Construction of a Fluid Flowfield from Discrete Point Data using Machine Learning

Yury Lebedev*
University of Florida, Gainesville, FL 32611

Michael W. Lee†
NASA Langley Research Center, Hampton, VA 23681

Alina Zare‡
University of Florida, Gainesville, FL 32611

Many verification and validation procedures in aerospace engineering involve the comparison of computational fluid dynamics (CFD) data to experimental results from sources like wind tunnel tests. However, an incongruity exists between the data available from these sources: flow visualization is available by default in computational data, whereas in most experimental setups the available data are far more discrete and far more limited: integrated forces and moments, discrete pressure and temperature probes, etc. When differences exist between quantities of interest like lift and drag coefficients, the lack of full-field flow data from the experiments complicates most attempts to reconcile why the different data sources disagree. To this end, a shallow neural network, constrained by certain fluid flow properties, was trained to approximate flow field snapshots given only discrete data like that available in a wind tunnel test. The constructed snapshots, even for complex incompressible fluid flows, were found to agree at the large scales with the true flow fields. With this tool, researchers can more readily and easily understand why quantities of interest differ between their experimental and computational datasets. This in turn improves the resulting data's uncertainty measures.

Nomenclature

G	=	nonlinear forward operator
H	=	measurement operator
p	=	grid point
R	=	coordinatewise scalar nonlinear activation function
Re	=	Reynolds number
s	=	sensor measurement set
W	=	weight matrix
x	=	spatially resolved flowfield information
$\hat{x}$	=	reconstructed flowfield information
z	=	hidden layer
$\mathbb{R}$	=	real number space
ϕ	=	spatial mode
Φ	=	spatial modal matrix
λ	=	constraint penalty parameter
ν	=	modal coefficient
ν	=	modal coefficient matrix
Ω	=	linear output layer
CFD	=	Conputational Fluid Dynamics
DMD	=	Dynamic Mode Decomposition

*Doctoral Candidate, Department of Electrical and Computer Engineering, University of Florida.
†Research Aerospace Engineer, Configuration Aerodynamics Branch, NASA Langley Research Center, AIAA Member.
‡Professor, Department of Electrical and Computer Engineering, University of Florida.

EDL = Entry, Descent and Landing
MLUQ = Machine Learning with Embedded Uncertainty Quantification
POD = Proper Orthogonal Decomposition
ROM = Reduced-order Model

I. Background and Motivation

MYRIAD engineering design applications demand ever-increasing fidelity in predictive tools, resulting in ever-rising development costs and ever-advancing computational tools. Yet, experimental data upon which the computational efforts are validated remains limited in its structure. Where computational fluid dynamics (CFD) presents the entire flowfield as a function of space and time - with limited fidelity - experimental data are often constrained to integrated forces and moments and a handful of velocity points sampled in the flow domain. Even advances in pressure-sensitive paint [1] yield limited understanding of how the volumetric flowfield develops within a wind tunnel. Experimentally acquired data frequently do not agree completely with simulated data on key measurements like lift and drag coefficients. The data incompatibility thus persists: experimental data will almost always lack the full-field flow data that can yield invaluable insight into why disparities exist between experimental and computational results.

Reduced Order Models (ROMs) [2] are one way to bridge the data structure divide between computational and physical experiments. ROMs by construction reduce the complexity of a simulated system while retaining dynamically significant coherent structures. They are often based on an empirical modal basis, which has been optimized for some objective function. The proper orthogonal decomposition (POD) [3] and dynamic mode decomposition (DMD) [4] are two such projection-based linear algorithms for basis construction that are commonly utilized in fluid mechanics. POD modes in particular have a rich history [5] of application for reduced-order modeling. In a linear, mean, normed sense, POD modes are the optimal basis for capturing the information in provided snapshots.

We focus on the problem of approximating flowfield snapshots using a very small set of flow volume data, such as that recorded within a wind tunnel using flow probes. While this flowfield approximation will be accomplished through a modal expansion, the basis itself will be constructed via a nonlinear optimization problem solved via machine learning rather than a more conventional linear optimization problem like the POD. Casting our problem into a regression task, where the model is defined partially as a function of available resolved scales, enables the underconstrained task of creating a complete flow snapshot from only a few data points to be a numerically solvable one. The challenge of employing a regressor to identify flow patterns without giving it to knowledge of the governing equations motivated a parallel structure to ROMs [6] where basis construction is decoupled from any application to analytical system models. Neural networks [7] have been widely used to produce ROMs from massive amounts of data acquired through experiments, sensor measurements, and large-scale simulations, based on the notion that data, even when integrated over surfaces or sampled very coarsely, is a manifestation of all underlying dynamics and processes. For example, San [8] suggested a single-layer feed-forward neural network for transient flows to provide accurate ROM coefficient predictions. Pawar [9] leveraged deep neural networks to circumvent the Galerkin projection stage and create a different form of ROM in which the evolution of POD modal amplitudes was predicted using residual and backward difference scheme formulas. In this work, we design a neural network based in a regression model for the dynamics of low-order states which identifies POD-like modes and coefficients that collectively reconstruct a mean flowfield snapshot from a very small amount of flowfield data. Nonlinear interaction between modes is treated similarly to Gao's implementation [10] instead of relying directly on the nonlinear terms in the flow's governing equations. In particular, a Shallow encoder network using limited sensor data inspired by Erichson [11] is designed and implemented.

