Real-Time Reconstruction of Complex Flow in Nanoporous Media: Linear vs Non-linear Decoding

Emmanuel Akeweje[1][0000−0002−1513−623X], Andrey Olhin[2][0000−0002−7714−5895], Vsevolod Avilkin[2][0009−0000−2346−7621], Aleksey Vishnyakov[3,4][0000−0002−1621−3858], and Maxim Panov[5][0000−0001−5161−2822]

[1] Trinity College Dublin, Dublin, Ireland
[2] Skolkovo Institute of Science and Technology, Moscow, Russia
[3] Moscow State University, Department of Physics, Moscow, Russia
[4] Krestov Institute of Solutions Chemistry, Ivanovo, Russia
[5] Technology Innovation Institute, Abu Dhabi, UAE
`eakeweje@gmail.com, maxim.panov@tii.ae`

Abstract. Physical field reconstruction from limited real-time data is a topical inverse problem that attracts substantial research effort, and complex geometries present a formidable challenge. The paper describes a reconstruction of the velocity field of a steady fluid flow through a two-dimensional porous structure from the real-time gauge readings (that is, velocity values obtained at specific fixed locations). The dataset is composed of 300 Lattice-Boltzmann simulations of the flow with different boundary conditions. The number of the gauges and their locations are varied. Two reconstruction techniques are applied: neural network (NN) and linear least squares solver. The linear solver outperforms the NN in terms of both speed and precision. Sensor locations are optimized by Monte Carlo method. The porous structure is mapped onto a graph and the optimization is performed by Metropolis type node-to-node trial displacements of the gauges. With 100 gauges, the linear method enables reconstruction of the velocity field in a porous structure discretized on 256 x 256 2D grid with the normalized error of 0.57%.

Keywords: physical field reconstruction · fluid flow · porous materials · neural networks · MCMC

1 Introduction

In engineering computing, there are situations when the laws governing the behavior of a system are well known, and its behavior can be predicted with good accuracy, but the first-principle solution is complicated either by a lack of information on the initial/boundary conditions, or by computational expenses (for example, the response is needed within seconds, whereas calculations take hours). At the same time, experimental real-time monitoring using gauges is possible only in a limited number of locations and/or at select moments in time due to equipment-related or economic reasons. Real-time monitoring requires fast restoration of the values in the locations where direct observations are unavailable from the available readings. Modification of this problem is restoration of properties that are difficult to measure from the properties easy to monitor. An obvious extension is the optimization of the gauge locations.

Physical field restoration is a classical inverse problem: the direct problem can be solved using deterministic methods with a reasonable accuracy given the initial and/or boundary conditions, but condition the current gauge readings may correspond to are unknown. The problem may or may not be well-posed: that is, there is no guarantee that similar readings cannot be obtained with different set of conditions. The reconstruction problem is of a great practical importance: building monitoring systems, temperature tracking for processors and chips, water resource monitoring, and industrial processes control require reconstruction of velocity, pressure, temperature and magnetic fields [1–3]. A number of data-driven approaches to the physical field restoration problem have been developed over the last few years, most of them quite recently.

In many applications, field reconstruction tasks are approached using a low dimensional representation of the field [4–9]. This low-dimensional representation can be viewed as a tailored basis of the field such as proper orthogonal decomposition (POD) basis, Fourier basis and wavelet basis. For efficient reconstruction of thermal field, Li et al. [5] obtained low dimensional representations of the high dimensional temperature distribution state of the thermal maps via POD techniques and then implemented a greedy algorithm for optimal sensor placement. Willcox [8] extended the gappy POD method to handle unsteady flow reconstruction. Tan et al. [10] demonstrated the application of POD in an iterative procedure to reconstruct incomplete or inaccurate aerodynamic data.

The main challenge that POD methods face is the sensitivity of the reconstructions to the location of the sensors. Neural networks (NNs) promise more stable solutions in physical field reconstruction [11–14], although they are substantially more computationally expensive to train. Some of the NN-based schemes for related CFD problems directly take into account the physical laws that govern the system behaviour [15–17], the others do not [18–21]. Erichson et al. [11] proposed a simple shallow neural network based learning algorithm for reconstruction. The shallow neural network learned an end-to-end mapping between a set of sensor measurements and the high-dimensional fluid flow field from which the measurements were taken, without requiring special data preprocessing. More sophisticated NNs were employed to tackle the field reconstruction problem in [12]; the authors employed reshaping operations to allowed for the possibility of harnessing the performance of some state of the art convolutional image models as opposed to just the fully connected network architecture. Two of their models have architectures similar to that of U-Net; with one of them having Fourier Neural Operator layers. NN-based inversion methods are powerful for learning and are increasingly used field reconstruction [13, 22, 23].

Reconstruction of physical fields from limited sensor measurements from the system could be expressed as a map of any form, be it linear or non-linear.

Clark et al. [24] approached the reconstruction problem with an algorithm based on linear maps and incorporated this algorithm in a greedy scheme for sensor placement. Thus, a field reconstruction technique strictly by linear map was introduced. This technique, even though yet to be explored in many applications, is advantageous for reconstruction problems in that it allows to simply determine the stability and optimality of a collection of (sensor) measurements, and also handy for building effective algorithms for optimal sensors placement. Despite the linearity constraint, this technique yields good reconstructions even with measurements from randomly placed sensors [24, 25].

