}^T\boldsymbol{S}$ with eigenvectors $\boldsymbol{V}$ and eigenvalues on the diagonal of $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ (as $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal they both satisfy $\boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{I}$, where $\boldsymbol{M}$ is an orthogonal matrix and $\boldsymbol{I}$ is the identity matrix).

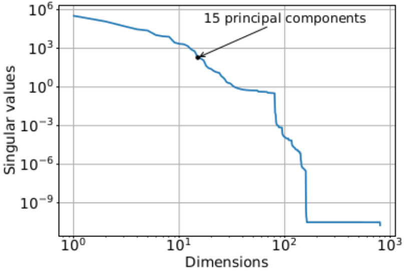
For SVD/POD-based methods, the $\boldsymbol{R}$ matrix maps from the high-dimensional space to the low-dimensional one:

$$\boldsymbol{u} = \boldsymbol{R}\boldsymbol{\alpha} \text{ and } \boldsymbol{\alpha} = \boldsymbol{R}^T\boldsymbol{u}, \text{ where } \boldsymbol{u} \in \mathbb{R}^N, \boldsymbol{\alpha} \in \mathbb{R}^M \text{ and } \boldsymbol{R} \in \mathbb{R}^{N\times M} \tag{9}$$

For autoencoder-based methods, the mappings come from the (trained) encoder (f^{enc}) and decoder (f^{dec}):

$$\boldsymbol{u} = f^{\text{dec}}(\boldsymbol{\alpha}) \text{ and } \boldsymbol{\alpha} = f^{\text{enc}}(\boldsymbol{u}), \text{ where } \boldsymbol{u} \text{ and } \boldsymbol{\alpha} \text{ as above.} \tag{10}$$

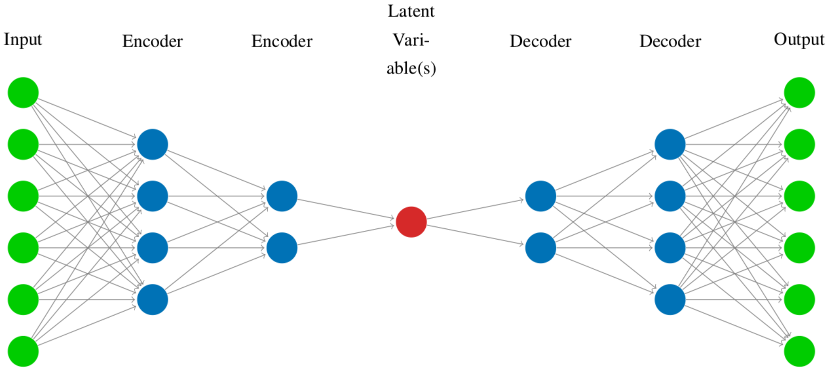




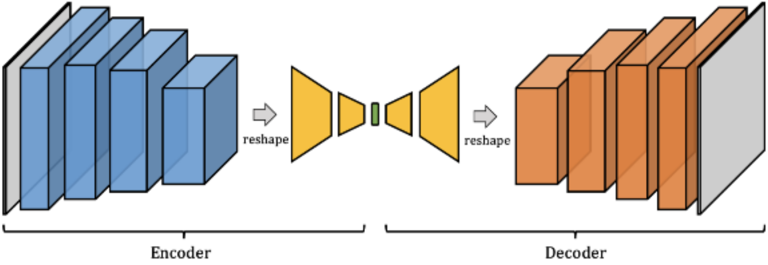
Left from Ahmed et al. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Physics of Fluids 31*, 126602 (2019).

Right from V. L. S. Silva, Data Assimilation Predictive GAN (DA-PredGAN): applied to determine the spread of COVID-19, *arXiv* (2021).

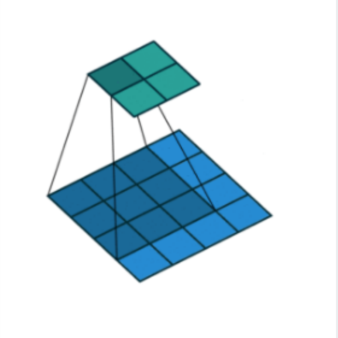
Reduced-Order Modelling
Autoencoders



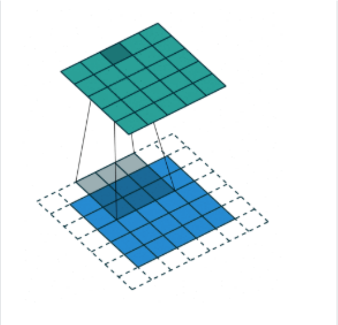
From Phillips et al. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion (2021).