The dimension of linear bases like classic POD approaches is known to grow exponentially [12] if a complex fluid flow must be recapitulated at all resolvent scales by a single basis. Similarly, as the system's complexity grows, a neural network's depth expands to learn the intricate nonlinear dynamics, and the number of trainable parameters rapidly grows. Deep neural networks have substantial epistemic uncertainty in the context of sparse data, which reduces the credibility of ROMs as remarked by Meidani [12]. To address this issue, we constrain the model's pattern recognition with required physical properties like mass conservation and accurate integrated values. In this way, the governing physics influence the identified coherent structures without being explicitly introduced into the model.

We first outline the model structure and the nature of the applied constraints in §II. We then apply the model to several test cases in §III where experimental data is simulated by sampling points of computational snapshots.

II. Methodology

Our objective is to estimate the flow field $x \in \mathbb{R}^m$ from sensor measurements $s \in \mathbb{R}^p$ by learning the relationship $x \mapsto x$ given $p \ll m$. The sensor measurements s are collected via a sampling process from the high-dimensional field x

$$s = H(x)$$

where $H : \mathbb{R}^m \to \mathbb{R}^p$ is the measurement operator. To reconstruct the flow, one must estimate the inverse model that produces the field x from the observation s

$$x = G(s)$$

where $G : \mathbb{R}^p \to \mathbb{R}^m$ is nonlinear forward operator. However, the measurement operator H may be unknown or highly nonlinear. As a result, the problem is frequently ill-posed, and the inversion ill-defined. In this work, a shallow neural network is trained to estimate the inverse function.

Standard strategies for estimating a state **x** from observations **s** will be described, as well as observability concerns. The principle behind established state reconstruction techniques is that a field **x** can be described in terms of a rank-k approximation, as described by Adler [13]

$$x \approx \hat{x} = \sum_{j=1}^{k} \phi_j \nu_j = \Phi \nu \tag{1}$$

where ϕ_j are the modes of the approximation and ν_j are coefficients. We may estimate the state x by learning the coefficients ν from the state observations, which typically gives us the minimum-energy solution, once we know the approximation modes Φ. Adler [13] also defines a fully connected neural network with K layers as a set of functions

$$\mathcal{F}(s; W) = R(W^k R(W^{k-1} \cdots R(W^1 s))) \tag{2}$$

where $R() : \mathbb{R} \to \mathbb{R}$ is a coordinatewise scalar nonlinear activation function and W is a set of weight matrices. We are provided a training set $\{x_i, s_j\}_{i=1}^n$ with n samples x_i and corresponding sensor measurements s_i. The network is created to learn a function $\mathcal{F} : s \mapsto \hat{x}$ which will minimize the error in Euclidean flow space over all sensor measurements.

$$\mathcal{F} \in \underset{F}{\operatorname{argmin}} \sum_{j=1}^{k} \|x_i - F(s_i)\|_2^2 \tag{3}$$

The nested nonlinear function for the Shallow encoder network is introduced by Erichson [11] in keeping with the POD methodology as follows:

$$\mathcal{F}(s) = \Omega(\nu(\phi(s))) \ . \tag{4}$$

The flow field estimate's interpretability is improved by modeling it on a small-scale basis, whose modes can be identified with the field's spatially coherent structures. This means that the estimate is a linear combination of k modes ϕ_j, weighted by coefficients ν_j as developed originally by Aubry [14]. These modes are determined by the inputs. Thus, following Erichson [11], we consider a network in which the output $\hat{x}$ is given by a linear functional, last layer of k inputs, interpreted as ν. These coefficients are nonlinearly guided by the sensor measurements s. This network allows both modes and coefficients to be learned simultaneously similar to Gao [10] network. A hidden layer captures the nonlinear map $s \mapsto \nu$, and its outputs ψ are measurement features that characterize nonlinear combinations of the input measurements s. The nonlinear combinations are then projected to the coefficients ν associated with the modes ϕ. While the discrete flow field x determines the size of the output layer, the size of the last hidden layer ν determines the size k of the dictionary Φ. The size is then calculated using dimensionality estimation methods [15] during network fine tuning. Following Erichson [11] closely, we model the Shallow encoder as a neural network with two hidden layers ψ and ν, followed by a linear output layer Ω. There are two types of hidden layers to consider: fully connected (FC) and convolution layers. The success of modern deep learning architectures in computer vision is due to the effectiveness of convolution layers. Xu [16] introduced a data-driven framework that included a convolutional encoder for identifying nonlinear basis functions, a temporal convolutional encoder for learning the temporal dynamics of latent variables, and a fully connected neural network for learning the mapping between system parameters and latent variables. He showed how well this approach predicted problems involving discontinuities, wave propagation, and coherent structures. Since convolution layers demand a significant number of instances and computational power for training due to the lack of spatial ordering in sensor readings, and because potential dynamical systems evolve on a curved domain, which

is commonly represented using an unstructured grid, Erichson [11] chose fully connected layers. Thus, the first and second hidden layers take the form

$$z^{\psi} = \psi(s) = R(W^{\psi}s + b^{\psi}) \tag{5}$$

and

$$z^{\nu} = \nu(z^{\psi}) = R(W^{\nu}z^{\psi} + b^{\nu}) \ . \tag{6}$$

The final linear output layer is then

$$\hat{x} = \Omega(z^{\nu}) = \Phi z^{\nu} + b^{\Phi} \tag{7}$$

where the columns of the weight matrix Φ are the dominant modes. So the architecture of the Erichson's [11] Shallow encoder becomes

$$s \mapsto \psi(s) \mapsto \nu(z^{\psi}) \mapsto \Omega(z^{\nu}) = \hat{x}$$

and graphically, the problem can be represented in Figure 1.