In this study, we consider data-driven techniques for reconstruction of steady flow in complex porous geometry from limited sensor readings and a dataset of velocity fields calculated for 300 different sets of boundary conditions. Most literature considered flow state reconstruction in simple geometries. Systems of complex morphology are more challenging to work with. The main **contributions** of this work are as follows.

– Using a least squares based linear solver and a neural network, we reconstruct the stationary state of the system from limited sensor measurements, see Sections 3 and 4.
– We optimize the location of the sensors and determine the minimum number of sensors sufficient for the reconstruction with a given precision, see Section 5.

2 Fluid flow in 2D porous medium

As a case study, we selected reconstruction of the velocity field of a 2D flow in a model porous media. The pore structure is generated using the Porespy package [26]. The algorithm generates random noise, then applies a gaussian blur to the noise. Parameter σ controlled by the desired level of sample blobiness as $\sigma = \bar{l}/(40 \cdot \text{blobiness})$, where $\bar{l}$ is mean value (in voxels) of the set width and height of the sample image. After that, we re-normalizing acquired data to uniform distribution. The resulting geometry is shown in Figure 1. The binary structure is discretized on 256×256 voxels (each site is either filled or open), has a 0.72 porosity ratio and the blobiness of 1 in all direction.

On the north and south borders, periodic boundary conditions are applied. A constant pressure difference of 0.0005 (Lattice units) is maintained between the west and the east boundaries to establish a steady flow. The relaxation ratio is $\tau = 1.0$. The parameters are chosen so that the value of $u_x^{\max} < 0.2$ in the resulting velocity profile does not exceed the stability limits of the numerical simulation. In each simulated system, a constant velocity profile is maintained at the western boundary. The velocity profile is generated in a random fashion using harmonic function as $u_x(y) = \sum_i A_i \sin(k_i y + 2\pi\xi)$,where $A_i = a\big(1 + (k_i L_0)^2\big)^{-1/2}$ are the amplitudes of the modes, ξ is random number with uniform distribution $[0, 1]$, $k_i = \frac{1}{2}\frac{2\pi}{L} i$ is the wave number. The parameter L_0 is the characteristic length of the undulations. The sample velocity profiles

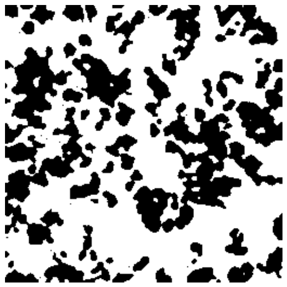


Fig. 1. The pore structure used as a case study for this work.

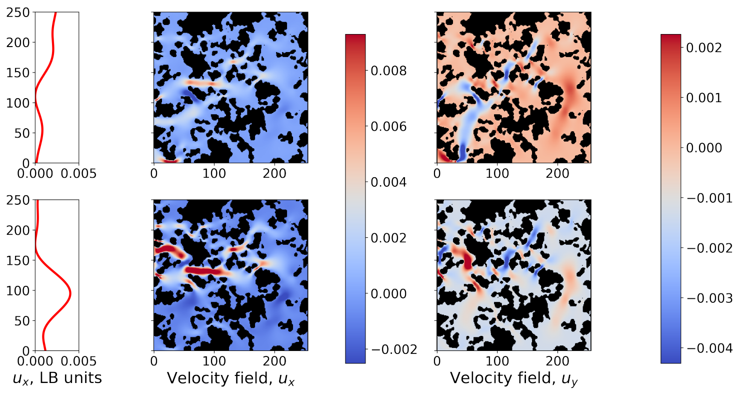


Fig. 2. The figure shows two examples of the velocity profile at the western boundary (left panels) and corresponding steady velocity fields in x (center panel) and y (right panel) dimensions

are shown at Figure 2. The velocity fields for each profile are generated by the LB simulations with BGK approximation of the Boltzmann equation for 2D, 9 discrete velocity model (D2Q9) [31]. Each simulation is carried out over 5000 iterations on average. The simulations are assumed to converge to the running average of the standard deviation of the energy over 500 iterative steps, decreased to $\epsilon = 10^{-4}$. In total, 300 random profiles are generated, and for each of them the steady state velocity fields are calculated. This dataset [32] is used for reconstruction of the velocity field from the gauge readings.

2.1 Mathematical framework for the velocity field reconstruction

The objective of velocity field reconstruction is to learn the relationship between gauge readings $X \in \mathbb{R}^{m\times 2}$ and velocity field data $Y \in \mathbb{R}^{n\times 2}$, with a constrain of limited gauge number $m \ll n$. For this study, each row of X and Y consist of velocity data in both x and y directions at a certain location (point) in the velocity field. The sensor measurements $\boldsymbol{x}_i \in X,\ 1 \le i \le m$ are collected from

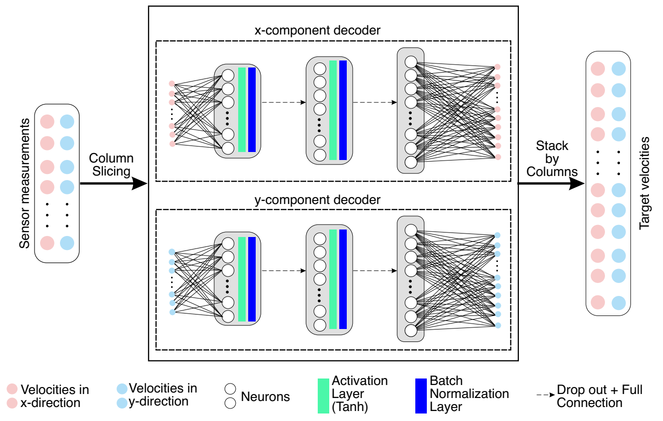


Fig. 3. Architecture of Shallow Neural Decoder.

