



Article

Tracking Turbulent Coherent Structures by Means of Neural Networks

Jose J. Aguilar-Fuertes [†], Francisco Noguero-Rodríguez [†], José C. Jaen Ruiz [†] and Luis M. García-RAffi and Sergio Hoyas ^{*}

Instituto Universitario de Matemática Pura y Aplicada, Universitat Politècnica de València, 46022 València, Spain; joagfue@etsid.upv.es (J.J.A.-F.); franorod@etsid.upv.es (F.N.-R.); jojaerui@etsid.upv.es (J.C.J.R.); lmgarcia@mat.upv.es (L.M.G.-R.)

* Correspondence: serhocal@mot.upv.es

† These authors contributed equally to this work.

Abstract: The behaviours of individual flow structures have become a relevant matter of study in turbulent flows as the computational power to allow their study feasible has become available. Especially, high instantaneous Reynolds Stress events have been found to dominate the behaviour of the logarithmic layer. In this work, we present a viability study where two machine learning solutions are proposed to reduce the computational cost of tracking such structures in large domains. The first one is a Multi-Layer Perceptron. The second one uses Long Short-Term Memory (LSTM). Both of the methods are developed with the objective of taking the the structures' geometrical features as inputs from which to predict the structures' geometrical features in future time steps. Some of the tested Multi-Layer Perceptron architectures proved to perform better and achieve higher accuracy than the LSTM architectures tested, providing lower errors on the predictions and achieving higher accuracy in relating the structures in the consecutive time steps.

Keywords: turbulence; turbulent structures; DNS; machine learning; neural networks



Citation: Aguilar-Fuertes, J.J.; Noguero-Rodríguez, F.; Jaen Ruiz, J.C.; García-RAffi, L.M.; Hoyas, S. Tracking Turbulent Coherent Structures by Means of Neural Networks. *Energies* **2021**, *14*, 984. <https://doi.org/10.3390/en14040984>

Academic Editor: Dimitris Drikakis
Received: 18 January 2021
Accepted: 9 February 2021
Published: 13 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In this paper, the terms “turbulent structures”, “Qs” and “sweeps, and ejections” are used interchangeably.

Turbulence is probably the open problem in physics with most applications in daily life. Wall-bounded turbulence is present in many aspects of our life, and it is responsible for up to 5% of the CO₂ dumped by humanity every year [1]. To completely understand turbulence is perhaps a too-ambitious problem, which we still are years apart, so we should focus on improving models such that they can simulate, in a reliable and fast way, the behaviour of flows. Because we want to understand these flows first, no modelling can be used, so we have to use the Direct Numerical Simulation (DNS) technique. This technique does not use any other modelling than the Navier–Stokes equations. Moreover, DNS are restricted to simplified geometries. The most successful of these idealized flows are Poiseuille turbulent channels, where the fluid is confined between two parallel plates and the flow is driven by pressure. However, the computational cost is extremely high. For doing this, supercomputers are and will be the right technological tool. Since the 90's of the last century, their power have increased in an exponential way, and the DNS have grown accordingly. Because the seminal work of Kim, Moin, and Moser [2], the main control parameter, the friction Reynolds number Re_τ has increased continuously [2–8].

In order to understand and model turbulence, in the last 100 years, the main mechanism has been the Reynolds decomposition, as well as to improve scaling laws. However, in the last years, we have seen a drift to structures. Since the seminal work of Chong [9] and others, we have been able to identify the basic coherent structures of turbulent flows. However, the interaction of these structures is a highly nonlinear problem. Their behaviour or

even the precedence (cause-effect) [10] of some of these structures is still an open problem. The idea of this work is to create a code able to identify and follow the different structures of the flow as the flow is simulated. This avoids the necessity of storing large databases.

Traditionally, the structure-following problem has been focused on *hairpins*: groups of U-shaped dissipative structures that are formed near the wall and go towards the outer region. Head and Bandyopadhyay [11] identified groups of structures at 45 degrees from the wall, producing a model that Adrian et al. [12] and Del Álamo et al. [13] evolved towards the vortex clusters model, identifying *hairpin clusters* whose lifetime was longer than their characteristic one. With the developments in computational power that have come in the last decade, the Reynolds number of large DNS simulations has risen, and the existence of these *hairpins* in large Reynolds number flows has been questioned by authors, such as Jiménez et al. [14].

Instead of looking for hairpins, the idea is to use the momentum transfer model of *sweeps* and *ejections*, high instantaneous Reynolds Stress events in the flow where the momentum transfer is large. Lozano-Durán et al. [15] found that these events are representative for the momentum transfer in the logarithmic layer, and tracked them through their lifetime to find that some of them become large attached to the wall and extend across the logarithmic layer, and they are responsible for most of the total momentum transfer [10]. Figure 1 contains a representation of a sweep-ejection pair extracted from a DNS simulation.

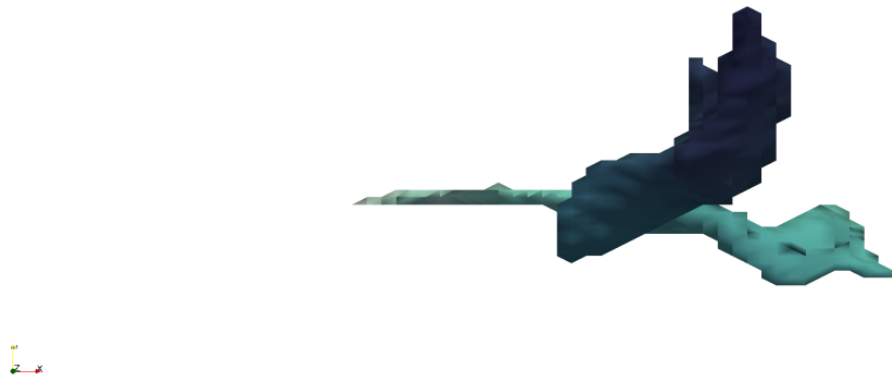


Figure 1. Representation of a sweep-ejection pair from a single DNS time step.

However, a big problem appears. Typically, one can expect millions of structures, like these ones per step. Because the time step is around 30s in the largest simulations of turbulence, the code has to be able to identify the vortexes and relate them with the ones of the previous step in this brief time. Machine Learning (ML) has been the strategy followed here. This technique has entered most scientific areas for a vast set of data processing tasks. Under this term, ML, we can find a broad range of algorithms and modelling tools, which includes the Artificial Neural Networks (ANNs). It was proved in the decade of 80's in the last century that ANNs are universal approximators [16,17]. ANNs can be used for fitting, classification, clustering, or time series among other classical problems in Applied Mathematics. They are able to recognise patterns in data and, in some sense, allow us to “mathematize the phenomenology”. Despite their bad reputation as “black boxes”, ANNs are a good mathematical solution for the approximation of functions when no explicit expression can be obtained by other techniques. The use of neural networks for turbulence research is developing rapidly in recent years. The need of DNS methods to produce complete information in the whole domain for solving each step grants an abundance of available training data; and, the problems associated with fluids, such as the difficulty to measure precisely experimental data or the high computational cost of simulations, are challenges that are similar to the ones solved with neural networks in other fields. The work of Srinivasan et al. in predicting boundary layer behaviour by means of convolutional and recurrent neural networks [18], the work of Park and Choi in

reconstructing data for flow control by means of convolutional neural networks [19], the work of Ling et al. in using ANNs to estimate the Reynolds Stress anisotropy tensor to augment the results provided by Reynolds Average Navier Stokes (RANS) models [20], and other ANN approaches to modelling the closure problem (modelling the time-averaged value of the Reynolds Stress tensor) [21,22] are good examples of this.