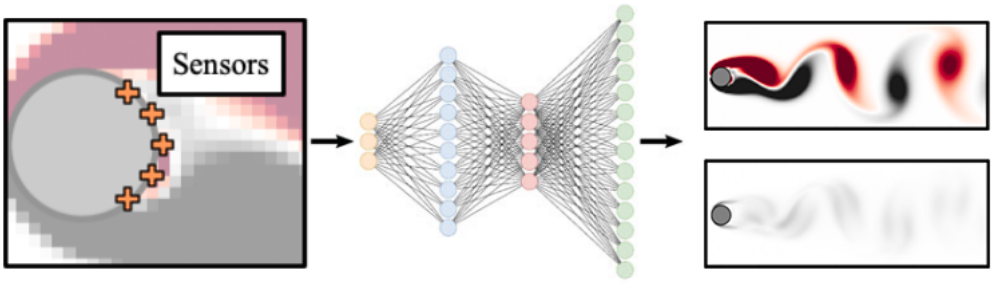

Fig. 1 Graphical representation of the network structure. Central image taken from Ref. [11].

The size of each layer must be adjusted depending on the dataset. By limiting the dimension of the space in which the estimation is performed, the problem is effectively regularized by filtering out most of the noise that does not exist in a low-dimensional space.

A. Regularization

Overfitting happens when a function fits a small collection of data points too accurately during training, which prevents it from generalizing well on a test dataset. Thus, to learn a function that generalizes to new observations that have not been used for training, further constraints are required. Early stopping criteria and weight penalties to regularize the function's complexity are common tactics for avoiding overfitting. To boost convergence and robustness, the designer of the shallow encoder [11] added batch normalization (BN) [17] and dropout layers (DL) [18]. Batch normalization compensates for the variation in output signal distribution across minibatches during training thus diminishing the effect of internal covariate shifts [17]. Dropout layers strengthen the network's generalization quality: during the training step, a small percentage of neurons are turned off at random. Thus, Shallow encoder architecture becomes

$$s \mapsto \psi(s) \mapsto BN \mapsto DL \mapsto \nu(z^{\psi}) \mapsto BN \mapsto \Omega(z^{\nu}) = \hat{x} \ .$$

Our final improvement is application of Ridge regularization [19] to reduce the variance of the estimator, which can be stated as follows. We are provided a training set $\{x_i, s_j\}_{i=1}^{n}$ with n samples x_i and corresponding sensor measurements s_i, the network is created to learn a function $\mathcal{F} : s \mapsto \hat{x}$ which will minimize the error in Euclidean flow space over all sensor measurements.

$$\mathcal{F} \in \underset{F}{\operatorname{argmin}} \sum_{j=1}^{k} \|x_i - F(s_i)\|_2^2 + \lambda \|W^i\|_2^2 \tag{8}$$

The Ridge regression term introduces L^2 regularization to the weight matrices with a parameter λ which was was found to be 0.03 during fine tuning of the network.

To train the shallow decoder, we employ the ADAM [20] optimization technique, with learning rate 10^{-2} and weight decay 10^{-4}. The learning rate determines how much the weights are adjusted in each epoch and weight decay regularizes network complexity. Experimentally, we found that reducing the learning rate (by 0.7 for each 100 epochs) improves the performance. These improvements to ADAM performed much better during our testing then Stochastic Gradient Descent (SGD) with momentum optimizer [21].

B. Physics-based Constraints

Many applications, such as turbulence modeling [22] and generative modeling of dynamical systems, have seen success with physics-informed data-driven modeling [23]. To prevent model overcomplication, we only introduce incompressible mass conservation and integrated force and moment constraints.

In order to ensure that the resulting flowfield reconstruction satisfies at least incompressible mass conservation, we introduce a discrete divergence operator L_{div} , which is given by a $Nx2N$ matrix associated with a finite difference scheme, and

$$(L_{div}x)_k \approx (\nabla \cdot w)(p_k) = 0 \tag{9}$$

where $w = (u, v) \in \mathbb{R}^2$ where u and v are the horizontal and vertical velocities, respectively, and $\{p_1, \cdots, p_N\}$ consist of N grid points p_n. To implement this as an extra regularizing term, we first create a custom loss function where we calculate the gradients of the velocity then with the gradients we can then calculate the divergence.

The net force and moment on the body of interest is computed by integrating the velocity stress tensor on the body's surface. We computed the velocity gradients previously for the incompressibility constraint. Using the Newtonian fluid assumption we can calculate the stress tensor and, by selecting all body-surface grid points, we can approximate the surface integral via a Riemann sum and obtain the model's calculated body force and moment components. By then constraining the model such that this surface integral equals the measured values, the network will be more reliable even if the smaller flowfield structures are not perfectly resolved.

C. Network Structure Optimization

The shallow encoder's loss function must be carefully refined to achieve resilience to noise and data sparsity. Here, we provide the mathematical framework of the loss function modification which ensures a matching of integrated forces on the boundary of the airfoil. Previously, the shallow encoder identified flow features with the first hidden layer and and then extracted the dominant modes and a dictionary of coefficients with the second layer. We cast the problem more generally as learning an unknown variable $q(x, t) \in \mathbb{R}$ defined on spatial domain Ω and on a time interval $[t_0, t_f]$ which satisfies

$$N(q, t) = f(x, t), \forall x, t \in \Omega \times [t_0, t_f] \tag{10}$$

$$q(x, t) = h(x, t), \forall x, t \in \Omega \times [t_0, t_f] \tag{11}$$

$$q(t, t_0) = q_0(x), \forall x \in \Omega \tag{12}$$

where N is a nonlinear differential operator with respect to spatio-temporal coordinates. The shallow encoder will then approximate the true $q(x, t)$ with $\tilde{q}(x, t) = SD(\theta, x, t)$ where the shallow decoder (SD) is defined by a set of trained parameters $\theta \in \mathbb{R}^p$. A conventional SD optimizes the set of parameters θ to minimize the loss function $\mathcal{L}_s$ – the average squared distance to specific measured virtual sensor values q_m, which is defined on a sample of coordinates V_m of size N_m

$$\mathcal{L}_s = \frac{1}{N_m} \sum_{x_m, t_m \in V_m} |\tilde{q}(x_m, t_m) - q_m|^2 \tag{13}$$