the high-dimensional field Y via a sampling procedure. This procedure can be described as:

$$X = \Phi(Y),$$

where $\Phi\colon \mathbb{R}^{n\times 2} \rightarrow \mathbb{R}^{m\times 2}$ denotes a measurement operator. Typically, the measurement operator could be related with a binary matrix used to mask out some flow data. Therefore, $X = MY$, where $M \in \mathbb{R}^{m\times n}$ is a matrix whose m rows comprise of the j-th standard basis vector of $\mathbb{R}^n$, i.e. vectors $\boldsymbol{e_j}$ with j representing the location index of the measurement in Y.

The task at hand requires the construction of an inverse model, which generates the state Y from observations X, described as $Y = \Gamma(X)$, where $\Gamma\colon \mathbb{R}^{m\times 2} \rightarrow \mathbb{R}^{n\times 2}$ denotes the forward operator. This problem is frequently ill-posed, and we cannot invert the measuring operator Φ to get the forward operator Γ immediately. However, a function Ψ to approximate the forward operator Γ can be derived from a set of available sensor measurement and velocity field data, that is $\{X_i,\ Y_i\}_{i=1}^{k}$, where k is the number of flow snapshot available in the dataset. In particular, the objective is to learn a function $\Psi\colon X \rightarrow Y$ that maps a limited number of measurements to a predicted state Y,

$$\hat{Y} = \Psi(X), \tag{1}$$

so that the misfit is minimized, for instance in a Euclidean sense, over all sensor measurements $\|\Psi(X) - \Gamma(X)\|^2 < \epsilon$, where ϵ is a very small positive number. In this study, Ψ is taken to be a regression function: shallow neural network or linear solver.

3 Shallow neural decoder

We first consider estimating Ψ in equation (1) with neural networks that are generally labelled as universal estimators. Taking inspiration from the works of

no. of	model hyperparameters			training hyperparameters				
sensors	1st hidden	2nd hidden	output	batch			scheduler	
(m)	layer size	layer size	layer size	size	lr	betas	step	gamma
10	200	1000	10686	32	0.005	(0.9, 0.95)	50	0.5
50	200	1000	10646	32	0.001	(0.95, 0.95)	-	-
500	200	500	10196	16	0.05	(0.95, 0.99)	100	0.5
996	200	500	9700	32	0.01	(0.95, 0.95)	100	0.5

Table 1. Hyperparameters for the optimal SNDs. There are no scheduler parameters for the SND for $m = 50$ sensors because the model was trained without scheduling the learning rate.

Erichson et al. [11], we developed a neural network architecture as in Figure 3; a common architecture for decoders consists of layers of non-decreasing sizes, which increase the size of the representation from low-dimensional observations to the high-dimensional field in a continuous manner. Hence, it is named as Shallow Neural Decoders (SND). The SND consists of three hidden layers, activation layers to introduce non-linearity, batch normalization layers and drop out. For the activation function, we found Tanh to to work better in our experiments, however other choices such as ReLU could produce more efficient result for other applications. The velocity of the flow in our system is two-dimensional, and for this reason we considered a separated-decoder architecture, such that the model learns the components of the target velocities independent of the other. Thus, the SND consists of x-component and y-component decoders as shown in Figure 3.

Setting the number of sensors, m, to 10, 50, 500, and 996, we experimented with 3-layered SNDs. Adam optimization technique was used to obtain the model parameters which minimize the mean squared error loss function. Training the SNDs heavily depend on the hyperparameters, and to obtain the optimum SND, we repeatedly trained the SNDs with different combinations from a range (or set) of each of the hyperparameter: learning rate, betas, batch size, number of layers in hidden units, scheduler step, and scheduler gamma. The combinations of hyperparameters, which yield the optimum decoder for each number of sensor m is presented in Table 1.

Metrics. In this study, three metrics: the normalized error (NE), normalized fluctuation error (NFE) and coefficient of determination (also known as R^2), are employed to evaluate the performances models and algorithms. The metrics are defined as follow:

$$\mathrm{NE}(Y,\hat{Y}) = \frac{\|Y-\hat{Y}\|_2}{\|Y\|_2}, \quad \mathrm{NFE}(Y,\hat{Y}) = \frac{\|Y'-\hat{Y}'\|_2}{\|Y'\|_2} = \mathrm{NE}(Y',\hat{Y}'),$$
$$\mathrm{R}^2(Y,\hat{Y}) = 1 - \frac{\sum_{i=1}^n (Y_i-\hat{Y}_i)^2}{\sum_{i=1}^n (Y_i-\bar{Y})^2} = 1 - \frac{\|Y'-\hat{Y}'\|_2^2}{\|Y'\|_2^2} = 1 - \mathrm{NFE}^2(Y,\hat{Y}),$$

where Y is the ground-truth velocity, $\hat{Y}$ is model prediction, $Y' = Y - \bar{Y}$, $\hat{Y}' = \hat{Y} - \bar{Y}$, and $\bar{Y}$ is the empirical mean. NE penalizes over-estimations and

