Journal of Computational Physics **182,** 1–26 (2002)
doi:10.1006/jcph.2002.7146

Neural Network Modeling for Near Wall Turbulent Flow

Michele Milano[1] and Petros Koumoutsakos[2]

Institute of Computational Sciences, ETH Zentrum, CH-8092 Zürich, Switzerland
E-mail: milano@inf.ethz.ch, petros@inf.ethz.ch

Received May 23, 2001; revised January 8, 2002

A neural network methodology is developed in order to reconstruct the near wall field in a turbulent flow by exploiting flow fields provided by direct numerical simulations. The results obtained from the neural network methodology are compared with the results obtained from prediction and reconstruction using proper orthogonal decomposition (POD). Using the property that the POD is equivalent to a specific linear neural network, a nonlinear neural network extension is presented. It is shown that for a relatively small additional computational cost nonlinear neural networks provide us with improved reconstruction and prediction capabilities for the near wall velocity fields. Based on these results advantages and drawbacks of both approaches are discussed with an outlook toward the development of near wall models for turbulence modeling and control.

Key Words: neural networks; turbulent flows; machine learning; adaptive systems.

1. INTRODUCTION

The development of low-order models for wall bounded turbulent flows is of paramount importance for the construction of turbulence models for LES and RANS calculations as well as for the development of efficient flow control strategies. A prototypical example of turbulent wall bounded flows is the turbulent channel flow. In turbulent channel flows the near wall region contains the most fundamental mechanisms for the generation of turbulence. The near wall region is characterized by coherent vortical structures, manifested by alternating low- and high-speed streamwise wall streaks [1]. The periodic formation and breakdown of these vortical structures is associated with the production of turbulence in wall bounded flows [2]. These structures suggest that suitable low-order models can be

[1] Present affiliation: Graduate Aeronautics Laboratories, California Institute of Technology, Pasadena, CA 91125.

[2] Also at CTR, NASA Ames 202A-1, Moffett Field, CA 94035.

0021-9991/02 $35.00

developed by projecting the governing Navier–Stokes equations, on a properly selected lower dimensional phase subspace.

For this projection a reasonable selection criterion for the base of the manifold is the maximization of the energy content. For a dynamical system this can be accomplished by applying the Karhunen–Loeve decomposition to a data set that is representative of the dynamics we wish to approximate [6, 8–10]. This operation is also called principal components analysis (PCA).

The application of the PCA technique to turbulent flows was pioneered by Lumley [5], as the proper orthogonal decomposition (POD). In this decomposition, the spatio-temporal velocity field is represented as the superposition of an infinite number of spatial structures whose amplitude changes with time. This decomposition is optimal in the sense that it maximizes the energy content of any finite truncation of the representation. The POD has been successfully applied to the derivation of low-dimensional models of the Navier–Stokes equations [3, 7, 8, 10], describing systems such as wall boundary layers, convective flows, and turbulent channel flows.

However, the application of the POD to three-dimensional turbulent flows is hindered by the size of the computational problem; studies of the performance of the POD have shown that as the Reynolds number increases the convergence rate of the truncated POD expansion to a given fraction of the total flow energy decreases, requiring larger numbers of eigenfunctions to represent the flow adequately [11].

The difficulty related to the size of the computational problem can be circumvented by considering a domain as small as possible when the POD is performed. Jimenez and Moin [12] found the smallest computational box in which turbulence can be sustained, naming it the minimal flow unit (MFU). Using this concept, the POD has been used to reconstruct an approximation to the near wall flow using wall shear stress measurements [13] as well as to derive a reduced-order dynamical model for the MFU [14]. However, in a MFU fewer eigenfunctions are needed to capture a given amount of energy than in a full channel flow [15], which means that a POD of an MFU may not provide adequate information on higher order modes.

Since the POD is the linear PCA, it is plausible to consider its nonlinear extension. The nonlinear extension of the POD can be easily formulated when its relation with a linear neural network (NN) is recognized; more specifically, the POD can be viewed as the application of a linear neural network to the identity-reconstruction problem. It can be shown analytically that training a linear neural network to perform an identity mapping on a given data set is equivalent to performing a linear PCA on the data set [16].

A neural network has been used to learn the correlation between wall shear stresses and wall actuation, in order to implement opposition control in a turbulent channel flow [27]. In this paper we focus on the use of a nonlinear neural network to perform a PCA, which extends and generalizes the POD methodology, providing in many cases a more efficient representation of the data [17]. The development and implementation of a neural network for reconstruction of the near wall flow field are examined for the first time in this paper. It will be also shown that the linear POD can be formulated as a specific class of linear neural networks, thus consistently generalizing the linear POD.

A neural network model has also been set up for unsteady surface pressure on an airfoil, having as inputs the instantaneous angle of attack and angular velocity [25]. This model has been set up using experimental data, and it can also be used to set up controllers for the wing motion [26]. A drawback of the neural network approach is the increased computational cost

associated with the training of the neural network. However, the training of a NN via back propagation is an inherently parallel process that can take advantage of today's powerful computer architectures. The selection of the data and the presentation to the NN are key aspects of the learning process. In this work, in order to ensure that the size of the data is computationally reasonable, and that the dynamics of the system contain all the information carried by higher order modes, the velocity field is sampled from a DNS of a full channel flow over a wall region containing an MFU.

In this paper we present results from the reconstruction of the near wall flow in a turbulent channel flow using wall only information. It is shown that a simple nonlinear neural network structure can be used to perform the same modeling tasks performed by the POD, but with better data compression capabilities. The advantages and drawbacks of this approach are discussed.

This paper is organized as follows: in Section 2 the linear POD is outlined, the basic structure of a neural network is presented, and the equivalence of a linear neural network to the linear POD is described; in Section 3 a nonlinear neural network extending the POD methodology is introduced, and linear POD and nonlinear PCA are compared in performance on the application to data coming from the randomly forced Burgers equation and from a turbulent flow contained in a MFU; in Section 4 it is shown how a nonlinear neural network can be used to derive a near wall model based on wall measurements, and the performance of this model is compared with the performance of a linear POD model; in Section 5 we present concluding remarks.

2. LINEAR POD AND NEURAL NETWORKS

In this section we present the POD and we outline its equivalence to a linear neural network. We consider two fundamental cases, pertaining to turbulent flow fields:

(I) no homogeneity in streamwise and spanwise directions
(II) homogeneity in streamwise and spanwise directions.

2.1. The Linear POD for a Nonhomogeneous Flow

We consider a velocity flow field observed in a stationary box Ω. The velocity field $\mathbf{u}(\mathbf{x}, t)$ can be expressed as

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{U}(\mathbf{x}) + \mathbf{u}'(\mathbf{x}, t), \tag{1}$$

where $\mathbf{x} = (x, y, z)$; $\mathbf{U}(\mathbf{x})$ is the mean and $\mathbf{u}'(\mathbf{x}, t)$ is the fluctuating velocity component.

Using the Karhunen–Loeve theorem $\mathbf{u}'$ can be expressed as

$$\lim_{n\to\infty} \hat{\mathbf{u}}'(\mathbf{x}, t) = \mathbf{u}'(\mathbf{x}, t), \tag{2}$$

where

$$\hat{\mathbf{u}}'(\mathbf{x}, t) = \sum_{i=1}^{n} a_i(t) \cdot \phi_i(\mathbf{x})$$

and ϕ_i are the first n orthonormal eigenfunctions, to which the coefficients a_i correspond; a_i are uncorrelated random coefficients of variance λ_i.

The orthonormal eigenfunctions $\phi_i(\mathbf{x})$ can be computed by considering an ensemble of K velocity fields $\{\mathbf{u}'(\mathbf{x}, t_k)\}_{k=1}^{K}$, and solving the Fredholm integral equation,

$$\int \langle u'_j(\mathbf{x}, t_k) u'_h(\mathbf{x}', t_k)\rangle \phi_i(\mathbf{x}')\, dx' = \lambda_i \phi_i(\mathbf{x}), \tag{3}$$

where angle brackets indicate averaging over the ensemble, the indices $j, h = 1, 2, 3$ denote the three velocity components, $u'_j(\mathbf{x}, t) u'_h(\mathbf{x}', t)$ is an outer product, and the index i denotes the ith eigenfunction ϕ_i with eigenvalue λ_i.

The POD decomposition has three important features: (i) it is optimal in the sense that any finite truncation of (2) captures the maximum possible energy [9]: (ii) both the POD components calculation and the original field reconstruction are written as matrix/vector products, i.e., linear operations; (iii) the coefficients $a_i(t)$ depend on time only, while the eigenfunctions $\phi_i(\mathbf{x})$ depend only on space.

Considering a spatial mesh discretizing the domain Ω, we represent the linear POD in matrix/vector form by writing the velocity fields as $\mathbf{u}'(\mathbf{x}_g, t)|_{\mathbf{x}_g \in \Omega} = \mathbf{u}'_g(t)$, where $\mathbf{x}_g$ are the coordinates of the N mesh nodes, and $\mathbf{u}'_g(t)$ are vectors of length $3N$ that collect the three velocity components from the mesh nodes. In this work a subscript g denotes this implicit spatial dependence. We also write the eigenfunctions $\phi_i(\mathbf{x})$ as $3N \times 1$ vectors, forming a matrix Φ_g with the n eigenfunctions as columns. Taking into account orthonormality of the eigenfunctions we can write the complete POD transformation of the flow, comprising the computation of the eigenvalues and the subsequent flow reconstruction

$$\begin{aligned} \mathbf{a}(t) &= \Phi_g^T \cdot \mathbf{u}'_g(t) \\ \hat{\mathbf{u}}'_g(t) &= \Phi_g \cdot \mathbf{a}(t), \end{aligned} \tag{4}$$

where $\mathbf{a}(t)$ is a $n \times 1$ vector containing the eigenvalues, and the superscript T indicates transposition. In what follows it will be shown that with this notation the POD can be regarded as a linear neural network, and a generalization of this structure will be obtained as shown in the following section.