In this implementation, we also introduce a residual term which is based on differential nonlinear operator N

$$\mathcal{L}_r = \frac{1}{N_{in}} \sum_{x_{in}, t_{in} \in V_{in}} |N(\tilde{q}(x_{in}, t_{in})) - f(q_{in}, t_{in})|^2 \tag{14}$$

where V_{in} is a sampling of the PDE domain $\Omega \times [t_0, t_f]$. For example, if Neumann boundary conditions are used then V_{in} is a sampling of $\partial\Omega \times [t_0, t_f]$ and N and f are adapted by the network. (Automatic differentiation is given to us "for free" by `pytorch`.) Our modified loss function thus becomes

$$\mathcal{L} = \mathcal{L}_s + \mathcal{L}_r \tag{15}$$

The model's parameters θ are optimized once the loss function $\mathcal{L}$ and sampling spaces V_m and V_{in} are established so that the approximate solution q best matches both equations and data.

To add externally measured integrated forces and moments into the network, we must carefully consider boundary conditions (no slip on the cylinder). Penalizing the error on a sample of points is one technique to take the boundary conditions into consideration. Peng [24] introduced the prior-dictionary method, the goal of which is to force the

approximated solution's shape to comply with certain requirements, particularly the Dirichlet conditions. If the condition to be matched

$$q(x,t) = h(x,t), \forall x \in \partial\Omega$$

without time dependency Peng [24] suggests to define the approximated solution as

$$\tilde{q}(x,t) = SD(\theta,x,t) \times f_\partial(x)h(x,t), \forall x,t \in \Omega \times [t_0,t_f] \tag{16}$$

where $f_\partial(x) = 0$ on $\partial\Omega$. For example, according to Peng [24] for canonical 2D Cylinder flow where $q = (u,v)$ and $x = (x,y)$ a no-slip boundary condition on $y = 0$ will be satisfied by a choice of $f_\partial(x) = tanh(\gamma(r - r_c))$, where $r^2 = (x-x_c)^2 + (y-y_c)^2$ with $(x_c,y_c) = (0,0)$ and $r_c = \frac{1}{2}$. The parameter γ defines the slope of $f_\partial(x)$ at the boundary and in below image we fine tuned $\gamma = 5$ for it's finite gradients and very brief transition zone. Figure 2 presents this enforcement in the formulation.

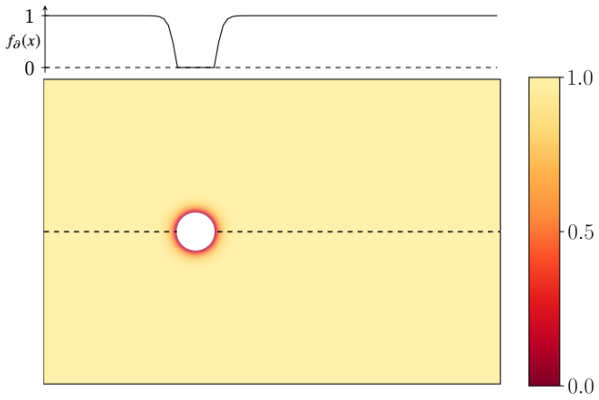

Fig. 2 Prior-dictionary $f_\partial(x)$ that enforces cylinder no-slip boundary condition.

However, to minimize the deformation of the neural network output $SD(\theta,x,t)$, it is challenging to choose a function $f_\partial(x)$ that is nearly flat on the entire domain except at particular borders.

To compute the integrated force on the border of the body, we again use a parametric equation for the surface defined by $(\lambda,\xi) \in [0,1]^2 \rightarrow (x_\partial, y_\partial, z_\partial) \in \mathbb{R}^3$. We can then directly calculate the normal vector to the boundary surface. More concretely, in 2D:

$$\vec{n}(\lambda) = (n_x(\lambda), n_y(\lambda)) = \Big(-\frac{\partial y_\partial}{\partial\lambda}(\lambda), \frac{\partial x_\partial}{\partial\lambda}(\lambda)\Big) \tag{17}$$

Then total integrated forces on the surface can be calculated by averaging the Monte Carlo simulations to integrate local forces $\vec{df}(x,y)$

$$\vec{F} = \int_{\partial\Omega_f} \vec{df}\, dl$$

For incompressible flow, local forces are calculated as Peng [24] directly from momentum conservation:

$$df_x = -pn_x + \frac{2}{Re}\frac{\partial u}{\partial x}n_x + \frac{1}{Re}\Big(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\Big)n_y \tag{18}$$

$$df_y = -pn_y + \frac{2}{Re}\frac{\partial v}{\partial y}n_y + \frac{1}{Re}\Big(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\Big)n_x \tag{19}$$

Finally, the integrated force is approximated via Monte-Carlo

$$\vec{F}(t) = \int_{\partial\Omega_f} \vec{df}(x,y,t)dl = \int_{[0,1]} \vec{df}(x_\partial(\lambda), y_\partial(\lambda), t)\Big|\frac{dl(\lambda)}{d\lambda}\Big|d\lambda \approx \frac{1}{|V_\lambda|}\sum_{\lambda\in V_\lambda} \vec{df}(x_\partial(\lambda), y_\partial(\lambda), t)\Big|\frac{dl(\lambda)}{d\lambda}\Big| \tag{20}$$

where V_λ is sampling of $[0,1]$ after which is mapped to the body's boundary by $\lambda \rightarrow x_\partial(\lambda), y_\partial(\lambda)$. We then employ adaptive sampling with Monte-Carlo [25] to compute $\Big|\frac{dl(\lambda)}{d\lambda}\Big|$ using the probability distribution of random variable λ.