ANNs have also been used to obtain successful results in other fields of fluid dynamics where DNS data are not available, while using the experimental results as training [23–25]. Solutions to the closure problem that use ANN to predict the Reynolds Stress Tensor in fluid fields have also been proposed using experimental data as training, such as Singh et al., while using ANN augment the Spallart–Almaras turbulence model [26]. Brunton et al. [27] and Duraisamy et al. [28] have made detailed reviews of the state of the art of Machine Learning in Turbulence.

In this article, we perform a viability study for a novel method to identify and track these structures with the aid of neural networks. The goal is to understand the particular life of every structure and identify the most energetic structures of the flow in an efficient way.

Section 2 describes the computational materials and methods used for this work, where Section 2.1 describes the DNS numerical model that has been used to obtain the fluid fields, Section 2.2 describes the process used for fluid structure and Sections 2.3 and 2.4 detail the neural network solutions to the problem. Section 3 presents the results of our analysis. Section 4 shows the conclusions.

2. Materials and Methods

2.1. DNS Numerical Model

In this work, we are going to simulate a Poiseuille (pressure-driven flow) plane channel flow. Figure 2 contains a schematic description of the channel used for the DNS simulation and the reference system used. The streamwise, wall-normal, and spanwise coordinates are respectively denoted by x , y , and z , and the corresponding velocity components are U , V , and W , or using index notation, U_i , for $i = 1, 2, 3$. Statistically averaged quantities are denoted by an overbar, whereas fluctuating quantities are denoted by lowercase letters, i.e., $U = \overline{U} + u = \overline{U}_1 + u_1$. The governing equations of the system are the incompressible Navier–Stokes equations,

$$\partial_j U_j = 0, \quad (1)$$

$$\partial_t U_i + U_j \partial_j U_i = -\partial_i P + \frac{1}{\text{Re}_\sigma} \partial_{jj} U_i. \quad (2)$$

The LISO code has been used to solve system (1) and (2). These code has been validated in many occasions, and it has been used to run some of the largest problems in turbulence. These equations have been solved using the LISO code, which has successfully been employed in order to run some of the largest simulations of turbulence [3,29,30]. The code is an evolution of the classic one of [2], but using five-point compact finite differences in y direction, with fourth-order consistency and extended spectral-like resolution [31]. A Runge–Kutta scheme, third-order semi-implicit, specially developed for this kind of flows [32] has been used for the temporal discretization. The wall-normal grid spacing is adjusted in order to keep the resolution approximately constant in terms of the local isotropic Kolmogorov scale $\eta = (v^3/\epsilon)^{1/4}$ at $\Delta y = 1.5\eta$, i.e. In this formula, ϵ is the dissipation rate. In wall units, Δy^+ varies from 0.3 at the wall, up to $\Delta y^+ \simeq 12$ at the centerline. Table 1 provides details of the simulation.

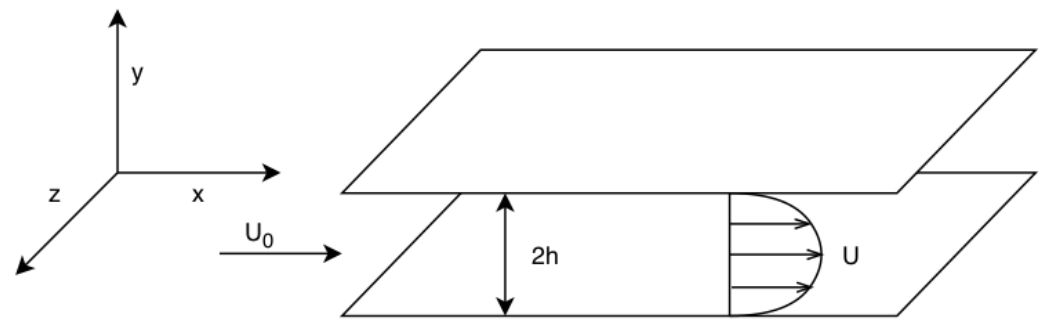


Figure 2. Schematic representation of the turbulent channel used in the simulations, including a description of the reference frame.

Table 1. Parameters of the flow, the solver, and the domain of the Direct Numerical Simulation (DNS) simulation, in which this work has been developed.

Flow Parameters	
Reynolds number Re	11,480
Wall Reynolds number Re_τ	500
Frition velocity u_τ	0.0487
Solver Parameters	
CFL	0.9
Time step Δt	0.0295
Geometrical Parameters	
Number of points in the x direction	1536
Number of points in the y direction	251
Number of points in the z direction	1152
Domain length in x	$8\pi h$
Domain length in z	$3\pi h$
Channel height in y	$2h$

In this simulation, high Reynolds Stress events are recognised by means of quadrant analysis, with the approach that was proposed by Wallace et al. [33]. This approach consists of studying turbulent regions with high instantaneous Reynolds Stress in the different quadrants, depending on the signs of u and v . The structures with higher instantaneous u and v are present in Q2 (ejections, $u < 0, v > 0$) and Q4 (sweeps, $u > 0, v < 0$).

According to Lozano-Durán and Jimenez [10], these structures are not only responsible for the momentum transfer inside the logarithmic layer, but they are also the primitive structures that originate the *hairpin* vortex clusters, having a key role in the drag generation in turbulent flows.

2.2. Tracking Sweeps and Ejections in Turbulent Flows

Sweeps and ejections are coherent regions of the flow, where the instantaneous point-wise Reynolds Stress is bigger than a threshold, established on the basis of percolation. Percolation theory is the study of the connected components on random graphs, and it was first applied to turbulent structure identification by Moisy and Jimenez [34]. If the limit is too permissive, a sponge-like structure forms, spanning all of the domain, but, if it is too restrictive, only very small structures will form. Between those, there is a value of the limit where the ratio between the volume of the largest structure and the total volume of all the detected structures raises sharply. This is the point selected for the value of the criteria. This limit was set to a factor of the standard deviation of the considered parameter (the instantaneous Reynolds Stress) to mitigate the dependence of the probability of selection to the distance to the wall, as del Álamo et al. did for vorticity [13] and Lozano-Durán et al. for Reynolds Stress [15].

Once the limit for the desired value has been properly established, all of the points in the flow can be evaluated against the designed criteria, obtaining a subset of the total points of the domain where the relevant magnitude (in this case, instantaneous Reynolds stress) is significant.

Subsequently, the coherent structures can be aggregated from the obtained points into a set of three-dimensional structures in a single time step, classifying them into *sweeps* and *ejections*. Figure 3 presents the schematic 2D representation of this step. This figure shows a section of a slice of the domain, where (a) marks, in black, all points fulfilling the criteria and (b) contains all of the structures aggregated, each one filled in a different colour. An efficient method to perform this step has been described in [35]. Finally, in order to reduce the huge number of structures, those with a volume of less of 30 wall units have been removed.

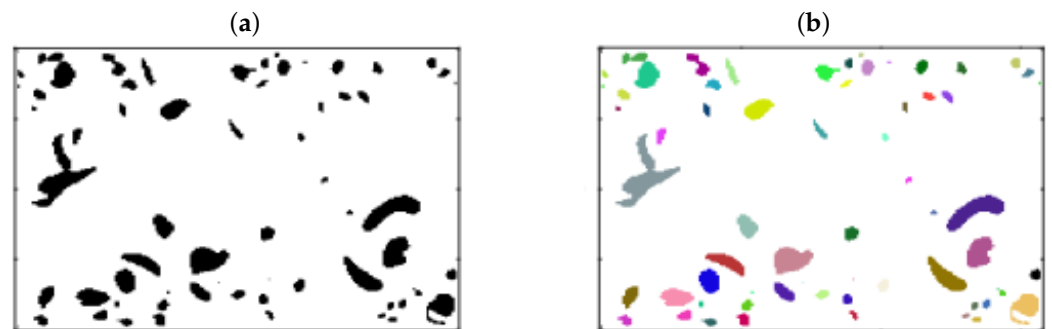


Figure 3. (a) Section of a slice of the fluid domain with all points fulfilling the criteria highlighted in black. (b) Coherent structures aggregated from the slice in (a), each filled in a different colour.