2.2. Nonlinear and Linear Neural Networks

A neural network structure can be defined as a collection of interconnected neurons. A neuron consists of a weighted summation unit and an activation function that relates the neuron's input and output. The nonlinear neural network implemented in this paper is called a multilayered feed-forward neural net (MFFN) [18], which can be expressed as

$$\begin{aligned} \mathbf{x}_1 &= f(\mathbf{W}_1 \cdot \mathbf{x}_{in}) \\ \mathbf{x}_2 &= f(\mathbf{W}_2 \cdot \mathbf{x}_1) \\ &\cdots \\ \mathbf{x}_i &= f(\mathbf{W}_i \cdot \mathbf{x}_{i-1}) \\ &\cdots \\ \mathbf{x}_{out} &= \mathbf{W}_{N_l} \cdot \mathbf{x}_{N-1}, \end{aligned} \tag{5}$$

where $\mathbf{x}_i$ is the output of the layer i, N_l is the number of layers, $\mathbf{x}_{in}$ is the network input vector, $\mathbf{W}_i$ are the weight matrices for each layer, $\mathbf{x}_{out}$ is the network output vector. The function $f(\cdot)$ is a monotonic nondecreasing function of its input. It serves as a way to introduce

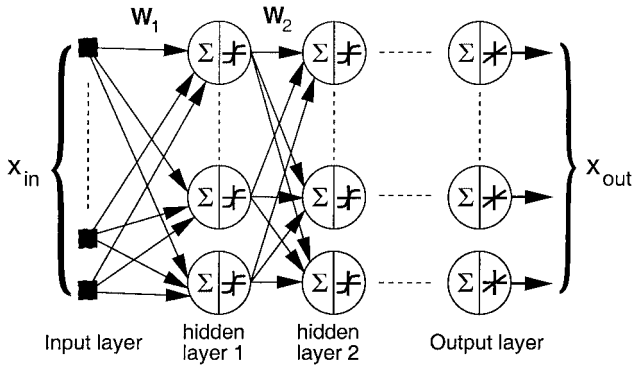

FIG. 1. Layer representation of a nonlinear neural network structure.

nonlinearity in the neural network and usually the hyperbolic tangent (i.e., $f(x) = \tanh(x)$) is the function of choice. A neural network is called *linear* if the function $f(x)$ is linear.

$$f(\mathbf{W}_i \cdot \mathbf{x}_{i-1}) = \mathbf{W}_i \cdot \mathbf{x}_{i-1} \quad (i = 1, \ldots, N_l - 1); \tag{6}$$

i.e., no nonlinearities are applied to the output of any neuron.

Neural networks are inherently parallel computational structures which can be represented by using a layer structure (Fig. 1). In this structure the neuron j in layer i performs the operation

$$x_{i,j} = f\left(\sum_{k=1}^{n_{i=1}} w_{i,j,k} \cdot x_{i-1,k}\right), \tag{7}$$

where n_{i-1} is the number of neurons in the previous layer, or the number of inputs if it is the first hidden layer.

From now on, the operations detailed here will be indicated concisely as

$$\mathbf{x}_{out} = NN(\mathbf{x}_{in}). \tag{8}$$

All the neurons in a given layer can operate at the same time, since there are no interneuron connections in this model. There are three types of layers:

(i) an input layer, which is used only to provide inputs to the neural network,

(ii) hidden layers that process the information received by the previous layer and provide the inputs to the following layer, and

(iii) an output layer, that is, the layer providing the network's output.

Each row in the matrices $\mathbf{W}_i$ contains the synapses of one neuron in the network. It can be demonstrated that a MFFN having at least one hidden layer with an arbitrary number of neurons, and using any continuous monotonic nondecreasing, bounded nonlinearity in the hidden layer, can approximate any continuous function on a compact subset; i.e., this structure is a universal approximator [18].

Having fixed the neural network architecture and the set of input/output pairs, to approximate values, a neural network can be trained using the so-called back-propagation

algorithm [18]. More specifically, assuming that a data set composed of M input/output pairs is available, the norm must be minimized,

$$\mathrm{SSE}(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{M} \left\|\mathbf{y}_k^d - \mathbf{y}_k(\mathbf{x}_k, \mathbf{w})\right\|^2, \tag{9}$$

where SSE is the sum of the squares of the errors, the superscript d denotes the desired output corresponding to the kth input $\mathbf{x}_k$, $\mathbf{y}_k(\mathbf{x}_k, \mathbf{w}) = NN(\mathbf{x}_k, \mathbf{w})$, and $\mathbf{w}$ is a vector containing all the values for the weights of the neural network. The back-propagation algorithm is an efficient way to compute the gradient $\nabla_{\mathbf{w}}\mathrm{SSE}$. The network weights are determined adaptively by applying a gradient descent technique to (9). Usually this adaptation process, called training in the literature, terminates when SSE reaches a given threshold.

2.3. The POD as a Linear Neural Network

A linear neural network performing an identity mapping for a flow field $\mathbf{u}'_g(t)$ can be expressed as

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{W}_1 \cdot \mathbf{u}'_g(t) \\ \hat{\mathbf{u}}'_g(t) &= \mathbf{W}_2 \cdot \mathbf{x}_1, \end{aligned} \tag{10}$$

where $\mathbf{W}_1 \in R^{n\times 3N}$, $\mathbf{W}_2 \in R^{3N\times n}$ are the network's weight matrices and N is the number of grid points containing the velocity field.

Setting $\mathbf{W}_1 = \Phi_g^T$ and $\mathbf{W}^2 = \Phi_g$ in Eqs. (10) Eqs. (4) are recovered. These equations are the equations of linear POD. Therefore the linear POD operations can be interpreted as the operations of a linear neural network.

The training of this neural network is conducted using standard back propagation [18] performing an identity mapping, thus minimizing the following norm:

$$\mathrm{SSE}(\mathbf{w}) = \frac{1}{2}\sum_{h=1}^{M} \left\|\mathbf{u}_g'^h - \hat{\mathbf{u}}_g'^h(\mathbf{w})\right\|^2. \tag{11}$$

Apart from a constant multiplicative factor this is the squared distance between the reconstruction $\hat{\mathbf{u}}'_g$ and the original $\mathbf{u}'_g$, i.e., the same norm that is minimized by the POD. Hence after this network is trained the weight matrices will be what are known to be the optimal matrices from the linear POD theory: $\mathbf{W}_1 = \Phi_g^T$ and $\mathbf{W}_2 = \Phi_g$. Therefore by training this neural network the optimal linear POD eigenfunctions are recovered.

Furthermore, it can also be demonstrated [16] that the only minimum of (11) is the linear POD solution, and all the other points in the weight space where the gradient vanishes are saddle points. Therefore, training the linear neural network (10) to perform an identity mapping on a data set is equivalent to performing a POD on the same data set.

2.4. Neural Network Training

As outlined earlier, the NN training is expressed as a minimization problem for the function (9). The training can be performed in two different ways, depending on the

availability of the input/output pairs for the training set:

- Offline training: If all the M input/output pairs are available at the same time, then the gradient of the function (9) can be computed explicitly and a standard minimization problem is set up.
- Online training: In many practical applications the size of the input/output pairs can be too large to allow for the storage of a sufficient number of input/output pairs or data are being produced in a continuous manner, while the network is being trained; in these cases the gradient of (9) is approximated by using only one pair at a time in the computation. The resulting algorithm is a stochastic gradient descent.

In this paper we use online learning for all the examples considered.

There are many techniques to stabilize and improve the convergence rate of online training; the most common is to define a separate training rate for each weight,

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mathrm{diag}(\mathbf{p}_k) \cdot \delta_k, \quad \delta_k = \nabla_{\mathbf{w}_k} \mathrm{SSE}(\mathbf{w}_k), \tag{12}$$

where $\mathbf{p}_k$ are the learning rates of the weights $\mathbf{W}_k$. In this work we implement an accelerated learning algorithm as proposed in [22]. The learning rates are adapted also by using an exponentiated gradient descent

$$\ln(\mathbf{p}_k) = \ln(\mathbf{p}_{k-1}) - \mu \frac{\partial NN(\mathbf{x}_k)}{\partial \ln(\mathbf{p}_k)} \tag{13}$$

$$\mathbf{p}_k = \mathbf{p}_{k-1} \exp(\mu \cdot \mathrm{diag}(\delta_k) \cdot \mathbf{r}_k), \quad \mathbf{r}_k = \frac{\partial \mathbf{W}_k}{\partial \ln(\mathbf{p}_k)}, \tag{14}$$

where μ is a global meta-learning rate. In this way it is ensured that all the $\mathbf{p}_k$ are strictly positive, and they will also cover a wide dynamic range, thus speeding up the overall weight adaptation process. For further details on the neural network training algorithm the reader is referred to [22].

3. THE NONLINEAR PCA

Equations (10) can be easily generalized, by adding two nonlinear layers to the basic linear neural network [19].

The resulting model, represented in graphic form in Fig. 2, can be expressed as

$$\mathbf{x} = \mathbf{W}_2 \cdot \tanh(\mathbf{W}_1 \cdot \mathbf{u}'_g(t)) \tag{15}$$

$$\hat{\mathbf{u}}'_g(t) = \mathbf{W}_4 \cdot \tanh(\mathbf{W}_3 \cdot \mathbf{x}), \tag{16}$$

where $\mathbf{W}_1 \in R^{n_1 \times 3N}$, $\mathbf{W}_2 \in R^{n_2 \times n_1}$, $\mathbf{W}_3 \in R^{n_1 \times n_2}$, $\mathbf{W}_4 \in R^{3N \times n_1}$, $\mathbf{x}$ are the n_2 nonlinear PCA components, the nonlinear analogue of the n components for the linear POD. This MFFN is also trained to perform an identity mapping, thus minimizing the norm (11) as in the linear case.

Note that it is necessary to add two nonlinear layers to the linear PCA neural network structure, because this is the minimum number of layers enforcing a nonlinear mapping when a PCA is performed [20]. This allows as to add more degrees of freedom to the model. However, the NN training implies additional computational cost, as it is an iterative process. On the other hand, this structure can be much more effective than the PCA in compressing data because of the additional degrees of freedom provided by the nonlinear transformation.

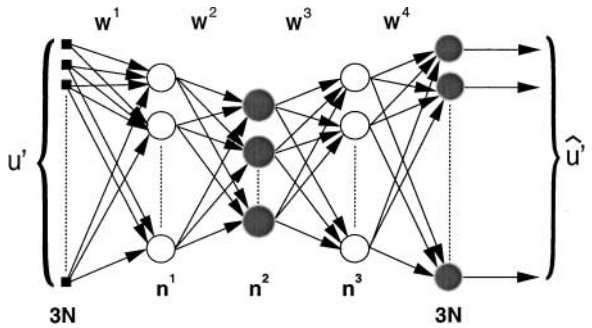

FIG. 2. Layer representation of a nonlinear PCA structure; the gray neurons are linear. The number of neurons comprising each layer is indicated at the bottom.

3.1. Simplifications for Nonhomogeneous Flow

It has been shown that in the nonhomogeneous case the PCA of a flow can be performed by training a nonlinear neural net to perform an identity mapping of the flow.