It is now possible to compute the residual term 14 by randomly sampling points V_{in} by evenly spreading 80% of the points and focusing the remaining 20% within a predetermined radius around the 2D cylinder as instructed in [25] and illustrated in Figure 3. This point distribution increases the relative weight of the residuals in the cylinder's boundary layer relative to those in the remaining fluid domain.

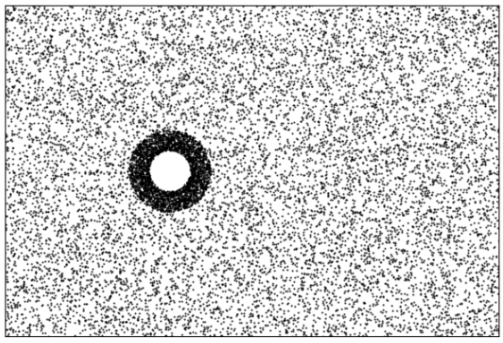

Fig. 3 Two zones sampling.

D. Root Mean-Square Modal Amplitudes and Bounds of Applicability

With this modified loss function, we use a root mean-square (RMS) methodology to rank the eigenmodes Φ_{nn}. The shallow decoder consists of three layers, with the last layer lacking any nonlinear activation; thus, the linear superposition of the eigenmodes is preserved. The first two layers are used to train the coefficient vector v and the last layer is used to train the eigenmodes Φ_{nn}.

We normalize each column of Φ_{nn} by dividing the column by its norm and correspondingly multiply the row of v_{nn} by that column norm. This normalization does not change the predicted value $\hat{f}$, but will properly define the eigenvectors to be of unit magnitude. We then compute the coefficient matrix V_{nn} formed by stacking the predicted values obtained by running the first two steps of the trained model for all data. Root mean-square modal amplitudes are then obtained by summing each row of V_{nn}. Finally, the RMS amplitudes are used to rank-order the eigenmodes. For the canonical 2D cylinder, the RMS model spectrum of the trained network is displayed in Figure 4.

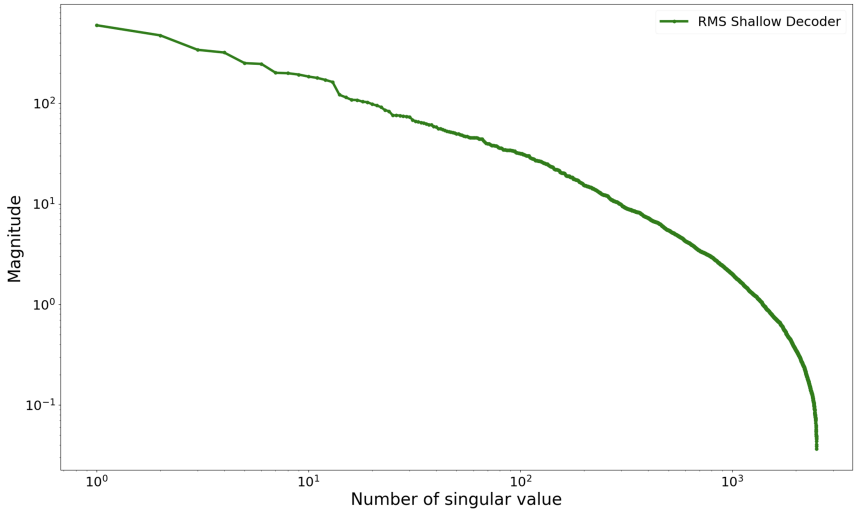

Fig. 4 RMS Modal Spectrum.

We now can compare the applicability (quantified by the captured flow spectrum) of our method against the true spectrum, the reconstruction by a standard POD approach, and a more conventional shallow encoder. This comparison is presented in Figure 5. This new method captures a more complete spectrum than do the other approaches.

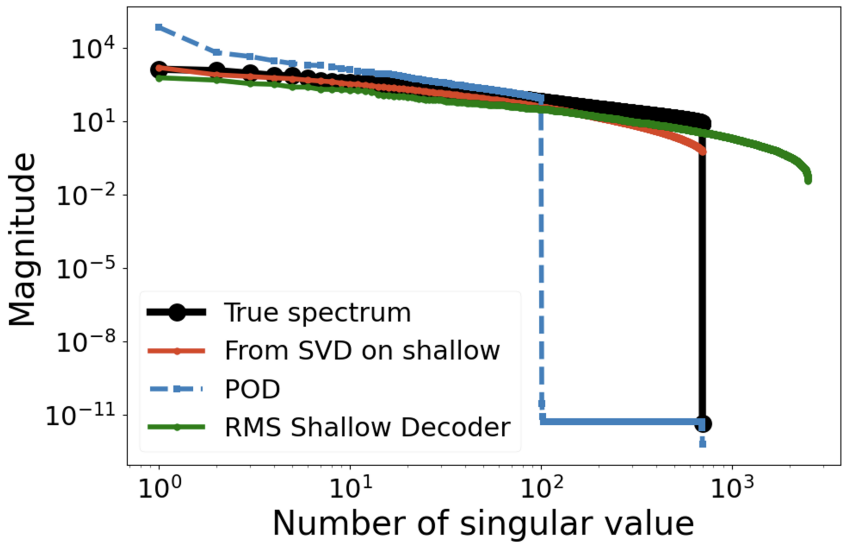

Fig. 5 All modes spectrum.