Gonzalez et. al. *Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems*, arXiv, 2018.



No padding, no strides

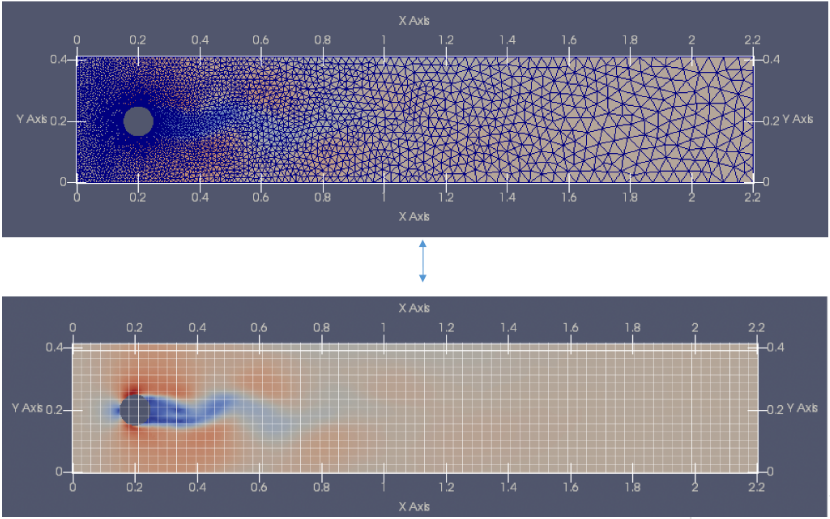


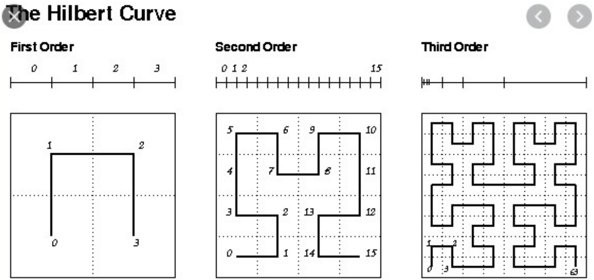
Half padding, no strides

from https://github.com/vdumoulin/conv_arithmetic

Reduced-Order Modelling

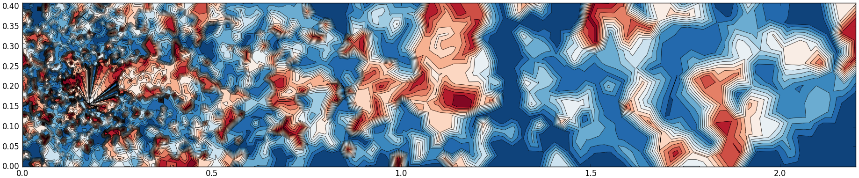
Mesh / grid

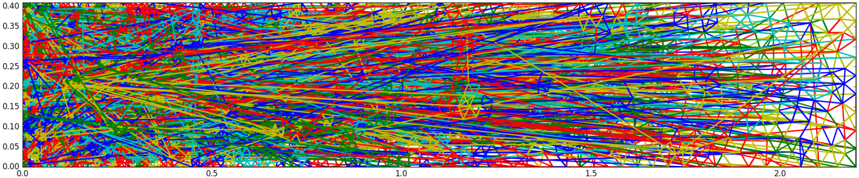




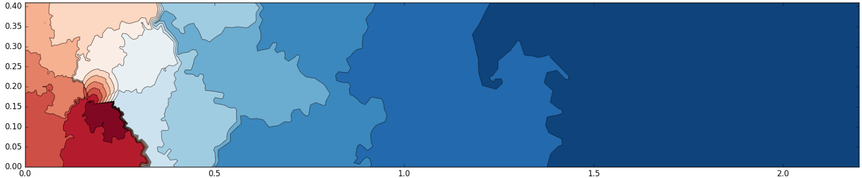
The Hilbert Curve
First Order
0
1
2
3
Second Order
0 1 2
15
Third Order
5
6
9
10
4
7
8
11
3
2
13
12
0
1
14
15
63

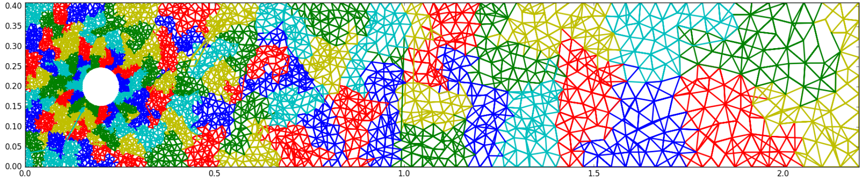
Without space-filling curve numbering





With space-filling curve numbering





2D flow past a cylinder, Re$=3900$, $20,550$ nodes

$[0, 2.2] \times [0, 0.41]$, cylinder centred at $(0.2, 0.2)$ radius 0.05

800 snapshots used for training, 100 for validation and 100 for testing.