A straightforward way to do this would be to "stack" together the different components in the input and output fields of a single neural network [6]. However, in this case the number of degrees of freedom becomes very large, about 2×10^7 in the example reported in the next section. The resulting network becomes difficult to train with an iterative method. Therefore in the nonhomogeneous case it was decided to neglect correlations between different components of the flow field in order to avoid the training of very large networks. Although this is quite a crude approximation, our results indicate that it is possible to obtain good performance even under this assumption.

When correlations between velocity components are neglected, the formulation of the PCA problem can be changed by considering a separate neural network for each velocity component, thus reducing the size of the input and output fields of the network by a factor of 3.

Using the Karhunen–Loeve theorem each component u_j of $\mathbf{u}'$ can be expressed as

$$u'_j(\mathbf{x}, t) = \sum_{i=1}^{n_j} a_{i,j}(t) \cdot \phi_{i,j}(\mathbf{x}), \quad j = 1, 2, 3, \tag{17}$$

where $\Phi_{i,j}$ are the $n_j (j = 1, 2, 3)$ orthonormal eigenfunctions, to which the coefficients $a_{i,j}$ correspond; $a_{i,j}$ are uncorrelated random coefficients of variance $\lambda_{i,j}$. In all the following, unless otherwise specified, the subscript j will be used to denote the three components of a velocity field. Therefore, all the equations containing j are to be read as a set of three equations, one for each flow field component; we will also use vectors $\mathbf{u}'_{g,j}(t)$, each containing the component $u'_j(\mathbf{x}, t)$ measured on the N mesh points.

The equations of the three neural networks performing the nonlinear PCA become

$$\mathbf{x}_j = \mathbf{W}_{2,j} \cdot \tanh(\mathbf{W}_{1,j} \cdot \mathbf{u}'_{g,j}(t)) \tag{18}$$

$$\hat{\mathbf{u}}'_{g,j}(t) = \mathbf{W}_{4,j} \cdot \tanh(\mathbf{W}_{3,j} \cdot \mathbf{x}_j), \tag{19}$$

where $\mathbf{W}_{1,j} \in R^{n_{1,j} \times N}$, $\mathbf{W}_{2,j} \in R^{n_{2,j} \times n_{1,j}}$, $\mathbf{W}_{3,j} \in R^{n_{1,j} \times n_{2,j}}$, $\mathbf{W}_{4,j} \in R^{N \times n_{1,j}}$, $\mathbf{x}_j$ are the $n_{2,j}$

nonlinear PCA components for the jth velocity component, the nonlinear equivalent of the n_j components for the linear POD.

3.2. Linear and Nonlinear PCA for Homogeneous Flow

In the case of a flow that is homogeneous in the streamwise and spanwise directions the eigenfunctions will have a harmonic dependence in the horizontal plane. This means that the optimal eigenfunctions wall parallel components can be computed by means of a Fourier transform in that plane.

The normal components of the eigenfunctions can be obtained by performing a Karhunen–Loeve decomposition on the most significant Fourier modes. These modes are contained in the field $\tilde{\mathbf{u}}'(y, m, n)$, which is the Fourier transform of $\mathbf{u}'(\mathbf{x}, t)$ in the homogeneous spanwise and streamwise directions.

For each wave number index pair (m, n) a separate POD has to be performed in order to find out which are the most significant modes in the normal directions. It is convenient to introduce here another set of eigenfunctions $\psi(y, m, n)$. These eigenfunctions and the corresponding eigenvalues $\lambda(m, n)$ can be obtained by solving the equation

$$\int \langle \tilde{u}_i'(y, m, n)\tilde{u}_j'^{*}(y', m, n)\rangle \psi(y', m, n)\, \mathrm{d}y' = \lambda(m, n)\psi(y, m, n). \tag{20}$$

Here the averaging is performed over all the samples; the asterisk means complex conjugation and the indices $i, j = 1, 2, 3$ denote the three components of the vector field. Each three-dimensional eigenfunction for the full flow field can be obtained by combining the ψ's with the Fourier sinusoidal components in the horizontal plane. By calling L_x, L_z the streamwise and spanwise dimensions of the channel, an eigenfunction is expressed as

$$\Phi(\mathbf{x}, m, n) = \psi(y, m, n)e^{2\pi imx/L_x}e^{2\pi inz/L_z}, \tag{21}$$

which is a complex valued vector field. To perform the POD in the y direction also, for each wave number index pair (m, n) a linear POD must be performed on $\psi(y, m, n)$.

Due to the homogeneity hypothesis the dependence on the wave numbers (m, n) is known to be harmonic in the horizontal plane, as explicitly noted in Eq. (21); therefore only the dependence on the wall-normal direction needs to be transformed. This is the reason the functions $\psi(y, m, n)$ are treated separately.

Therefore for each (m, n) a set of one-dimensional eigenfunctions is obtained, containing the Karhunen–Loeve decomposition of $\psi(y, m, n)$ in the y direction.

Using a quantum number q to specify the index of an eigenfunction corresponding to a particular (m, n) index pair, the triplet $\mathbf{k} = (q, m, n)$ completely determines a particular eigenfunction.

Furthermore it must be noted that the eigenfunctions are degenerate; the contribution of a single eigenfunction to the flow reconstruction can be written as

$$\begin{aligned}\mathbf{u}'^{(q,m,n)}(\mathbf{x}, t) = {} & a^{(q,m,n)}(t)\Phi^{(q,m,n)}(\mathbf{x}) + a^{(q,m,-n)}(t)\Phi^{(q,m,-n)}(\mathbf{x}) \\ & + a^{(q,-m,n)}(t)\Phi^{(q,-m,n)}(\mathbf{x}) + a^{(q,-m,-n)}(t)\Phi^{(q,-m,-n)}(\mathbf{x}),\end{aligned}$$

where the four eigenvalues are all the same and the eigenfunctions are complex conjugate pairs. The flow can be reconstructed by summation of all the contributions on all the wave number indices and quantum numbers.

Also in this case it is possible to generalize the linear POD by using nonlinear neural networks to compress the wall-normal components of the eigenfunctions $\psi(y, m, n)$, treating these functions separately. The transformation on the whole velocity field can be viewed as a linear POD performed in the horizontal planes and a *decoupled* nonlinear POD performed in the wall-normal direction. This decoupling is possible due to the homogeneity hypothesis, which allows us to express the eigenfunctions $\Phi(\mathbf{x}, m, n)$ as a product of three terms, each one dependent on one coordinate only (see Eq. (21).

Therefore also in this case the compression can be performed by considering each wave number index pair separately and collecting all the transformed samples corresponding to each (m, n). This gives us $N_x \times N_z/4$ separated training sets indexed by (m, n).

To perform the nonlinear POD a total of $N_x \times N_z/4$ neural networks can be trained, one for each index pair (m, n), in the same way as that described earlier for the nonhomogeneous case.

Here a preliminary analysis of the linear POD eigenvalues can identify the most energetic modes in order to train the neural networks only on these. In this way the number of neural networks to train can be greatly reduced.

It is also important to notice that this problem is trivially parallel: all the neural networks are trained independently at the same time.

3.3. Results

We present here results from the application of POD and nonlinear NN as reconstruction tools for the randomly forced Burgers equation and for the near wall flow field in a turbulent channel flow at $\mathrm{Re} = 2500$ based on the channel half height.

3.3.1. *Randomly Forced Burgers Equation*

To assess the capabilities of the nonlinear PCA vs the POD, we consider first its application to Burgers equation.

The randomly forced Burgers equation is known to generate solutions showing turbulentlike fluctuations [11]. It has been used as a benchmark problem for testing modeling and control techniques using neural networks [24], and for testing the POD [11].

The equation is expressed as

$$\begin{aligned} \frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2} + \chi(x, t), \quad 0 < x < 1, \\ u(0, t) &= 0, \quad u(1, t) = 0 \\ \langle \chi \rangle_x &= 0, \quad \langle \chi \rangle_x{}^2 = 1, \end{aligned} \tag{22}$$

where $\chi(x, t)$ is the random forcing, and ν is the viscosity coefficient. The numerical method used to solve this equation uses the implicit Crank–Nicolson time discretization and a second-order space differencing. The equation was solved numerically for $\nu = 10^{-3}$, in a periodic domain of [0, 1] on a grid of 2048 points in space and a nondimensional time step of 0.001. The stochastic forcing was a random Gaussian white process with variance 0.1, updated at every 100 time steps and held constant in between.

The linear POD was performed using 10,000 samples of the flow. It has been found that $n = 7$ linear POD components must be retained in order to retain 90% of the energy for

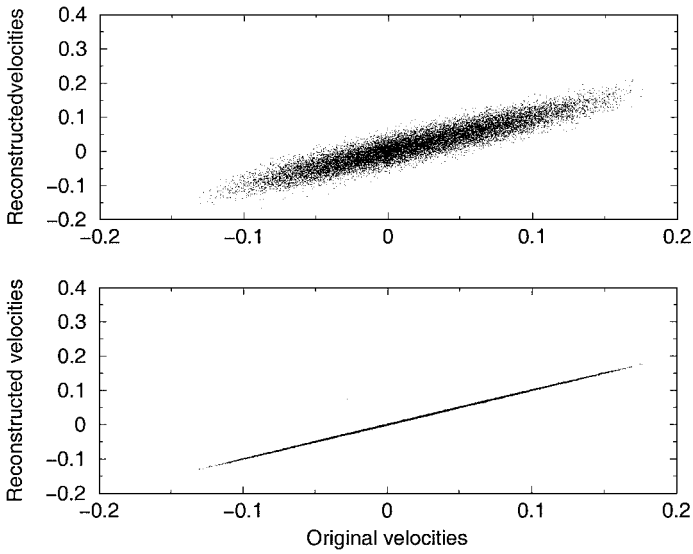

FIG. 3. Comparison between the linear (top) and nonlinear (bottom) POD performance on the randomly forced Burgers equation data.

this case. A nonlinear PCA neural network was trained to perform an identity mapping, with $n_1 = n_3 = 32$ and $n_2 = n = 7$. In Fig. 3, a performance comparison between the two techniques, on a data set of 1000 vectors not used during the training phase by either of the two algorithms, is reported. The nonlinear PCA structure is much more efficient in compressing these data, using the same number of components as that of the linear POD. This result demonstrates the generalization capabilities of both methods. We remark here that the fact that the nonlinear NN performs better than the linear PCA is not in contrast with the optimality of the linear PCA itself, since the linear PCA is the optimal *linear* decomposition.