E. Test Case: Flow Over a Circular Cylinder

Flow over a circular cylinder was generated using using an Immersed Boundary Projection Method (IBPM) [26] at critical and supercritical Reynolds numbers (Re=100, 500, 1,000, 5,000, 10,000, 20,000) such that the flowfield exhibited both oscillatory and chaotic dynamics. Two-dimension grid resolution was set to 2,000 points parallel to the freestream and 500 points normal to the freestream. A time step was set to keep the CFL condition close to unity. To mimic a wind tunnel experiment, a discrete number of points within the flow volume were sampled – with a majority of them residing on the cylinder surface – to provide the samples necessary for network training. A lower-resolution grid of 200-by-370 points near the cylinder was sampled to train the network, rather than the full 2,000- by 500-point grid, for computational speed reasons. Flow within this sampled window is visualized in the top of Figure 6.

F. Test Case: Lid-Driven Cavity

We also applied the procedure to a canonical 2D fluid flow, the regularized lid-driven cavity, with datasets previously published by the authors [27]. Data was sampled at Reynolds numbers ranging from 15,000 to 30,000, within which range the flow progressed from criticality through supercriticality to fully developed chaos.

III. Results and Discussion

A. Flow Over a Circular Cylinder

The network was trained on 700 flow snapshots and tested on 300 additional snapshots to measure predictive fidelity. When data from only 100 discrete points was provided to the network, 2000 neurons in the first hidden layer and 2500 neurons in the second were able to produce a flow reconstruction, which agreed at the large scales with the correct dynamics. The mean training error was computed to be 5.28% and the mean test error was 16.20%, indicating that the model was not over-trained. Figure 6 presents a contour of the reconstructed flow snapshot and a contour of the difference between this reconstruction and the truth. It is clear that only smaller-scale fluid structures are not captured by the network.

Another method of quantifying the quality of the flow reconstruction is by comparing the true global eigenspectrum to that captured by the reconstructed snapshot. An exact overlap of these two lines would indicate a perfect flow reconstruction. Figure 7 demonstrates the high quality of the spectrum reconstruction by the neural network. For comparison, a linear flow reconstruction by the standard POD is also plotted and is shown to perform much more poorly than this neural network.

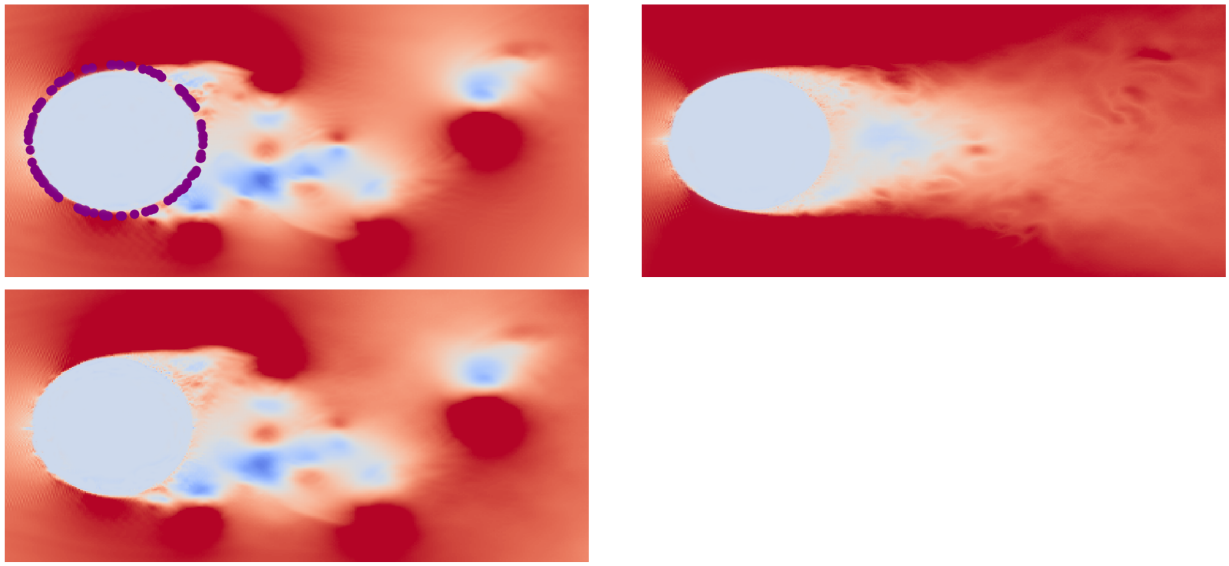

Fig. 6 Reference flow snapshot (horizontal velocity) with sensor locations (top left), reconstructed flow snapshot (bottom), and pointwise reconstruction error with white being higher error (right).