Finally, the temporal evolution of the sweeps and ejections must be reconstructed from the information available. This process does not have a deterministic solution, as there is no known analytic law defining the motion of *sweeps* and *ejections*. Several traditional programming (as opposed to machine learning) solutions have been used for this problem, such as the one used by Lozano-Durán and Jimenez, which computes the intersection of the structures in two consecutive time [10], and the one that was proposed by Muelder and Ma, which projects each structure onto the following time step and uses the projection an initial guess to aggregate the structures in that following time step [36]. Figure 4 shows the *sweeps* and *ejections* found in a section of the domain in two consecutive saved DNS time steps, with each independent structure colored differently. The objective of this step is to correlate them between the two time steps, so that the lifetime of the structures can be reconstructed later.

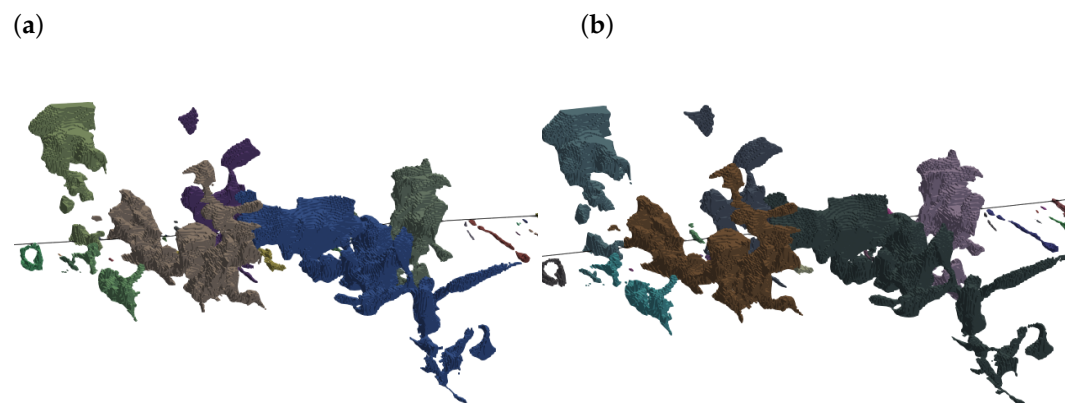


Figure 4. (a) Representation of some *sweeps* and *ejections* in a single time step of the DNS simulations. The (b) *Sweeps* and *ejections* in the same region following time step.

Once the full process is performed, the lifetime of the structures can be reconstructed for further study. Figure 5 shows several steps of the life of an *ejection*, extracted from the DNS simulation described in Section 2.1.

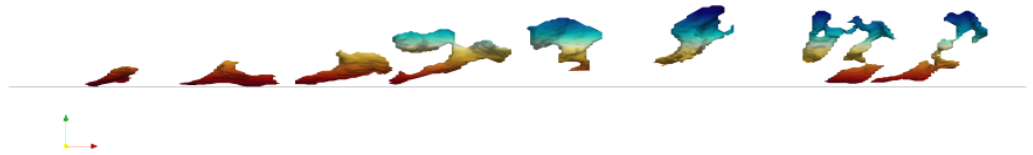


Figure 5. Representation of the shape and distance to the wall of an *ejection* through several steps of its lifetime.

2.3. Prediction of Geometrical Features of Sweeps and Ejections by Means of Multi Layer Perceptrons (MLP)

The tracking problem that was described in the previous section, understood as the correlation of structures from a time-step to the next, presents itself as a good opportunity for a neural network solution, as the classical programming solution is expensive computationally: it requires point-by-point access to a very large dataset in both memory contiguous and non-contiguous directions. To perform a simplification, the lack of any analytical description for the behaviour of *sweeps* and *ejections* can encourage the use of neural networks, as they have been proven to be universal approximators [17].

The problem must be formulated in terms of a series of inputs to provide to the network and a series of outputs that are expected as output. In this case, the geometrical features of the structures have been selected as both input and output. That way, no extra information regarding the flow needs to be passed to the routines that construct the structures, thus maximising the computational efficiency through the whole process. This geometrical features are:

- Centre of mass position (three components).
- Bounding box position, as given by the corners with minimum and maximum values for the three coordinates (six components).
- Volume (one component).

The objective of the neural network is to obtain the values of this set of features on the next time step to the one used as input, so that the structure that is closest to the prediction of all of the ones present in the next time step can be selected to be the continuation of the initial structure. Figure 6 shows the proposed implementation of the use of ANN in the process of tracking Qs in a DNS simulation: the objective of the ANN is to provide an estimation of the geometrical features of the set of Qs for the next evaluated time step. The predictions made can then be compared with the obtained geometrical features of the Qs in the next time step in order to extract the correspondence between the structures in both of the time steps.

Additionally notice that the idea of the presented algorithm is not to foresee the full characteristics of the Qs at t_{i+n} from time t_i , something that is extremely difficult if even possible, but to identify the structures obtained in step t_{i+n} with the ones of t_i .

Multi-Layer Perceptrons (MLP) are one of the most simple artificial neural networks. They consist on a series of nodes or *neurons* organised in fully connected layers. One node in an intermediate layer is connected to the nodes of the previous layer and produce a result by applying a function (activation function) to the values that are obtained from those nodes. The activation function is applied to the sum of each incoming connection multiplied by a *weight* and an overall *bias*, being the *weights* and *biases* of the full networks the parameters that must be adjusted by a training process (trainable parameters). For the present work, the rectified linear unit activation function was used for the neurons' internal activation function. Figure 7 contains a schematic representation of a neuron.

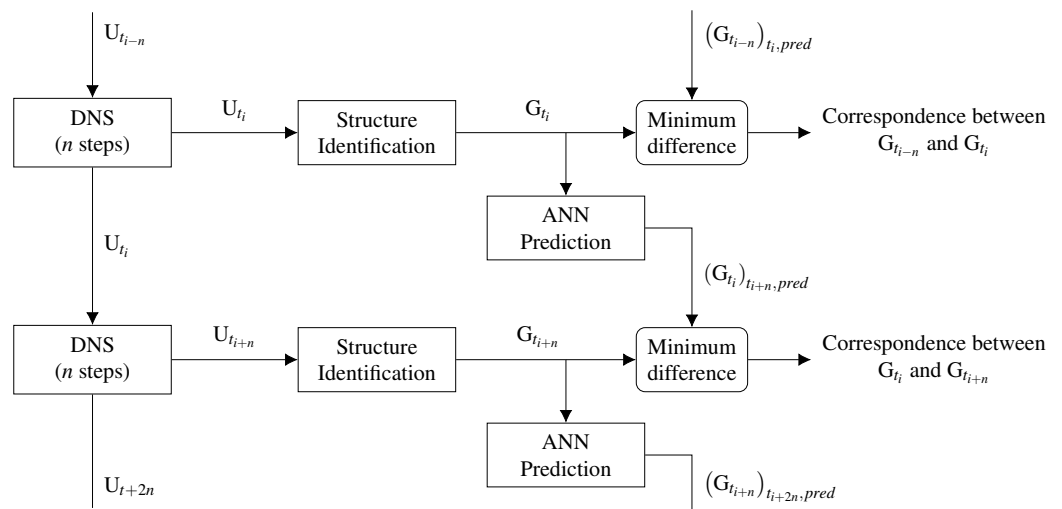


Figure 6. Flowchart of the proposed implementation of the use of Artificial Neural Networks (ANN) in the process of tracking Qs in a DNS simulation. U_{t_i} represents the velocity field in step t_i , and G_{t_i} the set of features of the Qs of time step t_i . The ANN takes the features of the Qs and predicts their values in the next timestep, so that they can be compared to the features of the Qs in the next time step.