FEM data

$\downarrow$

SFC ordering / Sparse layer

$\downarrow$

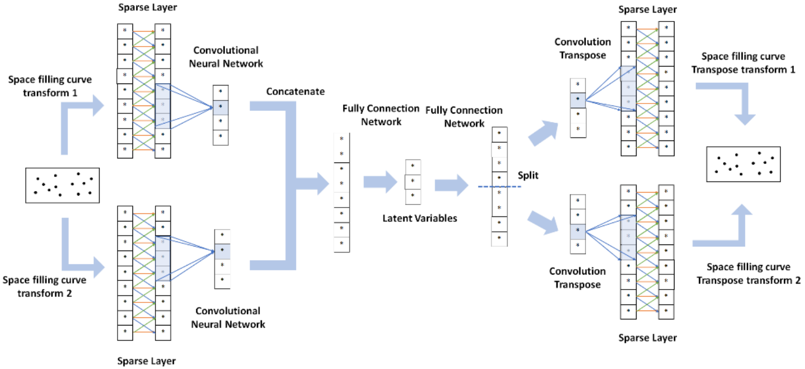
typical 1D convolutional autoencoder

$\downarrow$

inverse SFC ordering / Sparse layer

$\downarrow$

FEM data



C.E. Heaney, Y. Li, O.K. Matar and C.C. Pain. Applying Convolutional Neural Networks to Data on Unstructured Meshes with Space-Filling Curves (2020) `arxiv.org/abs/2011.14820`.

Train the GAN with the following data (assuming we only have 1 POD basis function):

$$\begin{pmatrix} \alpha^{t_3} \\ \alpha^{t_2} \\ \alpha^{t_1} \end{pmatrix}, \begin{pmatrix} \alpha^{t_4} \\ \alpha^{t_3} \\ \alpha^{t_2} \end{pmatrix}, \ldots, \begin{pmatrix} \alpha^{t_N} \\ \alpha^{t_{N-1}} \\ \alpha^{t_{N-2}} \end{pmatrix}$$

Once the generator $\mathcal{G}$ is trained, given a vector of random values $\boldsymbol{r}$, it produces the solution at three successive time levels:

$$\mathcal{G}(\boldsymbol{r}) = \begin{pmatrix} \alpha^{t_{n+2}} \\ \alpha^{t_{n+1}} \\ \alpha^{t_n} \end{pmatrix}$$

How can we produce a longer time series?

Say we have initial conditions of α^{t_k} and $\alpha^{t_{k+1}}$ from which we wish to march forward in time, we effectively want to minimise the difference between the known values and their associated output by the generator:

$$\min_{\boldsymbol{r}} \mathcal{M}\left(\begin{pmatrix} \alpha^{t_{n+2}} \\ \alpha^{t_{n+1}} \\ \alpha^{t_n} \end{pmatrix} - \begin{pmatrix} \\ \alpha^{t_{k+1}} \\ \alpha^{t_k} \end{pmatrix}\right)$$

where $\mathcal{M}$ is a function which squares each entry and takes the average, i.e.

$$\mathcal{M}(\boldsymbol{v}) = \frac{1}{N}\boldsymbol{v}\cdot\boldsymbol{v} \quad \text{where} \quad \boldsymbol{v}\in\mathbb{R}^N.$$

We can do this by backpropagating the error through the network and adjusting the values of $\boldsymbol{r}$, similar to how we backpropagate the error and adjusting the weights (parameters) when training neural networks.

Once we have

$$\begin{pmatrix} \alpha^{t_{n+2}} \\ \alpha^{t_{n+1}} \\ \alpha^{t_n} \end{pmatrix} - \begin{pmatrix} \alpha^{t_{k+1}} \\ \alpha^{t_k} \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

we take $\alpha^{t_{n+2}}$ as the prediction and set $\alpha^{t_{k+2}} = \alpha^{t_{n+2}}$.