	Training set			Validation set			Testing set		
m	NE	NFE	R^2	NE	NFE	R^2	NE	NFE	R^2
10	0.326	0.546	0.702	0.303	0.490	0.760	0.274	0.489	0.761
50	0.146	0.243	0.941	0.148	0.240	0.942	0.148	0.264	0.930
500	0.033	0.055	0.997	0.039	0.064	0.996	0.032	0.058	0.997
996	0.051	0.085	0.993	0.054	0.087	0.992	0.057	0.102	0.990

Table 2. Performance of SNDs.

under-estimations in model prediction. NFE eliminates the possible dominating empirical mean and focuses on the fluctuations in the field. Lastly, the coefficient of determination (R^2) checks how much of the variability in the dataset is explainable by a model. Using these metrics, the error in the flow state reconstruction can be quantified, and model performance is evaluate. We sought for models with the lower NEs, lower NFEs, and higher R^2 scores.

Results. The synthesized velocity dataset, as described in Section 2, was split into three sets such that 70% was used for training, 20% for validation and 10% for testing the models. As common with other neural networks, the validation set is to keep the model in check whilst training to prevent overfitting on the training set. We down-sampled (coarse-grained) each snapshots in the dataset to 128×128 for computational reasons, this resulted in having 10,696 velocity points in the "porous space". The performance of the SNDs given the number of sensors is presented in the Table 2. Although more sensor measurements generally produced better reconstruction, the results from the SND reconstructions reflects that reconstruction with 500 sensor measurement is better than that of 996 sensor measurements. This is an indication that for the system, there may be a sufficient number of sensor required for an efficient field reconstruction. With 500 sensor measurements, the SND produced reconstruction with R^2 scores of 0.997 on the testing set. Figure 4 is a sample of reconstruction produced by SND from the testing set.

4 Least squares linear algorithm

A major concern with the neural decoders for flow state reconstruction tasks is the need to always retrain the decoders whenever the sensor positions are altered. Some optimization problems related to physical field reconstruction includes identification of the optimal positions to place sensors, and having knowledge of the number of sensors sufficient for such reconstructions. Retraining neural decoders is usually computational expensive. This concern motivated us to consider algorithms which requires less computation resources for modeling the relationship between sensor measurements and target velocities.

It is quite easy to spot the important role of least squares solutions in classical reconstruction methods such as POD. Without the least squares solutions, reconstructions with tailored basis will become intractable. However, by imposing

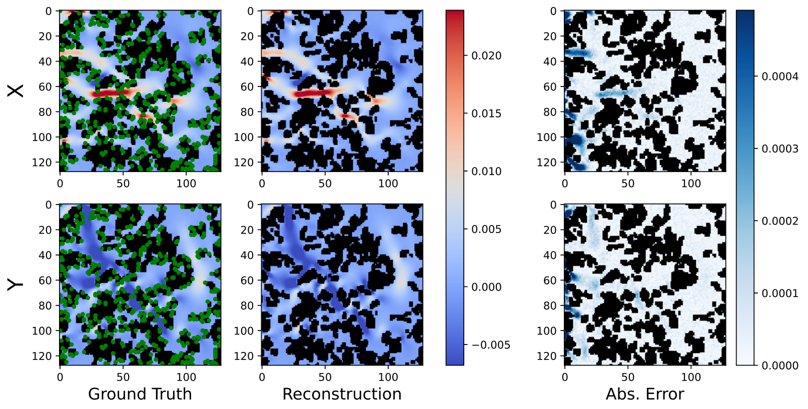


Fig. 4. SND reconstruction from 500 sensors. The green pixels represent the position of the sensors in the velocity field, the sensors were placed at randomly picked spots on walls of the porous medium for this experiment.

a linear relationship between sensor measurements and corresponding (missing) velocity information, the least square method alone can sufficiently reconstruct the state of the flow field. Hence, the name LS-decoder. The linear relationship in LS-decoder requires that the velocity data matrices, Y and X, be vectorized before imposing the linearity assumption. This algorithm is similar to the exact Dynamic Mode Decomposition where the Koopman operator could be approximated by least squares technique [27].

Let us consider a snapshot which consists of $m + n$ points such that there are m sensors which measures the velocity in both x- and y- directions, that is a total of $2m$ measurements would be recorded. To reconstruct the velocity field requires estimating missing $2n$ point velocities. Mathematically, these tall vectors are constructed from the velocity data matrices by vectorization:

$$\mathbf{x} = \text{vectorize}(X) \in \mathbb{R}^{2m}, \quad \mathbf{y} = \text{vectorize}(Y) \in \mathbb{R}^{2n}.$$