As remarked earlier the nonlinear PCA is performed by training a NN using back propagation, which is an iterative method; in contrast the linear POD can be efficiently performed by solving the underlying eigenproblem by means of singular value decomposition, which is not an iterative algorithm [15]. This is reflected in the fact that the time required for the nonlinear PCA to converge is much larger than the time required to perform a linear POD. This problem becomes more marked when the number of dimensions to deal with increases.

The effect of the size of the nonlinear layers was examined by performing the nonlinear PCA using different values for the nonlinear layers size $n_1 = n_3$, while keeping fixed the size of the inner layer to $n_2 = 7$. The results are reported in Fig. 4, where it can be noted that the quality of the approximation improves when the hidden layer size is increased; in particular the reconstruction performance degrades significantly when $n_1 < n_2$, suggesting that this condition should be avoided.

3.3.2. *Turbulent Channel Flow*

The study of fully developed turbulent channel flow is a benchmark problem for theoretical and experimental studies of near wall turbulence [21]. In this study we consider flow

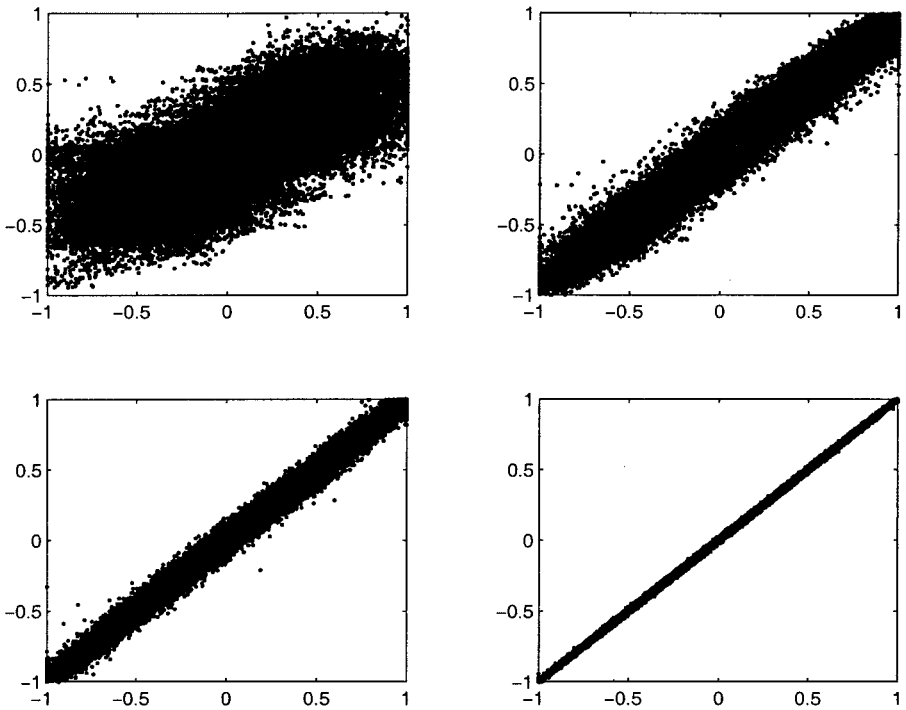

FIG. 4. Performance of the nonlinear POD on the randomly forced Burgers equation data, for different values of the nonlinear layer size $n^1 = n^2$. From top to bottom, clockwise: original versus reconstructed normalized data, $n^1 = 3$, $n^1 = 7$, $n^1 = 32$, $n^1 = 16$.

quantities obtained from a DNS of an incompressible turbulent flow, in a constant mass flow rate regime. The governing Navier–Stokes equations are expressed as

$$\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i} = -\frac{\partial P}{\partial x_j} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_j}{\partial x_j^2} \tag{23}$$

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{24}$$

where the Reynolds number is 2500 based on the channel half height (h), the bulk velocity (U_b), and the fluid kinematic viscosity ν ($\mathrm{Re} = U_b h/\nu$). The DNS is performed in a box of size $(L_x, L_y, L_z) = (2\pi h, 2h, 3/2\pi h)$. A mesh of $(129 \times 129 \times 65)$ points in the (x, y, z) directions was used; the mesh has a cosine spacing in the y direction. The numerical scheme employs for the spatial discretization Fourier series in the streamwise and spanwise directions and Chebychev polynomial expansion in the wall-normal direction; the time advancement scheme is Crank–Nicolson for the viscous terms and Adams–Bashfort for the nonlinear convective terms using a nondimensional time step of $\delta t^+ = 0.0025$. The numerical scheme used in this simulation is further described in [21].

In this paper we consider the reconstruction of the near wall region of the flow within $y^+ = 0$ and $y^+ = 40$. To reduce the computational cost for the POD and nonlinear PCA, this region was further subdivided into 12 adjacent subregions, each one of size $(L_x'^+, L_y'^+, L_z'^+) = (353.4, 40, 208.5)$. Each of these subregions contains one minimal flow unit [15]. The size

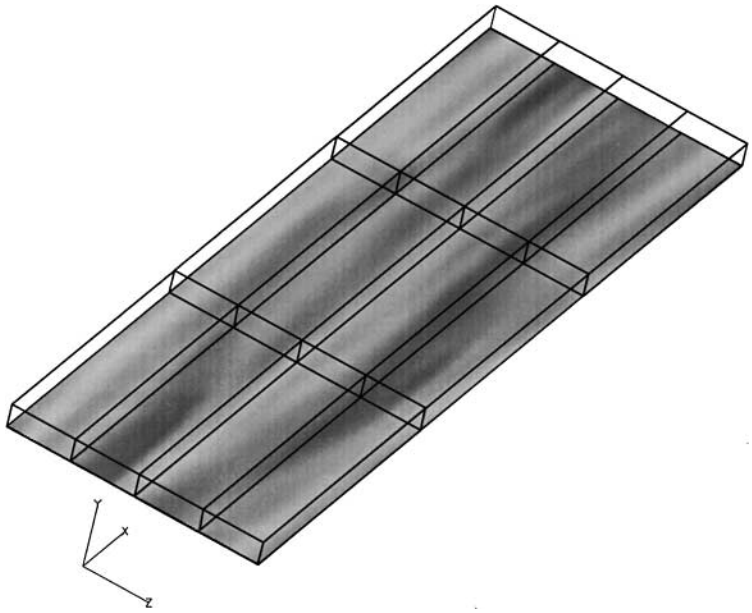

FIG. 5. Subdivision of the wall region in 12 modeling subregions, with streamwise shear stresses on the wall. The wall region extends until $y^+ = 40$.

of each modeling region (MR) in grid points is $(N_x, N_y, N_z) = (42, 40, 16)$. In Fig. 5 this subdivision is shown.

In the DNS the full size channel is simulated, and the POD and nonlinear PCA models are derived for a MR; the entire wall region is then reconstructed by tiling together the reconstructed MRs. This approach allows us to reduce the computational power needed for the linear POD and nonlinear PCA by considering only the smallest significant region of the channel to generate the eigenfunctions. At the same time since the full size channel is simulated it is ensured that all the higher order modes are present in the data used to generate the eigenfunctions. The data to which the linear POD and nonlinear PCA have been applied are the near wall velocities in a MR. We consider first a compression task, as for the Burgers equation, in which the near wall region is represented with a shorter vector containing the principal components. After the compression problem we address the modeling problem, in which it is assumed that only quantities in the wall region are accessible.

In the modeling problem only the streamwise and spanwise velocities are reconstructed, and the wall-normal velocity is derived by applying the continuity equation to the reconstructed components; in the compression problem all the three velocities are reconstructed independently by using three different models. In the case of the POD, a model is simply a set of eigenfunctions, while for the nonlinear PCA a model is a nonlinear neural network trained on the available data. We point out here that when we reference the number of components for the linear POD we imply the number of terms kept in the finite truncation of representation (2), while the number of components of the nonlinear PCA is the number of linear neurons contained in the hidden linear layer, i.e., n_2 neurons.

To generate the input data necessary for all the experiments performed 100 samples of the near wall region have been considered. To ensure that the fields were to a large extent

uncorrelated, these samples are collected every three flow through times for the MR (that is, taking as reference quantities the length and the average streamwise velocity in a MR). Since a MR contains a minimal flow unit, the resulting samples are sufficiently uncorrelated for our purposes. The DNS started from a fully turbulent initial condition and lasted for 60,000 time steps, i.e., a nondimensional time of 150. Since the wall region has been divided into 12 MRs, the total number of samples produced in this way amounts to 1200; this number is doubled by considering also samples from the upper channel wall therefore yielding 2400 samples for the training.

3.4. Linear POD and Nonlinear PCA, Nonhomogeneous Flow

The performance of linear POD and nonlinear PCA has been compared in a 3-D reconstruction task using the samples produced by the DNS described above, without enforcing the spatial homogeneity hypothesis. A linear POD was performed first on the available data, yielding an eigenvalue spectrum for the samples. In this case by keeping the first 40 eigenfunctions per component of the POD 90% of the energy is retained for all the three velocity components. This number of components is the same for the linear POD and for the nonlinear PCA, in order to compare their performance.

It is not easy to choose a fair criterion to compare the performance of these two algorithms, because the nonlinear technique presented here is an iterative method, in contrast to the linear POD; for this reason, the nonlinear PCA requires at least one order of magnitude more CPU time to converge. Moreover, the total number of degrees of freedom is much larger for the nonlinear PCA, because of the two further layers inserted. In fact the advantage of the nonlinear PCA over the linear POD is in the possibility of adding more degrees of freedom without enlarging the number of components for the transformation. This is possible because the additional layers are nonlinear. For these reasons, it seems appropriate to compare performance achieved with the same number of components.

For the nonlinear PCA, three different NNs have been used, one for each velocity component. Each nonlinear NN has $n_{1,j} = n_{3,j} = 128$ neurons in each nonlinear layer. We note here that the number of neurons is an additional degree of freedom in the nonlinear case. In this first application we try to simply perform a *data compression*; i.e., the entire near wall field is taken as input and the POD and neural network models are used to implement an identity mapping. To verify that the NN model will produce a statistically meaningful fit, a necessary condition to check is that the number of points to fit (i.e., the number of velocity field points, N in our case) should be at least four or five times larger than the number of degrees of freedom (i.e., the total number of neural network weights) [23].