To find the solution at the next time level we repeat the process, i.e.

$$\min_{\boldsymbol{r}} \mathcal{M}\left(\begin{pmatrix} \alpha^{t_{m+2}} \\ \alpha^{t_{m+1}} \\ \alpha^{t_m} \end{pmatrix}\right) - \left(\begin{pmatrix} \alpha^{t_{k+2}} \\ \alpha^{t_{k+1}} \end{pmatrix}\right)$$

to find $\alpha^{t_{k+3}}$, and so on.

Reduced-Order Modelling

Why autoencoders and not POD?

Why GANs and not other neural networks?

Some classic books:

F Chinesta, A Huerta, G Rozza, K Willcox. Model Reduction Methods. *Encyclopedia of Computational Mechanics Second Edition*, 1–36.

P Benner, M Ohlberger, A Cohen, K Willcox. Model Reduction and Approximation: Theory and Algorithms. *Society for Industrial and Applied Mathematics*.

Schilders, W. H., van der Vorst, H. A., Rommes, J. (Editors). Model Order Reduction: Theory, Research Aspects and Applications.

Many papers (Quarteroni, Rozza, Huerta, Schilders, Noack, Willcox, . . .).

New edition of classical methods for dimensionality reduction:

Holmes, P., Lumley, J., Berkooz, G. and Rowley, C. (2012). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry* (2nd ed., Cambridge Monographs on Mechanics). Cambridge.

POD and related methods:

Taira K, Brunton SL, Dawson STM, et al. Modal analysis of fluid flows: an overview. *AIAA J.* 2017; 55(12): 4013–4041.

(Old papers) comparing autoencoders to principal component analysis (PCA):

Oja E. A simplified neuron model as a principal component analyzer. J Math Biol. 1982; 15: 267–273.

Bourlard H, Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybern. 1988; 59: 291–294.

Baldi P, Hornik K. Neural networks and principal component analysis: learning from examples without local minima. Neural Netw. 1989; 2: 53–58.

Possibly the first to compare POD with AE within a NIROM framework:

Milano M, Koumoutsakos P. Neural network modeling for near wall turbulent flow. J Comput Phys. 2002; 182: 1–26.

How to apply CNNs to data on unstructured meshes:

CE Heaney, Y Li, OK Matar, CC Pain. Applying Convolutional Neural Networks to Data on Unstructured Meshes with Space-Filling Curves. *arXiv preprint arXiv:2011.14820*

R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman and D. Cohen-Or. MeshCNN: a network with an edge. (2019) *ACM Transactions on GraphicsJuly* 90.
Associated Github pages.

J. Tencer, K. Potter. A Tailored Convolutional Neural Network for Nonlinear Manifold Learning of Computational Physics Data using Unstructured Spatial Discretizations. *arXiv preprint 2006.06154*.

Projection-based ROM papers, one on ML & CFD in general, one on NIROM with GPR:

Projection-based ROM (with POD)
Benner P, Gugercin S, Willcox K. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. 2015; 57(4): 483–531.

Projection-based ROM (with AE):
Lee K, Carlberg KT. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. J Comput Phys. 2020; 404: 108973.

Phillips, TRF, Heaney, CE, Smith, PN, Pain, CC. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. Int J Numer Methods Eng. 2021; 1–32.

General - Machine learning and CFD:
Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech. 2020; 52: 477–508.

NIROM with POD and Guassian Process Regression:
D. Xiao, C.E. Heaney, L. Mottet, F. Fang, W. Lin, I.M. Navon, Y. Guo, O.K. Matar, A.G. Robins and C.C. Pain. A reduced order model for turbulent flows in the urban environment using machine learning, *Building and Environment*, 148: 323-337 (2019).

NIROM with AE instead of POD:

Gonzalez FJ, Balajewicz M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems; 2018. arXiv:1808.01346 [math.DS].

R Mojgani, M Balajewicz. Physics-aware registration based auto-encoder for convection dominated PDEs. *arXiv preprint arXiv:2006.15655*.

K Fukami, T Nakamura, K Fukagata. Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids* 32 (9) 095110.

T Murata, K Fukami, K Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics* 882, A13.

S Nikolopoulos, I Kalogeris, V Papadopoulos. Non-intrusive surrogate modeling for parametrized time-dependent PDEs using convolutional autoencoders. *arXiv preprint arXiv:2101.05555*.

NIROM with ANN for time prediction / parameter dependence:

Adil Rasheed, Omer San, Trond Kvamsdal, Digital Twin: Values, Challenges and Enablers From a Modeling Perspective, IEEE Access, 10.1109/ACCESS.2020.2970143, 8, (21980-22012), (2020).