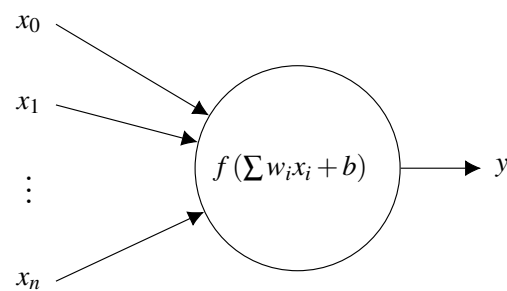


Figure 7. Schematic representation of a *neuron*, the building block of multi-layer perceptrons. It takes a vector as input and generates a scalar output by applying the function to the dot product of the input and the *weights* plus the *bias*.

The training process is an optimisation for the trainable parameters in the networks, in which a loss function that is representative of the difference between the outputs produced by the network and the expected outputs from a known series of cases is minimised. The hyperparameters, such as the learning rate, control this training process, which must be adjusted to ensure that the training process is successful.

The dataset used for the training, validation, and testing all of the networks in this work has been constructed from 994 non-consecutive time steps of the DNS simulation described in Section 2.1, the separation between each of them being 4.5 CFL, in which the sweeps and ejections have been tracked using *sweeps* and *ejections* classical computing techniques. A series of sequences have been obtained, each one representing the full lifetime of a *sweep* or an *ejection*, and from each sequence a series spanning n time steps a set of $n - 1$ data points, with each consisting of a pair input-output representing the evolution of the *sweep* or *ejection* from one time step to the next. This dataset has been split into three groups:

- 70% of all the available data has been used for training.
- 20% for validation, evaluating the value of the loss function and checking consistency with the values that were obtained in the training dataset.
- 10% for testing, examining the actual accuracy of the predictions.

Table 2 contains the number of sequences and data points for each of the groups.

Table 2. The number of data points (evolution of a structure from one time step to the next one) and number of sequences (total structure lifetime) in the dataset and in each one of the three sets it has been divided into.

	Number of Data Points	Number of Sequences
Training	13,713,321	880,362
Validation	3,894,344	251,532
Testing	1,962,774	125,766
Total	19,570,439	1,257,660

For this work, a number of different network topologies have been trained and their performance evaluated in the validation dataset. All of the evaluated network topologies present an input and output layer both with with 10 neurons to accommodate for the data shape to match the geometrical features that were selected for this. Between the input an output layers, a series of n hidden layers with u_i neurons $i = 1, \dots, n$ are present. Figure 8 contains a representation of the multi-layer perceptron used.

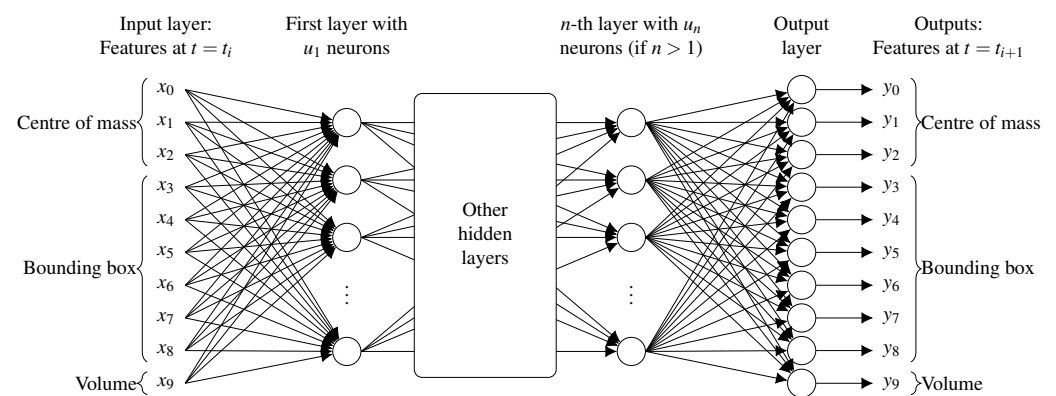


Figure 8. Schematic representation of the multi-layer perceptron network topology used, containing n hidden layers with u_i neurons each.

MLPs can have their architectures described by a simple matrix equation, with input vector x and output vector y . For a three-layer MLP (two hidden layers and the output layer), its matrix representation is as follows:

$$y = f^3 \left(W_{3,2} f^2 \left(W_{2,1} f^1 \left(W_{1,in} x + b_1 \right) + b_2 \right) + b_3 \right)$$

where f^j is the activation function for layer j , $W_{j,k}$ is the weights matrix from layer k to layer j and b_j is the bias vector for layer j .

As a metric for the network performance, the table presents the mean squared error of the predictions that were made by the network against the results that are present in the validation dataset. Table 3 lists the examined MLP architectures and the mean squared error that they produced in the validation dataset once converged.

Table 3. The list of evaluated network topologies, described by the number of hidden layers n and the neurons in each layer u_i , and the mean squared errors (MSE) produced by their predictions in the validation dataset. The IDs contain the number of hidden layers and the number of neurons of the first hidden layer, or two firsts hidden layers for the case where the first is equal.

ID	n	u_i	Trainable Parameters	MSE
MLP_1_8	1	{8}	178	0.0742
MLP_1_10	1	{10}	220	6.7×10^{-4}
MLP_2_10	2	{10, 10}	330	6.6×10^{-4}
MLP_3_10_8	3	{10, 8, 8}	370	0.0675
MLP_3_10_10	3	{10, 10, 10}	440	6.5×10^{-4}

In order to ensure that no overfitting is happening, the networks were trained for the same number of epochs than they took to reach a stable value of the loss function (MSE). During this training epochs, the validation Mean Squared Error was monitored for any upward trend. None of the studied networks presented any overfitting, which was made possible by the very high data to trainable parameters ratio.

Further sensitivity analysis to initial conditions was performed because of the similarity of the values of the Mean Squared Error for networks MLP_1_10, MLP_2_10 and MLP_3_10_10. Figure 9 contains the result of training these three architectures from 20 different random starting points. Not only the minimum error values are similar for these three MLPs, but their response to different initial conditions of the parameters do not present enough differences to establish that any of the three analysed architectures performs better.

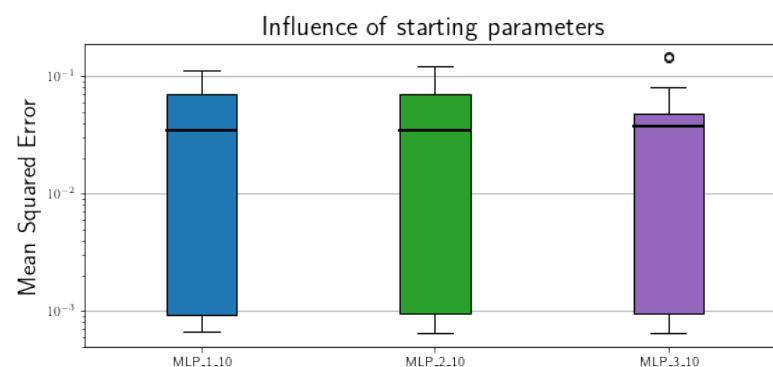


Figure 9. Limits and confidence intervals of the results of the MSE in the validation dataset after training MLP_1_10, MLP_2_10 and MLP_3_10_10 network architectures from 20 different random initial states.