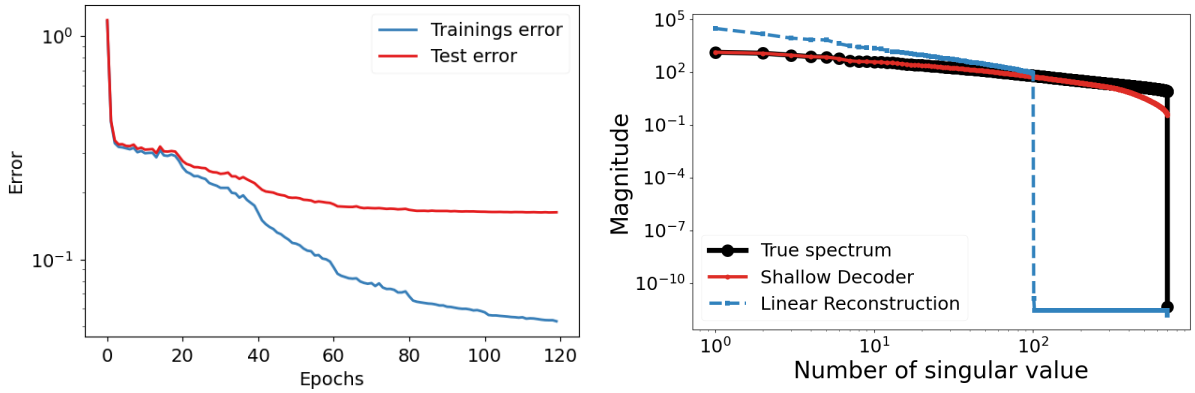

Fig. 7 Train and test error of the neural network as a function of epoch (left) and a comparison of the flowfield eigenspectra (right) for the circular cylinder.

B. Lid-Driven Cavity

The same network was trained on the lid-driven cavity data with 100 sensors placed randomly within the flow domain. The network was found to perform best with 200 neurons in the first layer and 950 neurons in the second. The mean training error was computed to be 8.0% and the mean test error was 7.8%. Contours of the true and reconstructed flowfield at two Reynolds numbers are presented in Figure 8, as well as the sampled point locations. The reconstructed flowfield is again shown to be qualitatively identical to the reference flowfield. The cavity spectrum decomposition, plotted in Figure 9 for the $Re = 30,000$ case (with similar behavior at the lower Reynolds number), likewise demonstrates the quality of the flowfield reconstruction.

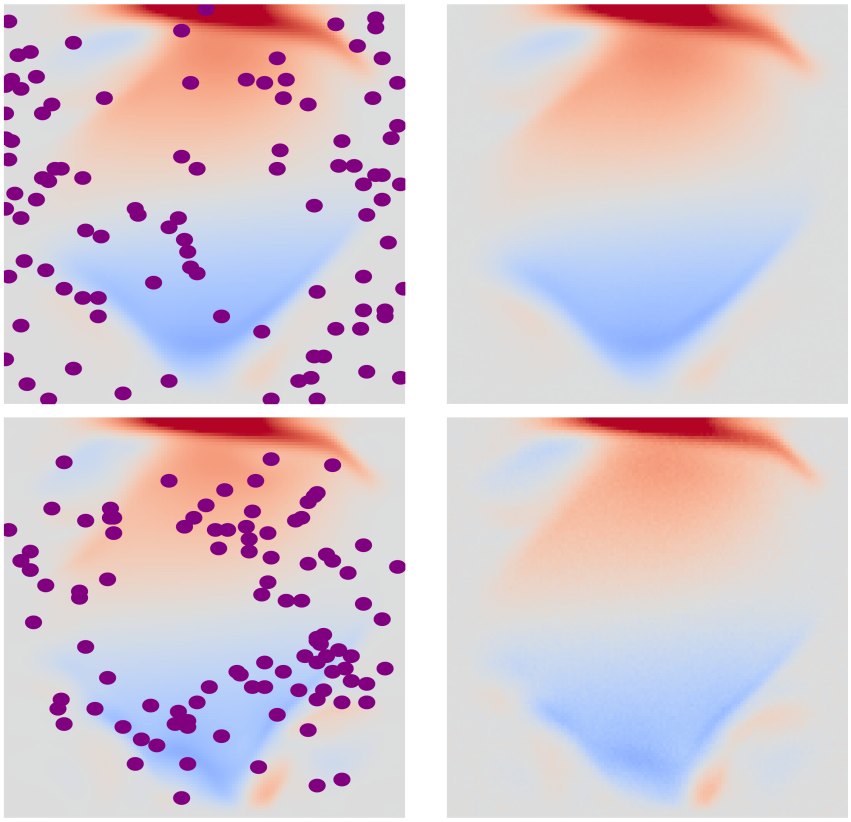

Fig. 8 Horizontal velocity contours at $Re = 15500$ (top) and $Re = 30000$ (bottom) for both the true flow data (left) and the reconstructed flow data (right). Sensor locations are also plotted on the true contours.

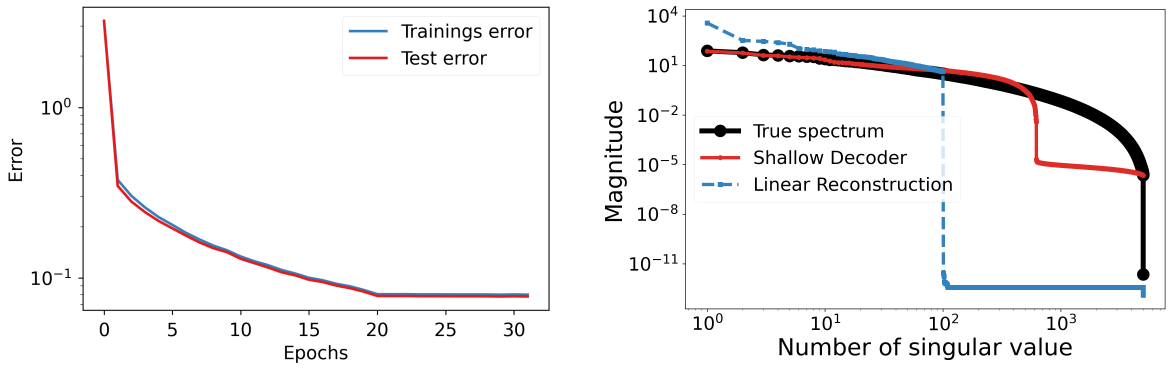

Fig. 9 Train and test error of the neural network as a function of epoch (left) and a comparison of the flowfield eigenspectra (right) for the cavity flow at $Re = 30000$.