The linearity relationship then implies that $\mathbf{y} \approx \tilde{A}\mathbf{x}$ with $\tilde{A} \in \mathbb{R}^{2n\times 2m}$. Typically, $m \ll n$. Matrix $\tilde{A}$ is the (rectangular) matrix of importance coefficients a_{ij}. The importance coefficient, a_{ij}, is interpreted as the contribution of the sensor measurement indexed j in $\mathbf{x}$ to estimation of velocity at missing point indexed i in $\mathbf{y}$. Again, this relationship means that each velocity estimate is computed as a weighted sum of available sensor measurements. Typically, we expect that the proximity of the points at which velocity is been estimated to the gauges directly influence the importance of each gauge in the estimation. The coefficient matrix is computed strictly via data-driven approach. To obtain a linear operator which best fits the data, we adopt the method of snapshot method from DMD technique [27]. We structure the data in two snapshot matrices $\tilde{X}$ and $\tilde{Y}$. That

```
Algorithm 1 Velocity reconstruction with LS-decoder
Inputs: Set of sensor measurements $\{X_i \in \mathbb{R}^{m\times 2}\}_{i=1}^{k}$ and set of target velocities
$\{Y_i \in \mathbb{R}^{n\times 2}\}_{i=1}^{k}$ from the training dataset of flow snapshots $\{S_1, S_2, \cdots, S_k\}$
Output: Reconstruction operator $\tilde{A} \in \mathbb{R}^{2n\times 2m}$.

1. Vectorize sensor measurements: $\{\mathbf{x}_i \in \mathbb{R}^{2m}\}_{i=1}^{k}$.
2. Vectorize target velocities: $\{\mathbf{y}_i \in \mathbb{R}^{2n}\}_{i=1}^{k}$.
3. Construct $\tilde{X}$ by stacking the sensor measurements together as columns:
   $\tilde{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_k] \in \mathbb{R}^{2m\times k}$.
4. Construct $\tilde{Y}$ by stacking the target velocities together as columns:
   $\tilde{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_k] \in \mathbb{R}^{2n\times k}$.
5. Construct the pseudo-inverse of $\tilde{X}$: $\tilde{X}^\dagger$.
6. Compute the reconstruction operator: $\tilde{A} = \tilde{Y}\tilde{X}^\dagger$.
```

is, given that there are k training snapshots, then $\tilde{Y} \in \mathbb{R}^{2n\times k}, k \ll 2n$ has k columns, and each column has a length of $2n$ since $\mathbf{y} \in \mathbb{R}^n$.

Considering the training data and the linear relationship between sensor measurements and the (missing) velocity information, we can write a compact expression as below:

$$\begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_k \\ | & | & & | \end{bmatrix} \approx \tilde{A} \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_k \\ | & | & & | \end{bmatrix}, \tag{2}$$

$$\tilde{Y} \approx \tilde{A}\tilde{X}. \tag{3}$$

This formulation makes it possible to compute the operator $\tilde{A}$ across the training set, and thus the operator can be used in reconstruction of the velocity field. Recall that the number of sensors available is very limited, thus the linear map is under-determined and has many solutions. However, given this new formulation in equation (3), we are interested in the "best fit" solution of

$$\min_{\tilde{A}} \|\tilde{Y} - \tilde{A}\tilde{X}\|_{\mathrm{F}} ,$$

in the Frobenius sense, which has the standard form $\tilde{A} = \tilde{Y}\tilde{X}^\dagger$ where $\tilde{X}^\dagger$ is the Moore-Penrose pseudo-inverse of $\tilde{X}$. The steps for implementing LS-decoder algorithm are highlighted in Algorithm 1.

Results. The dataset was separated into 2 sets: 70% for training and 30% for testing set. Validation set is not necessary for this algorithm since the model training is non-iterative. The performance of the reconstruction algorithm on these sets are recorded in Table 3. With 500 and 996 sensor, R^2 scores are almost perfect, and the errors are minimal. Figure 5 displays a sample reconstruction using the LS decoder with 500 sensor measurements. The absolute error in

	Training set			Testing set		
m	NE	NFE	R^2	NE	NFE	R^2
10	0.1124	0.1892	0.964	0.1264	0.2119	0.955
50	0.0371	0.0625	0.996	0.0412	0.0692	0.995
500	0.0017	0.028	1.000	0.0033	0.0055	1.000
996	0.0021	0.0035	1.000	0.0036	0.0060	1.000

Table 3. Performance of LS-decoder

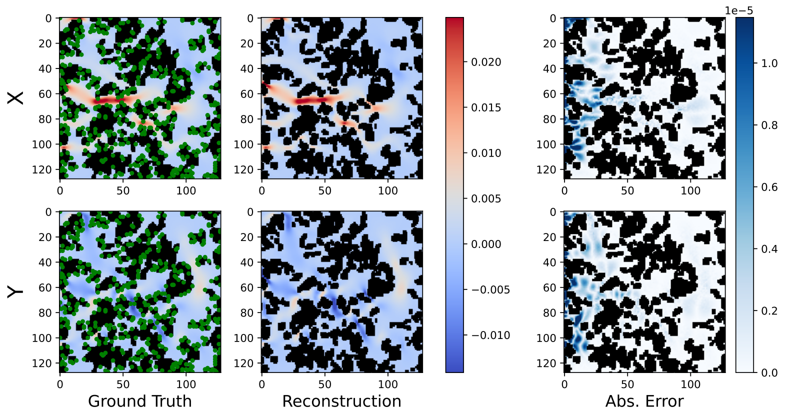


Fig. 5. LS-decoder reconstruction from 500 sensors. The green pixels represent the position of the sensors in the velocity field, the sensors were placed at randomly picked spots on walls of the porous medium for this experiment.

Figure 5 reflects the precision of the reconstruction as the error are very low. LS-decoder also recorded impressive performances with just 10 and 50 sensors.