2.4. Prediction of Geometrical Features of Sweeps and Ejections by Means of Recurrent Neural Networks

A Recurrent Neural Network is a neural network architecture that presents context-awareness by means of considering its previous state for the computation of their output. That feature allows for it to be specially fitted to predict time-series data, as opposed to the Multi-Layer Perceptron input to output mode of operation. However, RNNs are difficult to train as they present the problem of vanishing gradients: the gradients in the later layers restricts the learning rates upstream in the network, which can prevent the network from learning any further. Long-Short Term Memory (LSTM) networks, developed by Hochreiter and Schmidhuber [37], use a gating mechanism to control the dynamics of the recurrent connections to avoid the vanishing gradients problems. This gating mechanism was later improved by Gers et al. [38], which included the forget gate in the LSTM units. For this problem, due to the non-markovian nature of turbulence, LSTMs can be hypothesised to present an advantage with respects to MLPs, as the successful capture of the long-term dynamics of the structure can represent an advantage in predicting their position.

LSTM units consist on a cell and three gates: input, output, and forget. The cell contains the information on the LSTM and the gates regulate its information flow. The input gates take the information from the input vector and previous state of the whole LSTM layer to modify the cell state, the output gate regulates the flow of information downstream, and the forget gate regulates the information that stays in the LSTM cell [37].

The output value $y(t)$ of an LSTM unit with a forget gate is calculated, as follows, from the value of the n_{in} inputs $x_j(t)$ and previous state of the other n_u units in the layer $y_j(t-1)$:

$$\begin{aligned}
 y_f(t) &= f_f \left(\sum_{j=0}^{n_{in}} w_{f_j} x_j(t) + \sum_{j=0}^{n_u} v_{f_j} y_j(t-1) + b_f \right) \\
 y_{in}(t) &= f_{in} \left(\sum_{j=0}^{n_{in}} w_{in_j} x_j(t) + \sum_{j=0}^{n_u} v_{in_j} y_j(t-1) + b_{in} \right) \\
 y_{ou}(t) &= f_o \left(\sum_{j=0}^{n_{in}} w_{ou_j} x_j(t) + \sum_{j=0}^{n_u} v_{ou_j} y_j(t-1) + b_{ou} \right) \\
 c(t) &= y_f(t) \cdot c(t-1) + y_{in}(t) \cdot g \left(\sum_{j=0}^{n_{in}} w_{c_j} x_j(t) + \sum_{j=0}^{n_u} v_{c_j} y_j(t-1) + b_c \right) \\
 y(t) &= y_{ou}(t) \cdot h(c(t))
 \end{aligned}$$

where $y_f(t)$ is the value of the forget gate, $y_{in}(t)$ the value of the input gate, $y_{ou}(t)$ the value of the output gate and $c(t)$ is the internal state of the LSTM unit, w are the weight vectors with dimension n_{in} of the inputs for each gate, v are the weight vectors of dimension n_u of the other units in the layer for each gate, b are the biases for each gate and f_f , f_{in} and f_{ou} are the activation functions for each gate, g is the input squashing function, and h is the output squashing function, which, according to Gers et al., may not be needed [39]. Note that the LSTM layers feed the outputs of all their units in the previous step as inputs to all other units in the layer and, therefore, the number of trainable parameters per layer grows with the second power of the number of LSTM units per layer.

Several different LSTM configurations of n layers with u_i units per layer have been tested. The LSTM layers have been placed between an input of dimension 10 and an output layer consisting of 10 convolutional neurons, so that the output is compatible with the dataset dimensions. Table 4 lists the tested LSTM configurations and the mean squared error that they produced in the validation dataset once converged.

Table 4. List of evaluated Long Short-Term Memory (LSTM) network topologies, described by the number of hidden layers n and the number of LSTM units per layer u_i , and the mean squared errors (MSE) produced in the validation dataset.

ID	n	u_i	Trainable Parameters	MSE
LSTM_1_5	1	{5}	380	0.0120
LSTM_1_10	1	{10}	950	0.0073
LSTM_2_8	2	{8, 8}	1242	0.0049
LSTM_2_25	2	{25, 25}	8960	0.0042
LSTM_1_50	1	{50}	12,710	0.0030
LSTM_2_50	1	{50, 50}	32,910	0.0016

For the LSTM networks, overfitting was evaluated in the same way as in the previous case: monitoring the Mean Squared Error in the validation dataset for a number of epochs once the loss function (MSE) in the training dataset had stabilised. In this case, all of the evaluated networks presented some degree of overfitting and, therefore, a dropout rate of 20% was introduced for LSTM units, meaning that the internal status of the LSTM during training is reset, on average, in one of every five training data points. After the introduction

of the dropout, the networks were trained again to obtain the values that are presented in Table 4, and no overfitting was observed.

3. Results and Discussion

Once the networks have converged and it is assured that no overfitting is occurring, the next step consisted of relating the training metric (mean squared error) to the functionality that the networks are intended to serve. To do that, an environment that is similar to the operational one was set up for testing. Each prediction done by the networks is compared against all of the structures present in the next time step, for every step of every sequence:

1. Obtain the prediction of the geometrical features of the structure in the next time step using the network.
2. Obtain the norm of the difference between the prediction and feature vectors of all the structures present in the next time step.
3. The minimum value of that norm corresponds to the most similar structure present in the following time step. If they belong to the same *sweep* or *ejection*, the prediction is correct. Otherwise, it is not.

This process was applied to a subset of 2000 sequences from the testing dataset in order to evaluate the overall performance of the different networks. Figure 10 shows the success rate of the predictions of the different networks, as compared to their MSE.

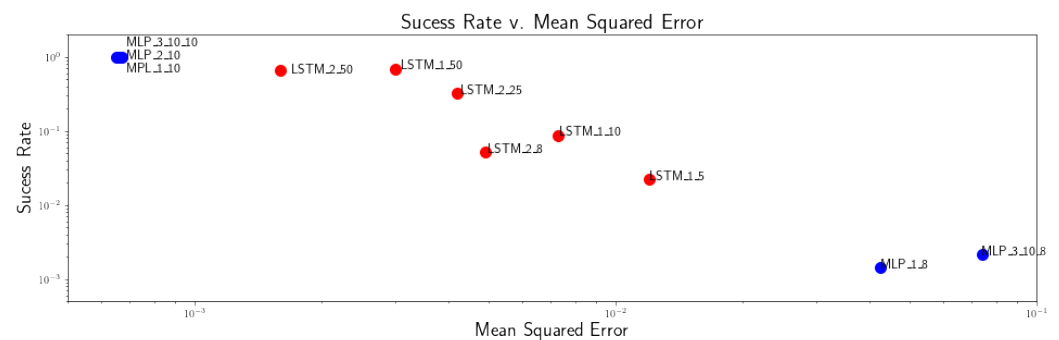


Figure 10. The success rate of the predictions of the different networks in a subset of 2000 sequences from the training data, and their Mean Squared Errors in the validation dataset.

One network of each group was further tested while using the same method in the full testing dataset in order to obtain further insight into the results that were produced by the networks. For the Multi-Layer Perceptrons, the MLP_1_10 was used and this network obtained an overall success rate of 98.74% for all of the predictions done in the testing dataset. For the LSTM networks, the LSTM_2_50 was chosen, obtaining an overall success rate of 67.59%. Figure 11 contains the distributions of MLP_1_10's results of the predictions with respect to Figure 11a the value of norm of the difference between the features vectors of the prediction and the selected structure, Figure 11b structure volume, and Figure 11c distance to the wall $y+$ of the structure' centre of mass. Figure 12 contains the same information for LSTM_2_50's predictions.