Each field sample is an input for the nonlinear neural network and can be viewed also as a vector of length $N = 42 \cdot 39 \cdot 16 = 26{,}208$. The total number of weights of each of the nonlinear neural networks used here is $N_{weights} = 2 \cdot (N \cdot n_{1,j} + n_{1,j} \cdot n_{2,j}) = 6{,}710{,}272$ weights, i.e., degrees of freedom; that is very probably one of the largest neural networks ever trained. The total number of points to fit using this model is equal to the number of model outputs multiplied by the total number of samples; i.e., $N \cdot 2400$. The ratio between the number of data points to fit vs the number of degrees of freedom is therefore, for each neural network, $(N \cdot 2400)/N_{weights} \approx 10$—that is an order of magnitude larger, providing a reasonable number of data for a statistically meaningful data fitting [19]. The NN training has been performed using an accelerated back-propagation algorithm [22]; the training

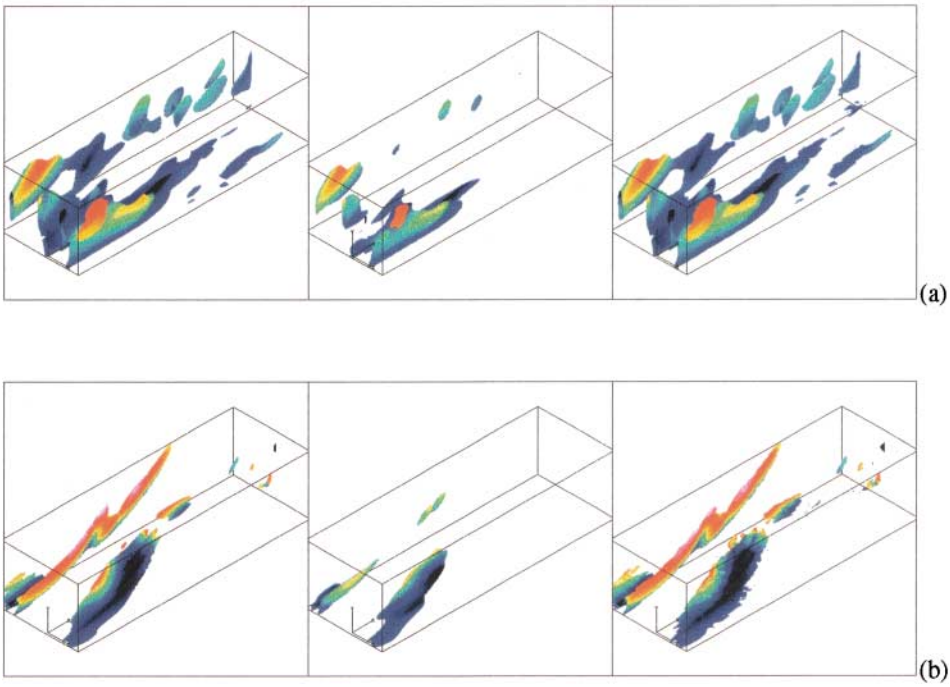

FIG. 6. Test samples used to test the reconstruction performance: vorticity magnitude isosurface, colored with the wall-normal component of the velocity. Only one quarter of the entire channel is shown. From left to right: original sample, linear reconstruction, and neural net reconstruction. (a) Test sample in which the neural net has a good performance. (b) Test sample in which the neural net has a worse performance.

lasted for 10,000 iterations. In Fig. 6 a qualitative visualization of the reconstruction performance is reported for two samples of a quarter of the wall region, not used to compute the POD eigenfunctions and to train the NN. These samples show structures formed by an isosurface of vorticity magnitude, colored with wall-normal velocity. The data reconstruction is obtained by applying the models to each of the 12 MR in the channel and then tiling the reconstructed MRs.

The nonlinear neural network performance is very good for a sample shown in Fig. 6a. In both the cases shown the linear POD reconstruction shows its typical "smoothing" effect on the flow field, due to the fact that the higher energy modes are neglected.

The neural network on the other hand performs a nonlinear transformation on the inputs before the linear PCA stage contained in the second hidden layer, as sketched in Fig. 2. In this way the information contained in the higher order modes is combined with that in the lower order modes, thus allowing a more detailed reconstruction. However, this nonlinear transformation may, in turn, introduce a distortion, as shown in Fig. 6b for another sample case. The two cases shown here are representative of high and low extremes in the neural network reconstruction accuracy.

To compare the two models also quantitatively, we calculate the correlation coefficients of the velocity components from the DNS and the reconstructed fields. Note that the DNS has been started from a different initial condition with respect to the initial condition used to generate the samples. The correlation coefficients are defined as

$$r(t) = \frac{\mathbf{u}_g(t) \cdot \hat{\mathbf{u}}_g(t)}{\|\mathbf{u}_g(t)\| \|\hat{\mathbf{u}}_g(t)\|}, \tag{25}$$

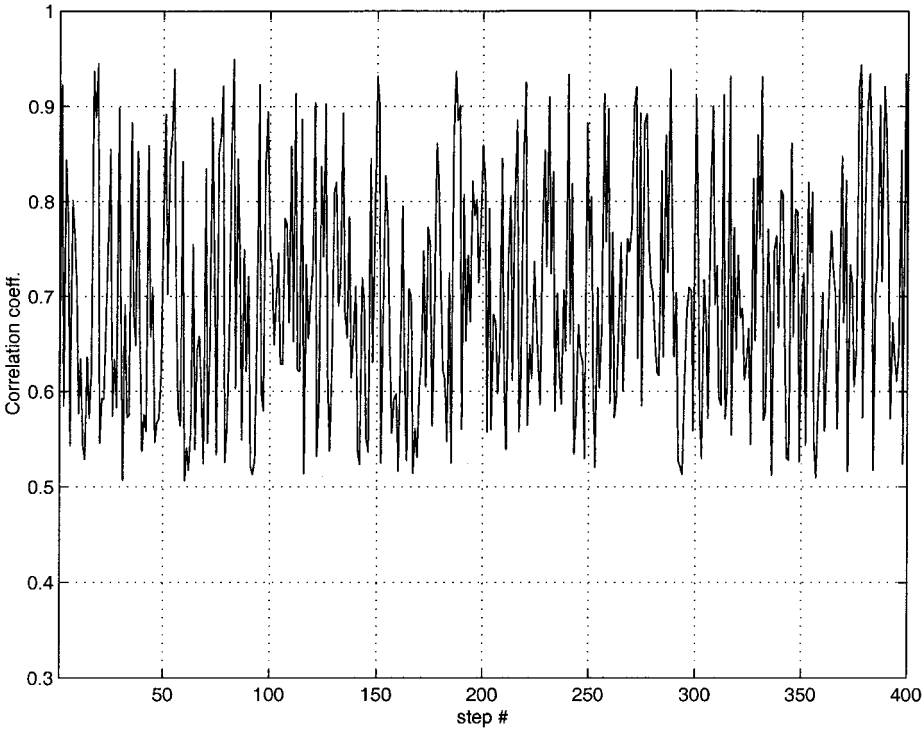

FIG. 7. Nonlinear reconstruction, DNS/reconstructed flow field correlation coefficient as a function of time, for a set of samples not used for training.

where u is the DNS velocity field and $\hat{u}$ is the reconstructed field and the dot indicates inner product. This equation expresses the correlation between two instantaneous flow fields. This quantity is useful in checking that the neural network model is effective also on samples not used for its training. We report this correlation for the neural network reconstruction in Fig. 7, where it is possible to see the neural network's performance as a function of time.

Another interesting quantity is the time averaged correlation

$$r_j(y) = \left\langle \frac{\hat{\mathbf{u}}_{g,j}(y,t) \cdot \mathbf{u}_{g,j}(y,t)}{\|\hat{\mathbf{u}}_{g,j}(y,t)\| \|\mathbf{u}_{g,j}(y,t)\|} \right\rangle, \tag{26}$$

where $\mathbf{u}_{g,j}(y_g,t) = u_j(\mathbf{x}_g,t)|_{y_g}$ are vectors containing the components of the DNS velocity field measured on horizontal MR planes defined by constant y, $\hat{\mathbf{u}}_{g,j}(y,t)$ contain the reconstructed velocity field, and the angle brackets indicate averaging both in time and over horizontal MR planes. This allows us to evaluate the model performance as a function of the distance from the channel wall.

The time averaging period was chosen to be sufficiently large to yield significant results; in the case of the correlations coefficient estimation, this amounts to the addition of one more sample to the average which does not change it beyond a small threshold. We found out that an averaging period of about three channel flow through times yielded correlation coefficients that are modified by less than 1% by the addition of further samples, so we chose that time as the averaging period for all the correlation coefficients.

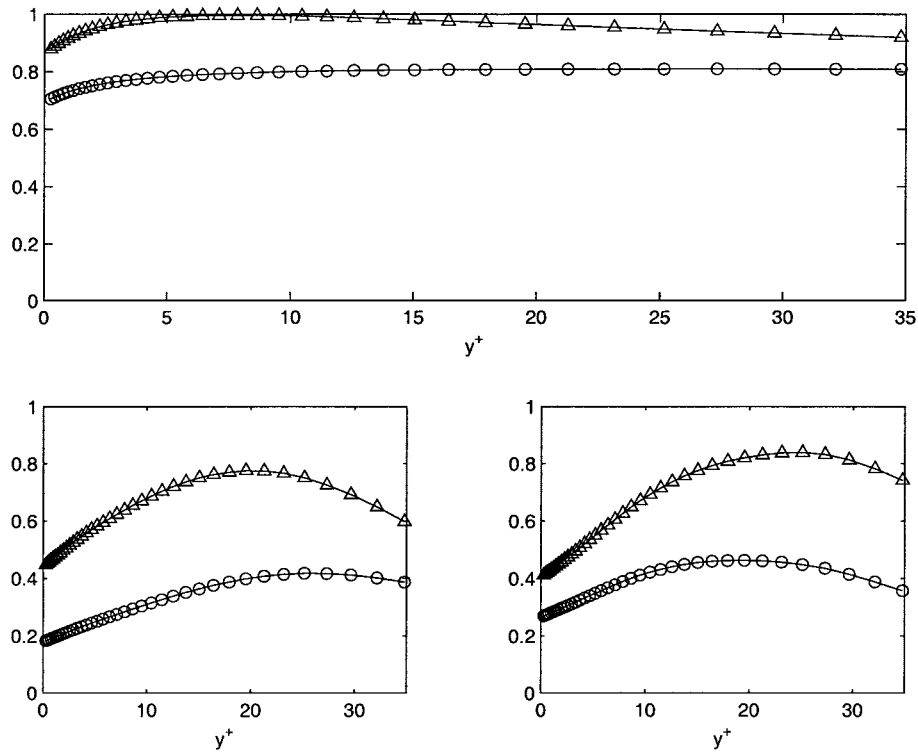

FIG. 8. Nonhomogeneous flow, comparison between the reconstruction performance, correlation coefficients. Circles, linear POD; triangles, nonlinear PCA reconstruction; top, streamwise component; bottom left, normal component; bottom right, spanwise component.