As with other deep learning models, back-propagation mechanism is employed to train the SNDs, that is to minimize the misfit between SNDs predictions and target velocities. Whereas for linear (LS decoder) algorithm, a standard minimum norm solution which is readily available is adopted. Obtaining the parameters of SNDs requires an iterative process of gradually approaching a potential solution which heavily depends on the choice of hyperparameters. The search for the optimal SNDs is computational expensive. The linear algorithm, on the other hand, does not involve such iterative process. The velocity field reconstruction with the linear algorithm requires lesser computation resources and time.

5 Monte Carlo optimization of gauge placement

The ability to swiftly re-train the LS decoder and calculate the error allows for the optimization of sensor placement. The number of gauges in the system

Algorithm 2 Metropolis Monte-Carlo algorithm

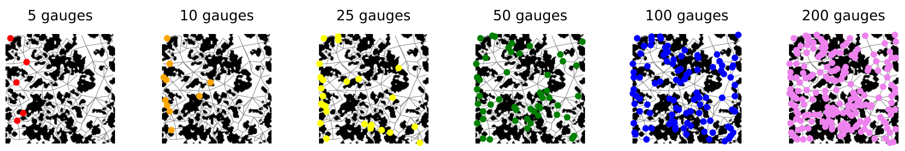
```
Inputs:
Initial gauge distribution.
Number of gauges $N_g \in [5, 10, 25, 50, 100, 200]$.
Number of iterations $N_{it} = 10000$.
Temperature parameter $T \in \{1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}, 5e^{-3}, 1e^{-2}, 5e^{-2}, 1e^{-1}\}$.
Output:
Optimal gauge positions.

Initialize the gauge distribution.
Cycle for $N_{it}$ iterations:
    1. Select a random gauge $\mathbf{A}$.
    2. Calculate an error of the approximation of the velocity field $\mathbf{E_A}$ for current gauge locations.
    3. Shift the gauge to random unoccupied node $\mathbf{B}$ near the gauge $\mathbf{A}$.
    4. Calculate an error $\mathbf{E_B}$ for the gauge locations after the shift.
    5. Calculate the transition probability $p_{\mathrm{A}\to\mathrm{B}} = \min\left(1, \frac{e_\mathrm{B}}{e_\mathrm{A}} \exp\left(\frac{E_\mathrm{B}-E_\mathrm{A}}{T}\right)\right)$.
    6. Generate a uniform random number $u \in [0, 1]$. Accept the gauge shift if $u > p_{\mathrm{A}\to\mathrm{B}}$, reject otherwise.
```

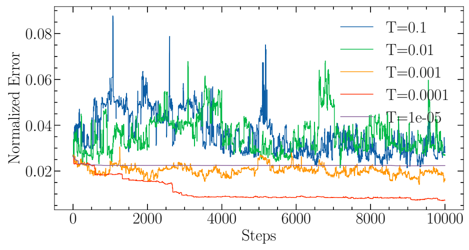
was varied from 5 to 500, which means up to 1000 independent gauge coordinates. MC methods are known for efficiency with large number of parameters. To optimize the gauge placement, the Canonical Metropolis MC method [28] is applied. The pore structure is presented as a graph network which was extracted via a marker-based segmentation algorithm known as the Sub-Network of an Over-segmented Watershed (SNOW) algorithm [29]. We implemented the SNOW algorithm made available in Porespy package [26] to extract the pore network of our 2D porous structure. In the initial configuration, the gauges are uniformly placed at the nodes, then the field is reconstructed and the normalized error (NE) calculated. The gauge locations are modified in a Markov stochastic process. At each step, we attempt a move of a randomly selected gauge located at node A to a neighboring node B along a randomly selected edge. The move is accepted with the probability of

$$p_{\mathrm{A}\to\mathrm{B}} = \min\left(1, \frac{e_\mathrm{B}}{e_\mathrm{A}} \exp\left(\frac{E_\mathrm{B}-E_\mathrm{A}}{T}\right)\right),$$

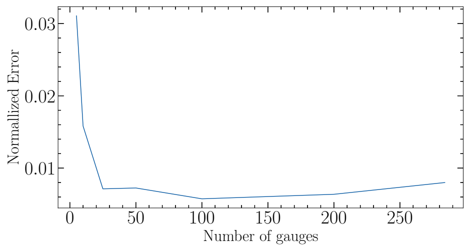
where E_A is the error of the approximation of the velocity field with the LS solver (of course, it is a function of coordinated of all gauges in the system), T is an optimization parameter (temperature), e_A and e_B are the number of edges that each node forms. If $T = 0$ only the moves leading to the improvement of approximation are accepted and thus the system never escapes any local minimum, $T \to \infty$ means any random displacement is accepted. The $e_\mathrm{B}/e_\mathrm{A}$ factor compensates for the asymmetry of the transition matrix to make the



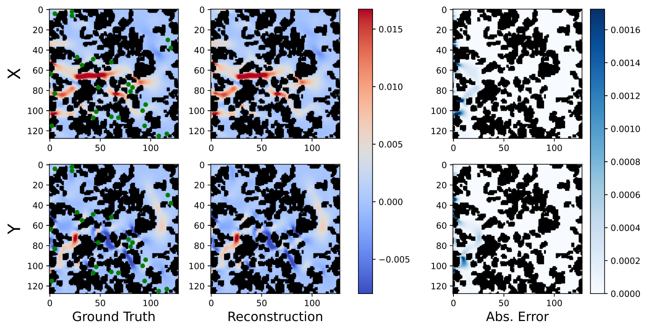
(a)



(b)



(c)



(d)

Fig. 6. (a) optimal locations of the gauges obtained in Monte Carlo optimization with $m = 5$(red), $m = 10$(orange), $m = 25$(yellow), $m = 50$(green), $m = 100$(blue) and $m = 200$(purple) (b) dependence of the error on time for $m = 25$ and $T \in [1e^{-5}, 1e^{-4}, \dots, 1e^{-1}]$ (c) dependence of the error on m for the optimal gauge placement (d) LS-decoder reconstruction from 25 sensors. The green pixels represent the position of the sensors in the velocity field, the sensors were placed optimally at the nodes of the graph network for this experiment.

algorithm ergodic and obey the detailed balance [30]. The steps of the Metropolis algorithm are highlighted in Algorithm 2.