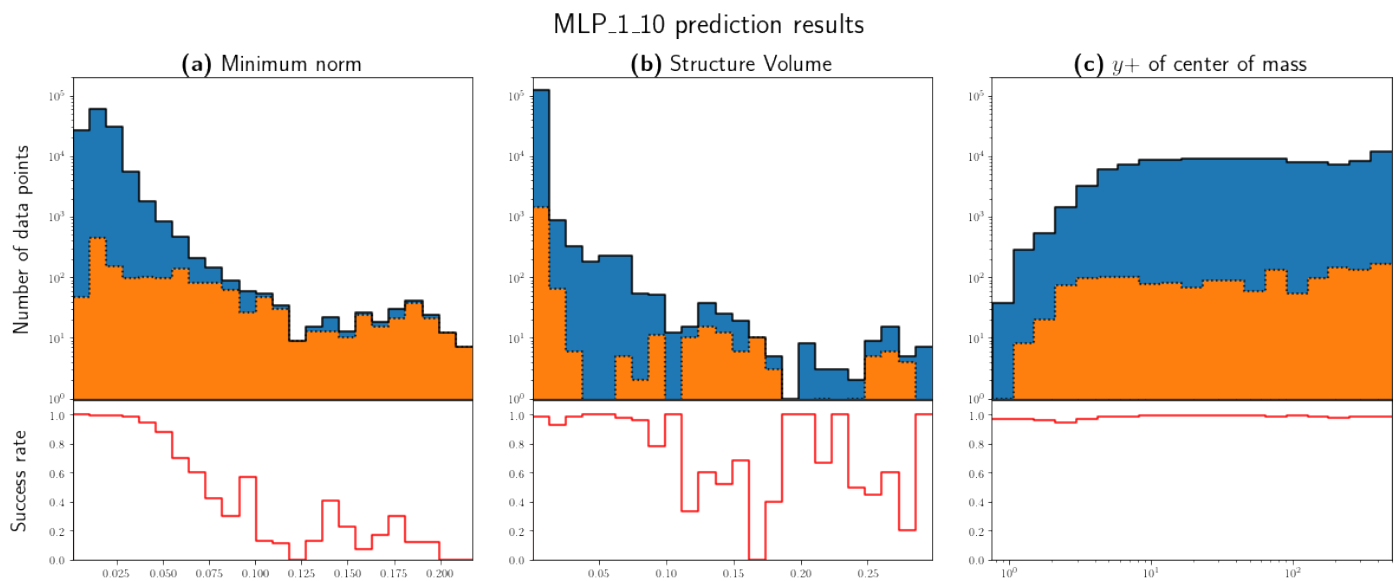


Figure 11. ■ Successful predictions, ■ Failed Predictions, - Total predictions, - Success rate. (a) Histogram of the MLP_1_10 prediction results (up) and success rate (down) with respect to the norm of the difference between the predicted structure features vector and the one of the selected structure of the target time step; (b) Histogram of the MLP_1_10 prediction results (up) and success rate (down) with respect to the original structure volume; and, (c) Histogram of the MLP_1_10 prediction results (up) and success rate (down) with respect to the original structure y_+ .

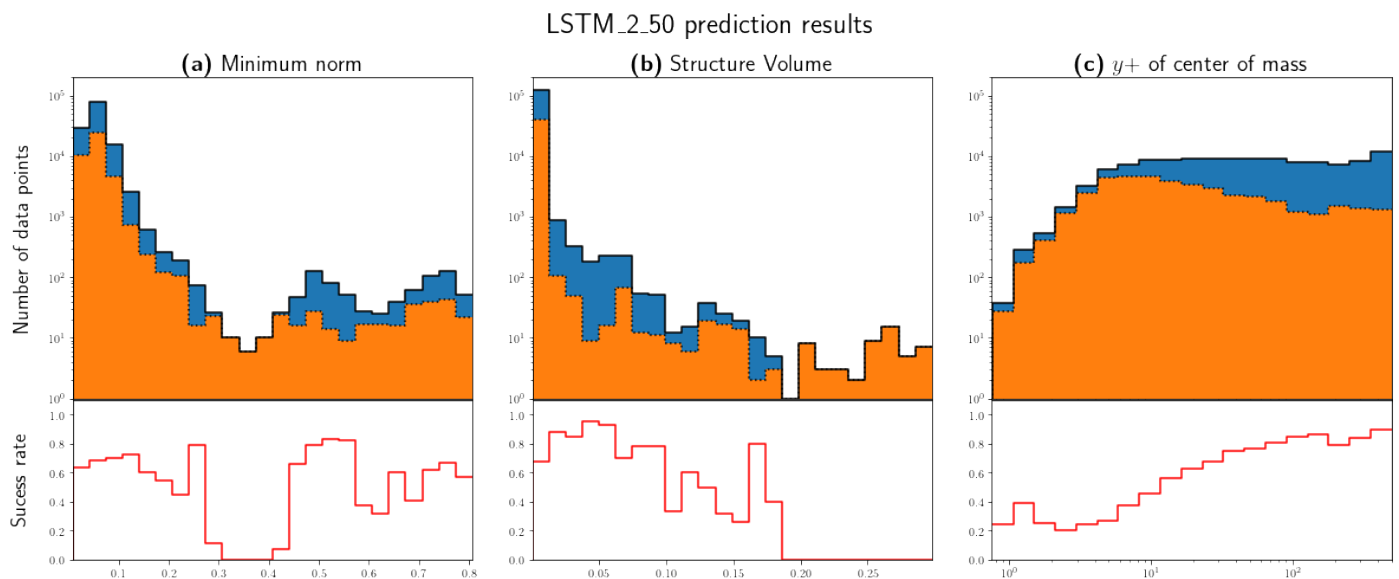


Figure 12. ■ Successful predictions, ■ Failed Predictions, - Total predictions, - Success rate. (a) Histogram of the LSTM_2_50 prediction results (up) and success rate (down) with respect to the norm of the difference between the predicted structure features vector and the one of the selected structure of the target time step; (b) Histogram of the LSTM_2_50 prediction results (up) and success rate (down) with respect to the original structure volume; and, (c) Histogram of the LSTM_2_50 prediction results (up) and success rate (down) with respect to the original structure y_+ .

Regarding the predictions that were made by MLP_1_10, the predictions are better for structures with lower volume, but there is no clear trend for the accuracy of the prediction as a function of the distance to the wall. As expected, the precision of the method is higher for a lower norm of the difference. Appendix A presents the coefficients of the tested network.

Overall, none of the LSTM architectures tested could achieve the error of the best MLP architecture or its precision. LSTM_2_50 presents an overall lower success rate in

the predictions, and it does not overcome the problems that MLP_1_10 presented with structures with higher volume.

The present work, due to its relatively low Reynolds number and size of the dataset, can be understood as a viability study for the incorporation of machine learning techniques in fluid structure tracking problems. For the practical introduction of the described methods in large Reynolds number channels (up to $Re_\tau = 10,000$, with meshes consisting of 8×10^{10} points [40]), it must provide a clear execution time advantage maintaining the precision of existing methods.

Thus, only summarising some of the tested multi-layer perceptrons has achieved said requirements in terms of precision. In terms of computational load, multi-layer perceptrons also present the advantage of not needing the history of the structure to generate the prediction, potentially saving the computational cost of reconstructing the full lifetimes of all the present structures of each analysed step.

4. Conclusions

In this work, we have compared the performance of several MLP and LSTM artificial neural networks in predicting the future geometrical features of the sweeps and ejections of a turbulent channel flow at a relatively low friction Reynolds number, $Re_\tau = 500$. Several multi-layer perceptron and recurrent (LSTM) neural network architectures were trained in order to predict the geometrical features of both structures in step $n + 1$ while using the knowledge gained in step n . Different benchmarks show that the tested MLPs proved to generate better prediction than the LSTM networks. However, that does not mean that no LSTM could produce accurate results. Probably more data and computational time for training and experimentation with the network topology are needed.