In Fig. 8 the correlation coefficients for the three components of the velocity field are reported. The POD performs well for the streamwise component reconstruction, with performance close to that of the NN PCA reconstruction. For the other two components the nonlinear PCA reconstruction is significantly better than that of the POD.

The much better correlation obtained in both cases for the streamwise component reconstruction with respect to the other two components is expected, as it is due to the strong inhomogeneity of the flow in the streamwise direction, a feature that is very well captured by both the compression methods. However, the NN outperforms the POD in the reconstruction of the wall-normal and the spanwise velocity components.

3.5. Linear POD and Nonlinear PCA, Homogeneous Flow

The samples produced by the DNS performed earlier have been used to compare the performance of linear POD and nonlinear PCA by exploiting the homogeneity property of the flow. In this case it is not possible to use the same samples as those used earlier, because the periodic boundary conditions are enforced on the whole channel. Therefore in the present case a sample consists in the velocity field in the whole channel, until $y^+ = 40$.

A linear POD has been performed on the Fourier transform of these data, yielding a total of 124 eigenfunctions to keep 90% of the flow energy.

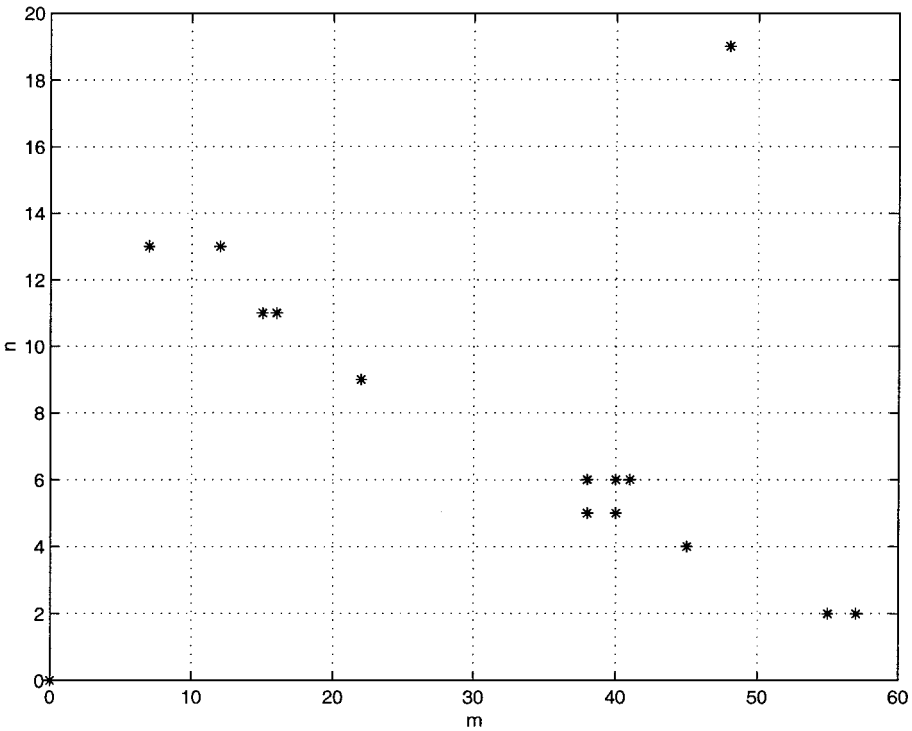

FIG. 9. Indices of the Fourier eigenmodes modeled by the nonlinear PCA. Each eigenmode for which both (m, n) are nonzero is 4-degenerate.

The modes actually present in the significant eigenfunctions, indexed by the wave number index pairs (m, n), are reported in Fig. 9. It is possible to note that only 15 modes are present, meaning that 15 different neural networks will have to be trained on the corresponding data.

Each neural network has a number of inputs equal to $N_{in} = 40 \times 3 \times 2 = 240$, because the components of the transformed field are complex numbers.

For each nonlinear PCA neural network used an architecture composed of $n^1 = n^3 = 240$ neurons in the hidden layers and $n^2 = 8$ neurons in the linear layer that contains the nonlinear PCA components.

The total number of weights per network in the present case is about 120,000, much smaller than the number of weights per network in the nonhomogeneous case. In this case the correlations between components are also taken into account, because all the field components are present in the input vector. On the other hand several different networks have to be trained at the same time and in order to reduce this number only the most representative modes in the linear POD case are considered for the training. The 15 nonlinear neural networks were trained in parallel on a Beowulf cluster, using a different processor for each neural network. This training procedure is embarassingly parallel thus reducing the computational time by a factor of 15.

The correlation coefficients for the linear POD and the nonlinear PCA with the DNS velocity fields are presented in Fig. 10. Also in this case the nonlinear PCA performs better than the linear POD, and the overall performance is better with respect to the nonhomogeneous case. This is due to the fact that with the homogeneity hypothesis enforced it has been possible to take into account the correlations between the velocity field components.

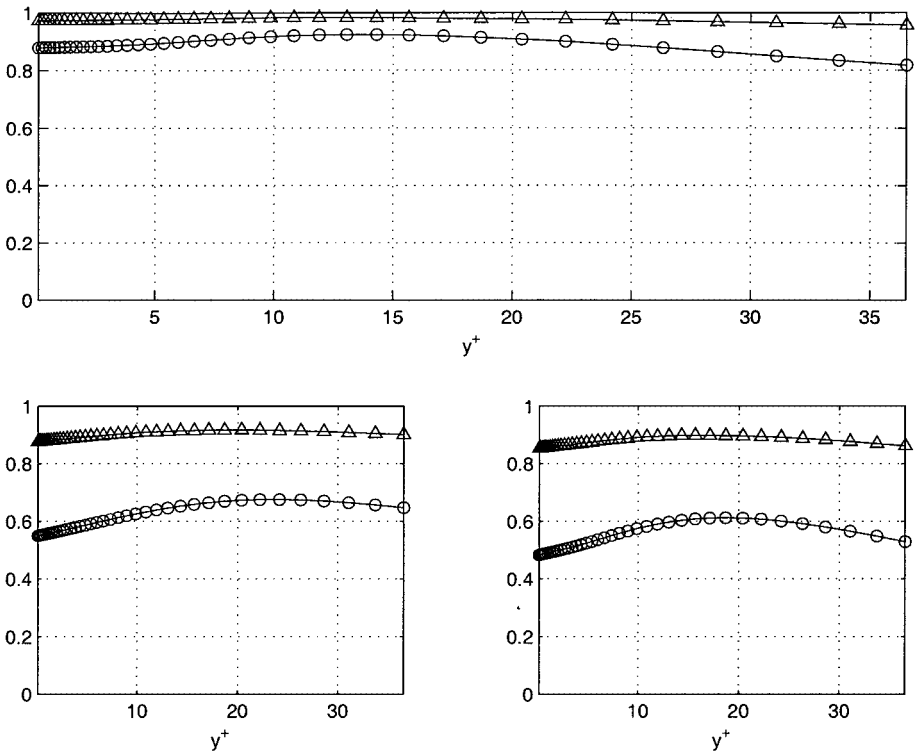

FIG. 10. Homogeneous flow, comparison between the reconstruction performance, correlation coefficients. Circles, linear POD; triangles, nonlinear PCA reconstruction; top, streamwise component; bottom left, normal component; bottom right, spanwise component.

4. NEAR WALL RECONSTRUCTION MODELS

The architectures presented in the previous sections can be viewed as compression models for the near wall flow field. They require the entire field as input, producing a compressed version described by the POD or nonlinear PCA components. Reconstruction of the flow field can be achieved using as input the compressed field. In the context of flow control it is desirable to reconstruct the flow field in the near wall region using wall only information such as wall pressure and shear stresses.

In what follows it will be outlined how this task is accomplished by using the linear POD [13] and a simple neural network.

4.1. Near Wall Reconstruction Using Wall Only Data

In the context of the POD it is possible to reconstruct the flow field using only the shear stresses. This is facilitated by the decoupling of the space and time information that is employed by the POD. More specifically the wall shear stresses can be expressed as

$$\left.\frac{\partial u_j(\mathbf{x},t)}{\partial y}\right|_{wall} = \sum_{i=1}^{n_j} a_{i,j}(t)\cdot \left.\frac{\partial \phi_{i,j}(\mathbf{x})}{\partial y}\right|_{wall}, \quad j\neq 2, \tag{27}$$

where $\phi_{i,j}(\mathbf{x})$ are the n_j POD eigenfunctions. The only unknowns in these equations are the eigenvalues $a_{i,j}$, which can be easily computed and used to reconstruct the original streamwise and spanwise components. Here the streamwise and spanwise shear stresses are used to reconstruct respectively the streamwise and spanwise components of the velocity field. The normal component can be reconstructed by using the continuity equation and the approximated streamwise and spanwise components.

This reconstruction is not so immediate in the nonlinear case, because in the nonlinear PCA there is no decoupling between space and time information. In the case of a nonlinear PCA NN, this is not possible. However, similar results can be obtained by using a nonlinear NN, training it using the results of a DNS. There is an advantage in this approach as the available inputs are not limited to the shear stresses only but it is possible to feed the pressure also into the network. This neural network structure is not anymore a nonlinear PCA structure, as it does not perform an identity mapping. The overall structure is

$$\mathbf{x}_j = \mathbf{W}_{2,j} \cdot \tanh(\mathbf{W}_{1,j} \cdot \mathbf{v}) \tag{28}$$

$$\hat{\mathbf{u}}'_{g,j}(t) = \mathbf{W}_{4,j} \cdot \tanh(\mathbf{W}_{3,j} \cdot \mathbf{x}_j), \tag{29}$$

where $\mathbf{v}$ is a vector containing the shear stresses and the pressure at the wall, $W_{1,j} \in R^{n_1 \times n_\mathbf{v}}$, $W_{2,j} \in R^{n_2 \times n_1}$, $W_{3,j} \in R^{n_1 \times n_2}$, $W_{4,j} \in R^{N \times n_1}$.