IV. Conclusion

A physics-constrained shallow neural network was developed, which can accurately reconstruct flowfield snapshots from discrete velocity samples within the control volume. The network can constrain the resulting predictions to agree with integrated forces and moments which may be available from other experimental data collection techniques. This network was tested on two canonical two-dimensional flow regimes: flow over a supercritical cylinder and chaotic flow within a lid-driven cavity.

This tool has immediate applicability for data fusion between experimental data sources – where spatially sparse, but highly trustworthy, data are available – and computational data sources – where spatially rich, but often less reliable, data are available. Such flowfield reconstructions also enable the real-time visualization of large-scale flow features within an experiment facility without the need for standalone flow visualization technologies to be integrated into the test framework.

Acknowledgments

This work was authored by employees of the University of Florida under Contract No. 80LARC21A005 with the National Aeronautics and Space Administration. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for United States Government purposes. All other rights are reserved by the copyright owner.

References

[1] Roozeboom, N., and Baerny, J. K., "Customer Guide to Pressure-Sensitive Paint Testing at NASA Ames Unitary Plan Wind Tunnels," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 1055.

[2] D. J. Lucia, P. S. B., and Silva, W. A., "Reduced-order modeling: New approaches for computational physics," *Prog. Aerosp. Sci*, Vol. 40, 2004, pp. 51–117.

[3] P. Holmes, J. L. L., and Berkooz, G., *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, New York, 1998.

[4] J. N. Kutz, B. W. B., S. L. Brunton, and Proctor, J. L., "Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems," *SIAM*, 2016.

[5] Holmes, P., Lumley, J. L., Berkooz, G., and Rowley, C. W., *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge university press, 2012.

[6] S. Fresca, L. D., and Manzoni, A., "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parame- trized pdes," *J. Sci. Comput.*, Vol. 87, 2021, pp. 1–36.

[7] B.C.Csaji, "Approximation with artificial neural networks," Master's thesis, Eotvos Lorand University, 2001.

[8] O. San, R. M., and Ahmed, M., "An artificial neural network framework for reduced order modeling of transient flows," *Commun. Nonlinear Sci. Numer. Simul.*, Vol. 77, 2019, pp. 271–287.

[9] S. Pawar, H. V., S. Rahman, and Vedula, P., "A deep learning enabler for nonintrusive reduced order modeling of fluid flows," *Phys. Fluids*, Vol. 31, 2019.

[10] H. Gao, J.-X. W., and Zahr, M. J., "Non-intrusive model reduction of largescale, nonlinear dynamical systems using deep learning," *Physica D*, Vol. 412, 2020.

[11] Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., and Kutz, J. N., "Shallow neural networks for fluid flow reconstruction with limited sensors," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 476, No. 2238, 2020, p. 20200097.

[12] Meidani, K., and Farimani, A. B., "Data-driven identification of 2D partial differential equations using extracted physical features," *Comput. Methods Appl. Mech. Eng*, Vol. 381, 2021.

[13] Adler J, O. O., "Solving ill-posed inverse problems using iterative deep neural networks," *Inverse Probl*, Vol. 33, 2017.

[14] Aubry, N., "On the hidden beauty of the proper orthogonal decomposition," *Theor. Comput. Fluid Dyn.*, Vol. 2, 1991, p. 339–352.

[15] Granata D, C. V., "Accurate estimation of the intrinsic dimension using graph distances: unraveling the geometric complexity of datasets." *Sci. Rep*, Vol. 6, 2016.

[16] Xu, J., and Duraisamy, K., "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics," *Comput. Methods Appl. Mech. Eng.*, Vol. 372, 2020.

[17] Ioffe S, S. C., "Batch normalization: accelerating deep network training by reducing internal covariate shift," *arxiv*, 2015.

[18] Srivastava N, K. A. S. I. S., Hinton G, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res*, Vol. 15, 2014, pp. 1929–1958.

[19] Hoerl, A. E., and Kennard, R. W., "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, Vol. 42, No. 1, 2000, p. 80–86.

[20] Kingma DP, B. J., "Adam: a method for stochastic optimization," *arxiv*, 2014.

[21] Sutskever I, D. G. H. G., Martens J, "On the importance of initialization and momentum in deep learning," *Proc. of the 30th Int. Conf. on Machine Learning*, Vol. 28, 2013, pp. 1139–1147.

[22] J. Ling, A. K., and Templeton, J., "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, Vol. 807, 2016, pp. 155–166.

[23] N. B. Erichson, M. M., and Mahoney, M. W., "Physics-informed autoencoders for Lyapunov-stable fluid flow prediction," *arxiv*, 2019.

[24] Peng, W., Zhou, W., Zhang, J., and Yao, W., "Accelerating Physics-Informed Neural Network Training with Prior Dictionaries," *arXiv*, 2020.

[25] Oh, M.-S., and Berger, J. O., "Adaptive importance sampling in monte carlo integration," *Journal of Statistical Computation and Simulation*, Vol. 41, No. 3-4, 1992, pp. 143–168. https://doi.org/10.1080/00949659208810398, URL https://doi.org/10.1080/00949659208810398.

[26] Taira, K., and Colonius., T., "The immersed boundary method: A projection approach." *J. Comput. Phys.*, Vol. 225, 2007, pp. 2118–2137.

[27] Lee, M., "Steady State 2D Regularized Lid-Driven Cavity Low-resolution Spatiotemporal Snapshots," Mendeley Data, 2021.