However, the solutions obtained with some of the tested MLP architectures were satisfactory, obtaining accurate predictions with a relatively simple architecture, meaning fast computations. Having proven the viability of this approach, the functionality of the network could be extended by adding some extra functionality to cover the cases in which the presented neural networks would not be enough: structure creation, structures crashing to form larger structure, structures splitting into smaller *children* structures, and structures dissipating and dying. These introductions would equate this method to the method that was used by Lozano-Durán et al. [10], which comes at a computational cost of checking the fluid domain and comparing for overlap in consecutive steps. The development of a faster method is a necessity because we intend to perform analyses of the *sweeps* and *ejections* at larger Reynolds numbers and/or bigger boxes for different flow.

Finally, the next step in the algorithm is to train it to follow the small vortexes with shorter life, and to reduce its memory needs in order to be able to simulate medium to high Reynolds numbers.

Author Contributions: Conceptualization, S.H. and L.M.G.-R.; methodology, S.H.; software, J.J.A.-F., F.N.-R., and J.C.J.R.; validation, J.J.A.-F., F.N.-R., and J.C.J.R.; formal analysis, J.J.A.-F., S.H., and L.M.G.-R.; investigation, J.J.A.-F.; resources, S.H.; writing—original draft preparation, J.J.A.-F.; writing—review and editing, S.H. and L.M.G.-R.; supervision, S.H.; funding acquisition, S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by RTI2018-102256-B-I00 of MINECO/FEDER. The computations of the new simulations were made possible by a generous grant of computing time from the Barcelona Supercomputing Centre, reference AECT-2020-2-0005.

Data Availability Statement: The raw data that support the findings of this study are available from the corresponding author upon reasonable request. One point statistics can be downloaded from the web page of our group: <http://personales.upv.es/serhocal/> (accessed on 13 February 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CFL	Courant–Friedrichs–Lewy condition
DNS	Direct Numerical Simulation
ML	Machine Learning
ANN	Artificial neural Network
MLP	Multi-layer perceptron
MSE	Mean Squared Error
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals

Appendix A. Weights and Biases of MLP_1_10

MLP_1_10 can be described as:

$$y = f^2(W_{2,1}f^1(W_{1,in}x + b_1) + b_2)$$

where f^1 and f^2 are the Rectifier Linear Unit (ReLU) function $f(x) = \max(0, x)$, x is the input features vectors scaled to the range $(0, 1)$ and the weights matrices and biases vectors are as follows:

$$\begin{array}{l}
 W_{1,in} = \begin{bmatrix} 0.087 & 0.464 & -0.3267 & -0.382 & -0.297 & -0.4144 & 0.3759 & -0.1478 & 0.0779 & 0.7566 \\ -0.3331 & 0.5551 & -0.0999 & 0.040 & 0.1863 & -0.1527 & -0.0288 & -0.5145 & 0.2281 & 0.5791 \\ 0.2727 & 0.5499 & -0.2724 & -0.046 & 0.3479 & 0.4961 & 0.0985 & 0.1795 & -0.20 & 0.6958 \\ -0.4474 & 1.3518 & -0.2158 & -0.7999 & -0.3272 & -0.0314 & 0.4993 & -0.058 & 0.0426 & 2.4128 \\ 0.0639 & 0.3354 & -0.1911 & -0.5413 & 0.0534 & 0.2987 & 0.4752 & -0.450 & 0.0129 & 0.5035 \\ -0.6186 & 0.7482 & -0.4178 & 0.1282 & -0.1273 & -0.052 & 0.0792 & 0.3593 & -0.1677 & 1.323 \\ 0.0954 & 0.3995 & 0.3595 & -0.0298 & -0.1099 & -0.4687 & -0.0464 & -0.1437 & 0.4926 & -0.0097 \\ 0.8319 & 0.6879 & 0.434 & -0.010 & -0.400 & 0.3825 & 0.093 & -0.6586 & -0.5243 & 1.5173 \\ 0.2551 & 0.3096 & -0.1399 & -0.0767 & -0.0331 & 0.0694 & 0.0823 & 0.1991 & -0.2991 & -0.688 \\ -0.090 & -1.214 & 0.483 & -0.1765 & 0.4517 & 0.3123 & -0.30 & 0.5473 & -0.1681 & -2.6362 \end{bmatrix} & b_1 = \begin{bmatrix} -0.15 \\ 2.3253 \\ -0.123 \\ -0.0824 \\ -0.3078 \\ -0.1198 \\ -0.0977 \\ -0.3545 \\ 0.3017 \\ 2.810 \end{bmatrix} \\
 \\
 W_{2,1} = \begin{bmatrix} -0.0866 & -0.2685 & 0.1129 & -0.1943 & -0.1595 & -0.1618 & -0.2171 & 0.5794 & 0.332 & -0.5484 \\ -0.069 & 0.4846 & 0.458 & -0.1014 & -0.0178 & 0.3396 & 0.5524 & -0.025 & 0.850 & -0.2666 \\ -0.0454 & 0.3871 & 0.0071 & 0.4928 & 0.0343 & 0.0547 & -0.0597 & -0.199 & -0.4736 & 0.5216 \\ -0.4837 & 0.3527 & 0.0352 & -0.3991 & -0.4975 & 0.3114 & 0.3077 & 0.3324 & -0.2448 & 0.200 \\ 0.2585 & -0.3268 & 0.1014 & 0.2576 & -0.199 & 0.4191 & -0.1864 & -0.2213 & -0.3148 & -0.3841 \\ -0.047 & -0.1589 & 0.4758 & 0.1386 & 0.0273 & -0.172 & -0.2346 & 0.0021 & 0.096 & -0.4138 \\ 0.890 & -0.660 & -0.1267 & 0.5318 & 0.9424 & -0.571 & -0.5886 & -0.5742 & 0.3416 & -0.1718 \\ -0.0395 & 0.1622 & -0.5074 & -0.1542 & 0.1226 & 0.0335 & -0.0663 & 0.235 & -0.2267 & -0.1586 \\ 0.1572 & 0.50 & -0.690 & 0.0959 & 0.1725 & 0.3926 & 0.520 & -0.7251 & -0.511 & -0.5142 \\ 0.0151 & -0.0359 & -0.190 & 0.0692 & -0.037 & 0.0179 & -0.1066 & 0.2818 & -0.6237 & 0.1262 \end{bmatrix} & b_2 = \begin{bmatrix} 0.026 \\ -1.362 \\ -0.4485 \\ 0.2879 \\ 0.0668 \\ -0.9171 \\ -1.2962 \\ -0.758 \\ 0.0528 \\ -0.1841 \end{bmatrix}
 \end{array}$$