This model can be improved by observing that the knowledge of the shear stresses and the pressure at the wall allows us to set up a second-order model for the near wall flow. In what follows, a w as a subscript indicates a wall-measured quantity; the second-order model is

$$u_j(\mathbf{x}, t) = \omega_{j,w} y + \frac{\mathrm{Re}}{2} \frac{\partial P_w}{\partial x_j} y^2 + \mathcal{O}(y^3), \quad j = 1, 3, \tag{30}$$

where $\omega_{j,w}$, $j = 1, 3$ are the wall tangential vorticity components and P_w is the wall pressure. A neural network can be used to approximate the higher order terms using wall quantities

$$u_j(\mathbf{x}, t) = \omega_{j,w} y + \frac{\mathrm{Re}}{2} \frac{\partial P_w}{\partial x_j} y^2 + NN_j(\mathbf{v}_w), \quad j = 1, 3, \tag{31}$$

where $\mathbf{v}_w$ is a vector containing the wall shear stresses and pressure. The normal component can be reconstructed by substituting in the continuity equation the reconstructed streamwise and spanwise components.

4.2. Near Wall Modeling Results

The NN model has been applied to the turbulent channel flow described above. The near wall region was subdivided into 12 modeling regions, exactly as in the previous case and the reconstruction of the near wall has been obtained by tiling all regions together. The structure of the POD and of the NN is kept as that in the previous case; i.e., for the linear POD the first 40 eigenfunctions were considered for the streamwise and spanwise components. Two NNs are used in order to generate the correction terms for the second-order models for the streamwise and spanwise components. Each NN takes the wall streamwise and spanwise shear stresses and pressure as inputs, i.e., a total of $42 \times 16 \times 3 = 2016$ inputs. There is a hidden nonlinear layer with 128 neurons, then an inner linear layer with

40 neurons, representing the nonlinear NN analogous of the 40 eigenfunction coefficients retained in the linear representation; after the linear layer another nonlinear layer with 128 neurons follows, and finally the NN output layer which produces directly a correction term for the second-order model. The number of outputs is $42 \times 39 \times 16 = 26{,}208$. This NN differs from the nonlinear PCA NN only in the smaller number of inputs, that is smaller; thus the number of degrees of freedom is also smaller. On the other hand the number of samples used in the training is the same; therefore the necessary condition for a statistically meaningful fit is satisfied also in this case. The NN is trained using the same accelerated back-propagation algorithm mentioned earlier [22]. The samples used for the training are the same as those already used for the nonlinear PCA neural network. The NN training is performed online, in parallel with the DNS of the flow. The time for the back propagation of a sample is a small fraction of the simulation time, corresponding to the DNS.

In Fig. 11 we present a sample of the streamwise and spanwise averaged u^+ profile, comparing a POD reconstruction and a nonlinear model reconstruction. This sample results from a simulation using samples that were not used in the NN training phase and in the computation of the POD eigenfunctions, thus showing the good generalization of this model. In Fig. 12 the correlation coefficients for the three velocity components of the DNS flow and the reconstructed flow as defined earlier are reported. It is clear that the correlation deteriorates when the distance from the wall increases, indicating progressive inefficiency of wall only information to reconstruct the whole flow field. This is confirmed by the

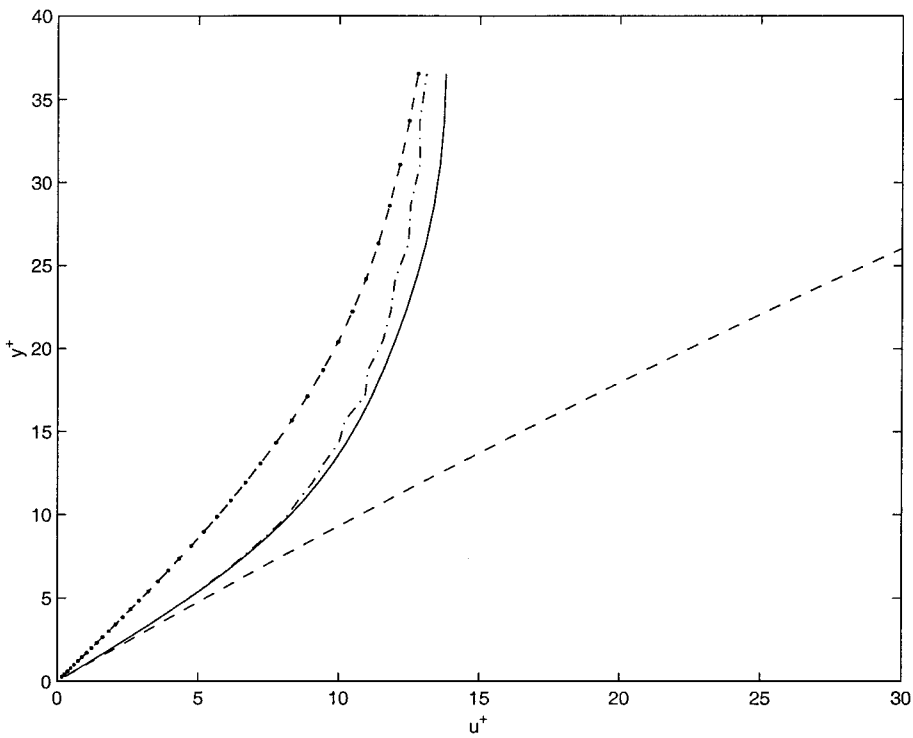

FIG. 11. Comparison of the reconstruction performance, streamwise and spanwise averaged profile. Continuous: original, dash-point: reconstruction using linear POD, dash-dot: reconstruction using NN, dashed: reconstruction with second-order model only.

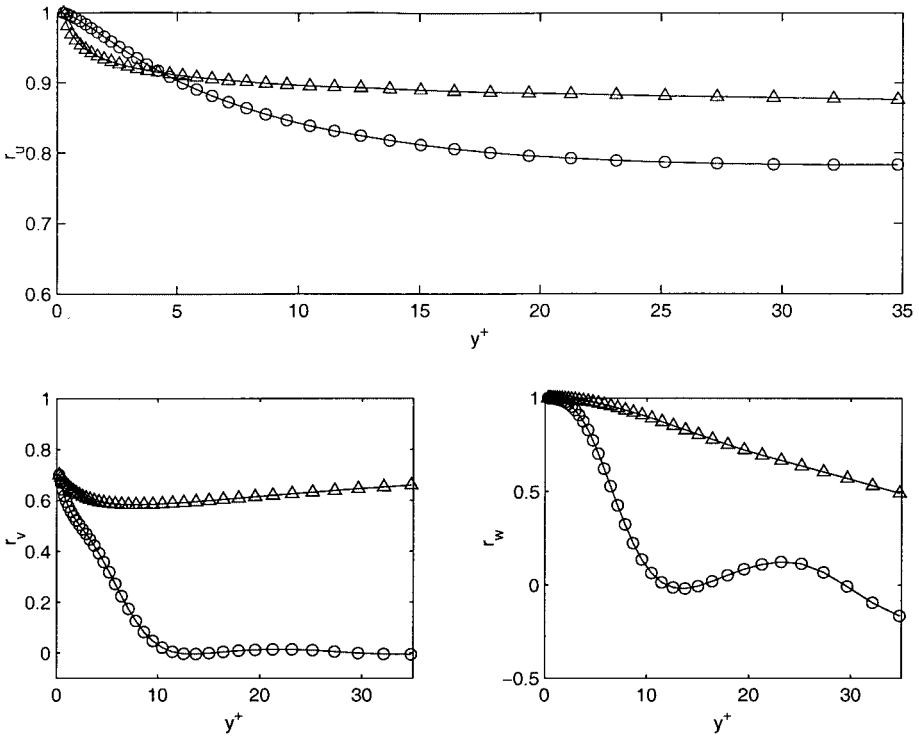

FIG. 12. Comparison between the modeling performance, correlation coefficients. Circles, linear POD; triangles, nonlinear PCA reconstruction; top, streamwise component; bottom left, normal component; bottom right, spanwise component.

comparison of the tangential and normal components of the Reynolds stress tensor, shown in Fig. 13.

Furthermore, the better correlation obtained by the linear POD reconstruction in the viscous sublayer region (i.e., $y^+ < 5$) for the streamwise and spanwise components, suggests that a linear model can approximate in a satisfactory way the flow in that region. The nonlinear NN exhibits in this region signs of overparameterization. However, at distances outside

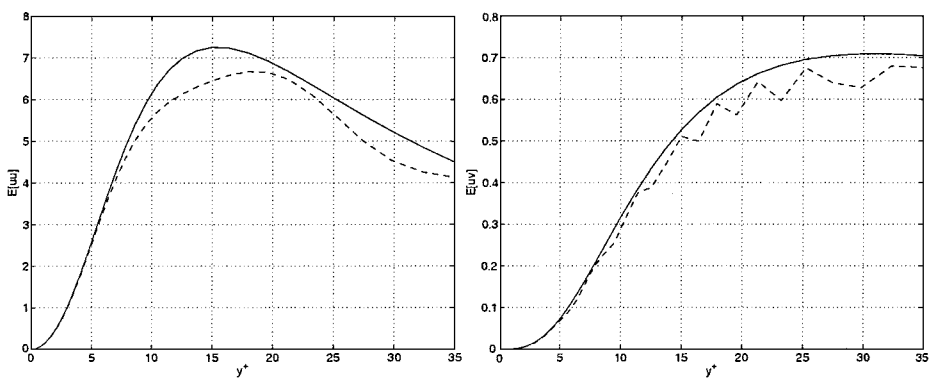

FIG. 13. Comparison between the modeling performance, components of the Reynolds stress tensor. Left, normal components; right, tangential components. Continuous line, DNS; dashed line, NN-reconstructed field.

the viscous sublayer the NN clearly outperforms the POD. In the case of the spanwise and normal components the overall reconstruction performance is poorer than the streamwise component reconstruction, as expected due to inhomogeneity as outlined earlier for the comparison between the nonlinear PCA and POD.

4.3. Nonlinear Neural Network Correction Only above the Viscous Sublayer

The results in the previous section indicate that the linear POD model performs better than the nonlinear neural network in the "linear" viscous sublayer region. To use this result to improve the neural network structure, we remark that the neural network used to correct the second-order model is not trained to perform an identity mapping. Hence the network does not perform a PCA but it simply implements a nonlinear mapping.

The first consideration implies that, in order to improve the model performance, one can use the correction term only above the viscous sublayer, modeling the flow in $y^+ < 5$ with the second-order model only. The second consideration means that since we do not have to implement a nonlinear PCA, the nonlinear neural network does not need to have two hidden nonlinear layers in order to implement a nonlinear mapping [19, 20].