Wiewel S, Becher M, Thuerey N. Latent space physics: towards learning the temporal evolution of fluid flow. Comput Graph Forum. 2019; 38(2): 71- 82. (Some nice animations online.)

SE Ahmed, SM Rahman, O San, A Rasheed, IM Navon. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Physics of Fluids* 31 (12), 126602

O San, R Maulik, M Ahmed. An artificial neural network framework for reduced order modeling of transient flows *Communications in Nonlinear Science and Numerical Simulation* 77, 271-287

S Pawar, SM Rahman, H Vaddireddy, O San, A Rasheed, P Vedula. A deep learning enabler for nonintrusive reduced order modeling of fluid flows. *Physics of Fluids* 31 (8), 085101

Karen Willcox:

KE Willcox, O Ghattas, P Heimbach. The imperative of physics-based modeling and inverse theory in computational science. *Nature Computational Science* 1 (3), 166–168.

MG Kapteyn, KE Willcox. Predictive Digital Twins: Where Dynamic Data-Driven Learning Meets Physics-Based Modeling *International Conference on Dynamic Data Driven Application Systems*, 3–7.

MIG Kapteyn, KE Willcox. From Physics-based models to Predictive Digital Twins via Interpretable Machine Learning. *arXiv preprint arXiv:2004.11356*

MG Kapteyn, DJ Knezevic, DBP Huynh, M Tran, KE Willcox. Datadriven physicsbased digital twins via a library of componentbased reducedorder models. *International Journal for Numerical Methods in Engineering*.

SA Niederer, MS Sacks, M Girolami, K Willcox. Scaling digital twins from the artisanal to the industrial. *Nature Computational Science* 1 (5), 313-320.

Physics-informed methods:

M Raissi, A Yazdani, GE Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations (?) *Science* 367 (6481), 1026–1030.

M Raissi, P Perdikaris, GE Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. (?) *Journal of Computational Physics* 378, 686–707.

M Raissi, GE Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations *Journal of Computational Physics* 357, 125–141.

M Raissi, P Perdikaris, G Karniadakis. Machine learning of linear differential equations using Gaussian processes. (?) *Journal of Computational Physics* 348 (Supplement C), 683–693.

M Raissi, Z Wang, MS Triantafyllou, GE Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics* 861, 119–137.

Original GAN:
I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Networks *arXiv* (2014).

DC GAN:
A. Radford, L. Metz, S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* (2015).

WGAN:
M. Arjovsky, S. Chintala, L. Bottou. Wasserstein GAN. *arXiv* (2017).

WGAN-GP:
I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville. Improved Training of Wasserstein GANs. *arXiv* (2017).

Predicting with conditional GANs:

T. Kadeethum, D. O'Malley, J. N. Fuhg, Y. Choi, J. Lee, H. S. Viswanathan, N. Bouklas A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks (2021) arXiv 2105.13136.

M Cheng, F Fang, CC Pain, IM Navon. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network (2020) *Computer Methods in Applied Mechanics and Engineering* 365, 113000.

J. Morton, M. J. Kochenderfer, F. D. Witherden, Parameter-Conditioned Sequential Generative Modeling of Fluid Flows, AIAA Journal, 10.2514/1.J059315, 59, 3, (825-841), (2021).

Predicting with GANs:

VLS Silva, CE Heaney, Y Li, CC Pain. Data Assimilation Predictive GAN (DA-PredGAN): applied to determine the spread of COVID-19 (2021) *arXiv preprint arXiv:2105.07729*.

VLS Silva, CE Heaney, CC Pain. GAN for time series prediction, data assimilation and uncertainty quantification (2021) *arXiv*.

P. Stinis, T. Hagge, A. M. Tartakovsky and E. Yeung. Enforcing constraints for interpolation and extrapolation in Generative Adversarial Networks (2019) *Journal of Computational Physics* 397 108844.

A.V. Umanovskiy, Generative adversarial neural networks for the heuristic modelling of a two-phase flow in porous media, Computational Continuum Mechanics, 10.7242/1999-6691/2020.13.2.18, 13, 2, (231-241), (2020).

Combinations of ideas from VAE and GAN:

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey. Adversarial Autoencoders. *arXiv* (2015).

D. Ulyanov, A. Vedaldi and V. Lempitsky. It Takes (Only) Two: Adversarial Generator-Encoder Networks. *arXiV* (2017).