Figure 6(b) shows the optimization progress for Metropolis stochastic processes with $m = 25$ gauges and different values of T. Starting uniform gauge distribution is not an optimal one. For high T values (> 0.01) the acceptance rate is too high and the NE does not seem to decrease. If the T value is too low ($1e^{-5}$), acceptance rate is too low and the NE converges to a non-optimal

	Metropolis algorithm			Best out of 10000		
m	NE	NFE	R^2	NE	NFE	R^2
5	**0.0311**	**0.0521**	**0.99728**	0.0431	0.0722	0.99479
10	**0.0158**	**0.0265**	**0.99930**	0.0277	0.0464	0.99785
25	**0.0071**	**0.0120**	**0.99986**	0.0172	0.0288	0.99917
50	**0.0072**	**0.0122**	**0.99985**	0.0124	0.0208	0.99957
100	**0.0057**	**0.0096**	**0.99991**	0.0079	0.0133	0.99982
200	**0.0064**	**0.0107**	**0.99989**	0.0065	0.0109	0.99988
284	0.0080	0.0134	0.99981	0.0080	0.0134	0.99981

Table 4. Performance of LS-decoder for optimal gauge disposition obtained with the Metropolis algorithm and picking the best gauge locations out of 10000 random ones

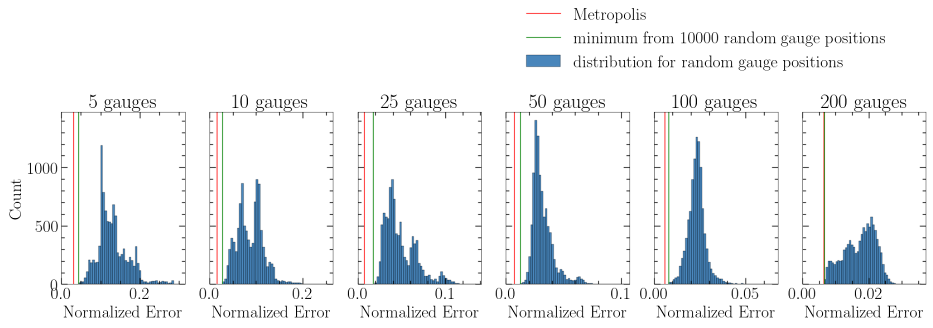


Fig. 7. Comparison of the NE distributions for random gauge position selection and result of MC simulations

local minimum. The optimal T for this algorithm and m value is close to $1e^{-4}$, which corresponds to acceptance probability equal to 56%. The fact that the system reaches the “equilibrium” and high acceptance ratio for the attempted displacement moves means there many sets of gauge coordinates that allow approximately similar reconstruction quality. The coordinates corresponding to the lowest error are recorded as the optimal sensor locations, which are also shown in Figure 6(a). The optimal locations are by no means counter-intuitive in this system: the gauges occupy the key "straights" and mostly located near the western border where the velocity profiles are set. As expected, the reconstruction quality improves with the number of gauges. Improvement is very fast when there are just a few gauges in the system, and slows down as the number of gauges reaches 25. As the number of gauges grows beyond the optimal number of 100, the NE rises slowly, which could be seen at Figure 6(c) or in Table 4.

The Metropolis algorithm could be compared to the random selection of gauge positions. On the Figure 7 it is shown that gauge positioning greatly affects the NE and the Metropolis algorithm gives better results than just selecting the best one from the random selection. It is especially reasonable for low number of gauges with a big variability in gauge positions.

6 Conclusion

In this work, we applied fast shallow decoders to reconstruction of a velocity field for complex steady flow through an irregular porous medium using a database of 300 simulated flows with different boundary conditions and gauge readings from a selected number of locations. Two decoders were applied: shallow nonlinear neural decoder and linear least square decoder. Despite a non-linearity of the Navier-Stokes equation that governs the flow, the linear least square solver outperformed the neural network in both precision and speed. The locations of the gauges are optimized with the Metropolis Monte-Carlo algorithm. It was found that 25 gauges are sufficient to reconstruct the velocity field in 256×256 2D grid with about 99% precision; taking precision with respect to NFE. The strategy developed here can be applied to the monitoring of water resources, pipe circuits at chemical plants or applied in anomaly detection.

Individual contributions: E.A.: NN and LS implementation, text; A.O.: LB simulations; V.A.: MC optimization; M.P.: conceptualization, ML methodology, text; A.V.: conceptualization, methodology, text. Authors declare no conflict of interests.