For these reasons, the structure of the nonlinear neural network used to approximate the correction term can be greatly simplified by eliminating all the correction in the viscous sublayer and by using two layers only (an input and an output), instead of four layers. The new simplified correction term will therefore have the structure

$$NN_j(\mathbf{v}_w) = \mathbf{W}_{2,j} \tanh(\mathbf{W}_{1,j} \cdot \mathbf{v}_w), \quad j \neq 2, \tag{32}$$

where $\mathbf{W}_{1,j} \in R^{n_{1,j} \times N_1}$, $W_{2,j} \in R^{N_2 \times n_{1,j}}$; N_1 and N_2 are the total number of input vector components and output vector components, respectively.

The number of grid points in the viscous sublayer for the configuration used in this paper is $N_{sub} = 42 \times 17 \times 16 = 11{,}424$, which are computed not considering the wall at $y^+ = 0$. The number of outputs of this nonlinear NN correcting only the flow above the viscous sublayer is $N_{tot} - N_{sub} = 14784$, i.e., a bit more than half of N_{tot}.

In this case the goal is not to achieve a compression of the information, i.e., to represent the flow itself with the smallest possible number of components; this means that we have the freedom to choose the number of hidden neurons $n_{1,j}$ by looking only at the corrector performance and not aiming at using the least number of components to represent the flow. In the present case, in order to compare fairly the nonlinear correction model with the linear POD reconstruction system, it has been chosen to compare the two models at equal degrees of freedom, i.e., the total number of nonlinear NN weights has been set equal to the total number of components of the four eigenfunctions used for the linear POD of the near wall flow. As one weight implies a computational cost of one multiplication for both the models considered, it can also be viewed as an "equal computational cost" approach. The "hybrid" network is shown to perform better overall, by fully taking advantage of the linearity in the viscous sublayer while away from it, it recovers the advantage of the nonlinear neural network representation. The performance of the hybrid model is practically the same as that of the "complete" model shown earlier; in Fig. 14 a comparison between the correlation coefficients of the linear POD model and the "hybrid" model is reported for the streamwise direction component. The two curves are obviously identical in $y^+ < 5$, i.e., the region where the model used is the same. This result shows that it is possible to take advantage of the

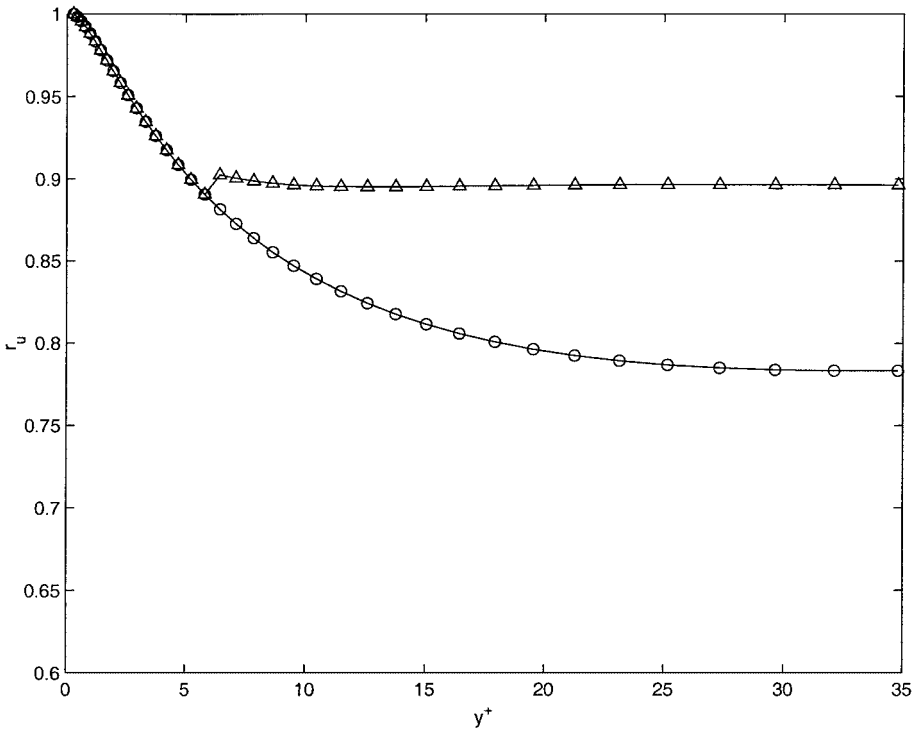

FIG. 14. Comparison between the reconstruction performance for the simplified model and the linear POD model, correlation coefficients for the streamwise component. Circles, linear POD; triangles, nonlinear PCA reconstruction.

linear POD reconstruction methodology to greatly reduce the model size and computational cost, which as specified earlier is proportional to the number of neural network weights.

5. CONCLUSIONS

We have presented a nonlinear neural network model for the reconstruction of the near wall flow in a turbulent channel flow. The linear POD is shown to be a subset of a more general family of nonlinear transformations, implemented by a nonlinear neural network. The performance of the nonlinear model compares favorably to the performance of a linear POD model performing the same task.

The neural network model provides better compression capabilities than the POD as it results in better reconstruction for data in which it has not been trained, both in the case in which spatial homogeneity hypotheses are enforced and in the more general case in which no such hypotheses are made.

On the other hand the drawbacks of this approach are that the decoupling between space and time that is present in the linear case is lost, and also the computational time to train the nonlinear PCA structure can be larger than the time needed to construct the linear POD eigenfunctions, since the multilayered feed-forward neural network training is an iterative process. However, the additional cost, as compared to the POD, induced by the NN training is compensated by the highly parallel character of the training process. We plan to improve further this basic model by experimenting with more complex structures, such as time

delayed and/or recurrent neural networks, in order to take time-dependent information also into account.

The neural networks were also implemented in order to provide reconstruction of the flow field using wall only information, a desirable situation for problems of flow control. The results have shown that a straightforward application of neural networks may not be advantageous for flow reconstruction in the viscous sublayer. An improved neural network architecture is developed that takes advantage of this linearity while maintaining the advantages of the nonlinear approach.

In turbulent flows accurate reconstruction of the near wall region is essential to devise successful control schemes [13]. The positive results presented in this paper for near wall modeling with larger nonlinear neural networks, assuming wall only information, encourage experimentation on related control schemes.

Looking ahead, we envision that neural network models can also be useful in the construction of near wall models for flow solvers using RANS or LES, by exploiting their compression capabilities. Work is under way to develop such models by further exploiting large scale numerical and experimental databases.

REFERENCES

1. S. K. Robinson, Coherent motions in the turbulent boundary layer, *Annu. Rev. Fluid Mech.* **23**, 601 (1991).
2. H. T. Kim, S. J. Kline, and W. C. Reynolds, The production of turbulent near a smooth wall in a turbulent boundary layer, *J. Fluid Mech.* **50**, 133 (1971).
3. G. Berkooz, P. Holmes, and J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, *Annu. Rev. Fluid Mech.* **25**, 239 (1993).
4. D. Ruelle and F. Takens, On the nature of turbulence, *Comm. Math. Phys.* **20**, 167 (1971).
5. J. L. Lumley, The structure of inhomogeneous turbulent flows, in *Atmospheric Turbulence and Radio Wave Propagation*, edited by A. M. Yaglow and V. I. Tatarski (Nauka, Moscow, 1967).
6. S. Y. Shvatrsman and I. G. Kevrekidis, Nonlinear model reduction for control of distributed systems: A computer assisted study, *AICHE J.* **44**(7), 1579 (1998).
7. L. Sirovich, Turbulence and the dynamics of coherent structures, Part I: Coherent structures, *Quart. Appl. Math.* **45**(3), 561 (1987).
8. L. Sirovich, Turbulence and the dynamics of coherent structures, Part III: Dynamics and scaling, *Quart. Appl. Math.* **45**(3), 583 (1987).
9. C. H. Sirovich and L. Sirovich, Low dimensional description of complicated phenomena, *Cont. Math.* **99**, 277 (1989).
10. N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, The dynamics of coherent structures in the wall region of the wall boundary layer, *J. Fluid Mech.* **192**, 115 (1988).
11. D. H. Chambers, R. J. Adrian, P. Moin, D. S. Stewart, and H. J. Sung, Karhunen–Loeve expansion of Burgers' model of turbulence, *Phys. Fluids* **31**, 2573 (1988).
12. J. Jimenez and P. Moin, The minimal flow unit in near-wall turbulence, *J. Fluid Mech.* **225**, 213 (1991).
13. B. Podvin and J. Lumley, Reconstructing the flow in the wall region from wall sensors, *Phys. Fluids* **10**, 1182 (1998).
14. B. Podvin and J. Lumley, A low-dimensional approach for the minimal flow unit, *J. Fluid Mech.* **362**, 121 (1998).
15. G. A. Webber, R. A. Handler, and L. Sirovich, The Karhunen–Loeve decomposition of minimal channel flow, *Phys. Fluids* **9**, 1054 (1997).
16. P. Baldi and K. Hornik, Neural networks and principal component analysis: Learning from examples without local minima, *Neural Networks* **2**, 53 (1989).

17. Y. Takane, Multivariate analysis by neural network models, in *Proceedings of the 63rd Annual Meeting of the Japan Statistical Society* (1995).
18. S. Haykin, *Neural Networks* (McMillan College, New York, 1997).
19. V. Cherkassky and F. Mulier, *Learning from Data* (Wiley, New York, 1998).
20. H. Bourlard and Y. Kamp, *Neural Networks for Pattern Recognition* (Oxford Univ. Press, London, 1995).
21. J. Kim, P. Moin, and R. Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, *J. Fluid Mech.* **177**, 133 (1987).
22. N. Schraudolph, *Local Gain Adaptation in Stochastic Gradient Descent*, Technical Report IDSIA 9 (1999).
23. P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences* (McGraw–Hill, New York, 1969).
24. Y. Cho, R. K. Agarwal, and K. Nho, Neural network approaches to some model flow control problems, in *4th AIAA Shear Flow Conference* (1997).
25. S. J. Schreck, W. E. Faller, and M. W. Luttges, Neural network prediction of unsteady separated flowfields, *J. Aircraft* **32**(1), 178 (1995).
26. W. E. Faller, S. J. Schreck, and M. W. Luttges, Real-time prediction and control of three-dimensional unsteady separated flow fields using neural networks, in *32th AIAA Aerospace Sciences Meeting and Exhibit* (1994).
27. C. Lee, J. Kim, D. Babcock, and R. Goodman, Application of neural networks to turbulence control for drag reduction, *Phys. Fluids* **9**(6), 1740 (1997).
28. H. Choi, P. Moin, and J. Kim, Active turbulence control for drag reduction in wall-bounded flows, *J. Fluid Mech.* **262**(75), 75 (1994).