References

1. Jiménez, J. Near-wall turbulence. *Phys. Fluids* **2013**, *25*, 101302. [\[CrossRef\]](#)
2. Kim, J.; Moin, P.; Moser, R. Turbulence statistics in fully developed channels flows at low Reynolds numbers. *J. Fluid Mech.* **1987**, *177*, 133–166. [\[CrossRef\]](#)
3. Hoyas, S.; Jiménez, J. Scaling of the velocity fluctuations in turbulent channels up to $Re_\tau = 2003$. *Phys. Fluids* **2006**, *18*, 011702. [\[CrossRef\]](#)
4. Simens, M.P.; Jimenez, J.; Hoyas, S.; Mizuno, Y. A high-resolution code for turbulent boundary layers. *J. Comput. Phys.* **2009**, *228*, 4218–4231. [\[CrossRef\]](#)
5. Avsarkisov, V.; Hoyas, S.; Oberlack, M.; García-Galache, J. Turbulent plane Couette flow at moderately high Reynolds number. *J. Fluid Mech.* **2014**, *751*, R1. [\[CrossRef\]](#)
6. Lozano-Durán, A.; Jiménez, J. Effect of the computational domain on direct simulations of turbulent channels up to $Re_\tau = 4200$. *Phys. Fluids* **2014**, *26*, 011702. [\[CrossRef\]](#)
7. Gandía-Barberá, S.; Hoyas, S.; Oberlack, M.; Kraheberger, S. The link between the Reynolds shear stress and the large structures of turbulent Couette-Poiseuille flow. *Phys. Fluids* **2018**, *30*, 041702, [\[CrossRef\]](#)
8. Lluesma-Rodríguez, F.; Hoyas, S.; Pérez-Quiles, M. Influence of the computational domain on DNS of turbulent heat transfer up to $Re_\tau = 2000$ for $Pr = 0.71$. *Int. J. Heat Mass Transf.* **2018**, *122*, 983–992. [\[CrossRef\]](#)
9. Chong, M.; Perry, A.; Cantwell, B. A general classification of three-dimensional flow fields. *J. Phys. A* **1990**, *2*, 765–777. [\[CrossRef\]](#)

10. Lozano-Durán, A.; Jiménez, J. Time-resolved evolution of coherent structures in turbulent channels: Characterization of eddies and cascades. *J. Fluid Mech.* **2014**, *759*, 432–471. [[CrossRef](#)]
11. Head, M.; Bandyopadhyay, P. New aspects of turbulent boundary-layer structure. *J. Fluid Mech.* **1981**, *107*, 297–338. [[CrossRef](#)]
12. Adrian, R.J.; Meinhart, C.D.; Tomkins, C.D. Vortex organization in the outer region of the turbulent boundary layer. *J. Fluid Mech.* **2000**, *422*, 1–54. [[CrossRef](#)]
13. Del Álamo, J.C.; Jiménez, J.; Zandonade, P.; Moser, R. Self-similar vortex clusters in the turbulent logarithmic region. *J. Fluid Mech.* **2006**, *561*, 329–358. [[CrossRef](#)]
14. Jiménez, J. Coherent structures in wall-bounded turbulence. *J. Fluid Mech.* **2018**, *842*, P1. [[CrossRef](#)]
15. Lozano-Durán, A.; Flores, O.; Jiménez, J. The three-dimensional structure of momentum transfer in turbulent channels. *J. Fluid Mech.* **2012**, *694*, 100–130. [[CrossRef](#)]
16. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [[CrossRef](#)]
17. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [[CrossRef](#)]
18. Srinivasan, P.; Guastoni, L.; Azizpour, H.; Schlatter, P.; Vinuesa, R. Predictions of turbulent shear flows using deep neural networks. *Phys. Rev. Fluids* **2019**, *4*, 054603. [[CrossRef](#)]
19. Park, J.; Choi, H. Machine-learning-based feedback control for drag reduction in a turbulent channel flow. *J. Fluid Mech.* **2020**, *904*, A24. [[CrossRef](#)]
20. Ling, J.; Kurzawski, A.; Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **2016**, *807*, 155–166. [[CrossRef](#)]
21. Jiang, C.; Mi, J.; Laima, S.; Li, H. A novel algebraic stress model with machine-learning-assisted parameterization. *Energies* **2020**, *13*, 258. [[CrossRef](#)]
22. Tokarev, M.; Palkin, E.; Mullyadzhyanov, R. Deep Reinforcement Learning Control of Cylinder Flow Using Rotary Oscillations at Low Reynolds Number. *Energies* **2020**, *13*, 5920. [[CrossRef](#)]
23. Bahrami, M.; Akbari, M.; Bagherzadeh, S.A.; Karimipour, A.; Afrand, M.; Goodarzi, M. Develop 24 dissimilar ANNs by suitable architectures & training algorithms via sensitivity analysis to better statistical presentation: Measure MSEs between targets & ANN for Fe–CuO/Eg–Water nanofluid. *Phys. A Stat. Mech. Its Appl.* **2019**, *519*, 159–168.
24. Giwa, S.; Sharifpur, M.; Goodarzi, M.; Alsulami, H.; Meyer, J. Influence of base fluid, temperature, and concentration on the thermophysical properties of hybrid nanofluids of alumina–ferrofluid: Experimental data, modeling through enhanced ANN, ANFIS, and curve fitting. *J. Therm. Anal. Calorim.* **2020**, 1–19. doi:10.1007/s10973-020-09372-w. [[CrossRef](#)]
25. Shadloo, M.S.; Rahmat, A.; Karimipour, A.; Wongwises, S. Estimation of pressure drop of two-phase flow in horizontal long pipes using artificial neural networks. *J. Energy Resour. Technol.* **2020**, *142*, 112110. [[CrossRef](#)]
26. Singh, A.P.; Medida, S.; Duraisamy, K. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA J.* **2017**, *55*, 2215–2227. [[CrossRef](#)]
27. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [[CrossRef](#)]
28. Duraisamy, K.; Iaccarino, G.; Xiao, H. Turbulence modeling in the age of data. *Annu. Rev. Fluid Mech.* **2019**, *51*, 357–377. [[CrossRef](#)]
29. Kraheberger, S.; Hoyas, S.; Oberlack, M. DNS of a turbulent Couette flow at constant wall transpiration up to $Re_\tau = 1000$. *J. Fluid Mech.* **2018**, *835*, 421–443. [[CrossRef](#)]
30. Alcántara-Ávila, F.; Hoyas, S.; Pérez-Quiles, M. DNS of thermal channel flow up to $Re_\tau = 2000$ for medium to low Prandtl numbers. *Int. J. Heat Mass Transf.* **2018**, *127*, 349–361. [[CrossRef](#)]
31. Lele, S.K. Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.* **1992**, *103*, 16–42. [[CrossRef](#)]
32. Spalart, P.R.; Moser, R.D.; Rogers, M.M. Spectral methods for the Navier-Stokes equations with one infinite and two periodic directions. *J. Comput. Phys.* **1991**, *96*, 297–324. [[CrossRef](#)]
33. Wallace, J.M.; Eckelmann, H.; Brodkey, R.S. The wall region in turbulent shear flow. *J. Fluid Mech.* **1972**, *54*, 39–48. [[CrossRef](#)]
34. Moisy, F.; Jiménez, J. Geometry and clustering of intense structures in isotropic turbulence. *J. Fluid Mech.* **2004**, *513*, 111. [[CrossRef](#)]
35. Aguilar-Fuertes, J.J.; Noguero-Rodríguez, F.; Jaen-Ruiz, J.C.; García-Raffi, L.M.; Hoyas, S. Following Vortices in Turbulent Channel Flows. In *International Workshop on Soft Computing Models in Industrial and Environmental Applications*; Springer: Berlin, Germany, 2020; pp. 490–496.
36. Muelder, C.; Ma, K.L. Interactive feature extraction and tracking by utilizing region coherency. In Proceedings of the 2009 IEEE Pacific Visualization Symposium, Beijing, China, 20–23 April 2009; pp. 17–24.
37. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
38. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Comput.* **1999**, *12*, 2451–2471. [[CrossRef](#)]
39. Gers, F.A.; Schraudolph, N.N.; Schmidhuber, J. Learning precise timing with LSTM recurrent networks. *J. Mach. Learn. Res.* **2002**, *3*, 115–143.
40. Hoyas, S.; Oberlack, M.; Kraheberger, S.; Alcántara-Ávila, F. Turbulent channel flow at $Re_\tau = 10000$. In Proceedings of the 72nd Annual Meeting of the APS Division of Fluid Dynamics, Seattle, WA, USA, 23–26 November 2019; p. H19-001.