References

1. Zhou, H. et al.: An information-theoretic framework for optimal temperature sensor allocation and full-chip thermal monitoring. In Proceedings of the 49th Annual Design Automation Conference 2012, pp. 642–647
2. Reda, S. et al.: Improved thermal tracking for processors using hard and soft sensor allocation techniques. IEEE Trans. Computers, **60**(6), 841-851 (2011)
3. Ranieri, J. et al.: Near-optimal thermal monitoring framework for many-core systems-on-chip. IEEE Transactions on Computers, **64**(11), 3197-3209 (2015)
4. Clenet, S., Henneron, T., Korecki, J.: Sensor placement for field reconstruction in rotating electrical machines. IEEE Trans. Magnetics, **57**(6), 1-4 (2021)
5. Li, B., Liu, H., Wang, R.: Data-driven sensor placement for efficient thermal field reconstruction. Science China Technological Sciences, **64**(9), 1981-1994 (2021)
6. Chaturantabut, S., Sorensen, D. C.: Nonlinear model reduction via discrete empirical interpolation. SIAM Journal on Scientific Computing, **32**(5), 2737-2764 (2010)
7. Bui-Thanh, T. et al.: Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. AIAA journal, **42**(8), 1505-1516 (2004)
8. Willcox, K.: Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. Computers & fluids, **35**(2), 208-226 (2006)
9. Everson, R., Sirovich, L.: Karhunen–Loeve procedure for gappy data. JOSA A, **12**(8), 1657-1664 (1995)
10. Tan, B. T., Willcox, K. E., Damodaran, M.: Applications of proper orthogonal decomposition for inviscid transonic aerodynamics. AIAA journal, 4213 (2003)
11. Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., Kutz, J. N.: Shallow neural networks for fluid flow reconstruction with limited sensors. In: proceedings of the Royal Soc. A, 476(2238), 20200097 (2020)
12. Özbay, A. G., Laizet, S.: Deep learning fluid flow reconstruction around arbitrary two-dimensional objects from sparse sensors using conformal mappings. AIP Advances, **12**(4), 045126 (2022)

13. Yu, J., Hesthaven, J. S.: Flowfield reconstruction method using artificial neural network. AIAA J., **57**(2), 482-498 (2019)
14. Li, Y., Liu, Z.m Wang, Y., Liu, Y., Xie Y.: Real-time physical field reconstruction for nanofluids convection using deep learning with auxiliary tasks. Numerical Heat Transfer, Part A: Applications, **83**(2), 213-236 (2023) https://doi.org/(10.1080/10407782.2022.2091359)
15. Raissi, M., Yazdani, A., Karniadakis, G. E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367, 1026-1030 (2020)
16. Raissi, M., Yazdani, A., Karniadakis, G. E.: Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data (2018). arXiv preprint arXiv:1808.04327.
17. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G. E.: Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica, 37(12), 1727-1738 (2021)
18. Hennigh, O.: Lat-net: compressing lattice Boltzmann flow simulations using deep neural networks (2017). arXiv preprint arXiv:1705.09036.
19. Guo, X., Li, W., Iorio, F.: Convolutional neural networks for steady flow approximation. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining 481-490 (2016)
20. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.: Learning to simulate complex physics with graph networks. In International conference on machine learning (pp. 8459-8468). PMLR (2020)
21. Chen, J., Hachem, E., Viquerat, J.: Graph neural networks for laminar flow prediction around random two-dimensional shapes. Phys. Fluids, 33(12), 123607 (2021)
22. Fukami, K., Fukagata, K., Taira, K.: Super-resolution reconstruction of turbulent flows with machine learning. J. Fluid Mech., 870, 106-120 (2019)
23. Carlberg, K. T., Jameson, A., Kochenderfer, M. J., Morton, J., Peng, L., Witherden, F. D.: Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning. J. Comp. Phys., 395, 105-124 (2019)
24. Clark, E., Askham, T., Brunton, S. L., Kutz, J. N.: Greedy sensor placement with cost constraints. IEEE Sensors Journal, **19**(7), 2642-2656 (2018)
25. Koo, B., Son, H., Kim, H., Jo, T., Yoon, J. Y.: Model-order reduction technique for temperature prediction and sensor placement in cylindrical steam reformer for HT–PEMFC. Applied Thermal Engineering, 173, 115153 (2020)
26. Gostick, J. T., Khan, Z. A., Tranter, T. G., Kok, M. D., Agnaou, M., Sadeghi, M., Jervis, R.: PoreSpy: A python toolkit for quantitative analysis of porous media images. J. Open Source Software, **4**(37), 1296 (2019). https://doi.org/(10.21105/joss.01296)
27. Proctor, J. L., Brunton, S. L., Kutz, J. N.: Dynamic mode decomposition with control. SIAM Journal on Applied Dynamical Systems, **15**(1), 142-161 (2016)
28. Metropolis, N., Ulam, S.: The monte carlo method. Journal of the American statistical association, **44**(247), 335-341 (1949)
29. Gostick, J. T.: Versatile and efficient pore network extraction method using marker-based watershed segmentation. Physical Review E, **96**(2), 023307 (2017)
30. Frenkel, D., Smit, B.: Understanding molecular simulation: from algorithms to applications (Vol. 1). Elsevier, (2001)
31. Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, Erlend Magnus Viggen: The Lattice Boltzmann Method: Principles and Practice. Springer(2017), ISBN 978-3-319-44649-3
32. Olhin, Andrey: Lattice Boltzmann velocity fields dataset, Mendeley Data, V2, (2023). https://doi.org/(10.17632/kbrprbvtjw.2)