Nonlinear Aeroelastic Reduced Order Modeling By Recurrent Neural Networks

Andrea Mannarino[a,*], Paolo Mantegazza[b]

[a] *Ph.D. candidate, Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy*

[b] *Professor, Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy.*

Abstract

The paper develops a reduction scheme based on the identification of continuous time recursive neural networks from input-output data obtained through high fidelity simulations of a nonlinear aerodynamic model at hand. The training of network synaptic weights is accomplished either with standard or automatic differentiation integration techniques. Particular emphasis is given to using such a reduced system in the determination of aeroelastic limit cycles. The related solutions are obtained with the adoption of two different approaches: one trivially producing a limit cycle through time marching simulations, and the other solving a periodic boundary value problem through a direct periodic time collocation with unknown period. The presented formulations are verified for a typical section and the BACT wing.

Keywords: Continuous Time Recurrent Neural Networks, Limit Cycle Oscillation, Periodic Collocation Method, Nonlinear Aeroelasticity

Nomenclature and common abbreviations

b	airfoil/wing semi-chord, measured in [m];
c	airfoil chord, measured in [m];
C_L, C_M	coefficients of lift and moment;
$\mathbf{e}$	network output error;
h, θ	plunge and pitch degree of freedoms, measured in [m] and [deg] respectively;
$k = \dfrac{\omega c}{V_\infty}$	reduced frequency;

[*] Corresponding author, email address: `andrea.mannarino@polimi.it`

l	wing span, measured in [m];
m	airfoil/wing mass, measured in [kg];
$r_\theta^2 = \dfrac{J_\theta}{mb^2}$	nondimensional airfoil/wing moment of inertia;
t	physical time, measured in [s];
$\mathbf{u}$	network input;
$V^* = \dfrac{V_\infty}{\omega_\theta b\sqrt{\mu}}$	reduced velocity;
$\mathbf{W}^x$, $\mathbf{W}^a$, $\mathbf{W}^b$, $\mathbf{W}^c$	network synaptic weights;
$x_\theta = \dfrac{S_\theta}{mb}$	nondimensional airfoil/wing static unbalance;
$\mathbf{x}$	network state;
$\mathbf{y}$	network output;
β	mixing parameter employed in the periodic collocation method;
$\mathbf{\Lambda}$	network Jacobian matrix;
$\mu = \dfrac{\mathrm{m}}{\pi\rho_\infty\, l\, b^2}$	fluid-mass ratio;
ρ_∞	fluid density, measured in [kg/m^3];
$\tau_s = \omega_\theta t$	structural adimensional time;
$\tau_a = \dfrac{V_\infty t}{b}$	aerodynamic adimensional time;
$\Phi(v)$	network activation function;
ω_h, ω_θ	uncoupled plunging and pitching circular frequencies, measured in [rad/s];
AD	automatic differentiation;
CFD	computational fluid dynamics;
CTRNN	continuous time recurrent neural network;
LCO	limit cycle oscillation;
NN	neural network;
PCM	periodic collocation method;
ROM	reduced order model.

1. Introduction

Nonlinear unsteady aerodynamic loads can be determined through high fidelity Computational Fluid Dynamics (CFD), which enables an accurate solution of many aeroelastic problems. Nevertheless, even with the computational power currently available, the costs of solving the related nonlinear high order problems still impede their routine adoption in the many repeated calculations required in preliminary aircraft design phase, making them more viable for the detailed validations typical of advanced design phases (Romanelli et al., 2012; Morton and Beran, 1999; Timme et al., 2011). It is therefore of utmost importance to develop Reduced Order Models (ROM) out of high fidelity numerical schemes to permit the use of nonlinear aerodynamics in a far wider analysis and design spectrum, while maintaining the needed level of accuracy required by aeroelastic stability and response calculations. An overview of ROM techniques can be found in (Lucia et al., 2004). The main approaches to ROM can be subdivided roughly into three main branches.

The first is the group of subspace projection methods, which exploits vector bases provided for example by a Proper Orthogonal Decomposition (POD) or Balanced POD (BPOD) (Schilders et al., 2008; Rowley, 2005; Mastroddi et al., 2012). A reduced (B)POD basis aims to retain most of the generalized numerical energy, i.e. a norm, of a system through a Singular Value Decomposition (SVD) of the matrix of the snapshots of appropriately accurate high fidelity responses. A smaller set of aerodynamic states can then be obtained by projecting the parent high fidelity approximation onto the determined (B)POD subspace. It should be remarked that when a reduction can be focused on periodic responses only, e.g. limit cycles, it is possible to include also the Harmonic Balance (HB) scheme within such a group (Thomas et al., 2002).

The second branch encompasses generalized interpolation methods, e.g. Radial Basis Function (RBF) or Kriging interpolators (Timme et al., 2011; Timme and Badcock, 2011). Such methods employ a high fidelity model to determine the solutions associated to a discrete set of parameters, to which a high order interpolation is applied afterward to evaluate any needed response at any intermediate point of interest. Therefore, the interpolator works as a general nonlinear input-output mapping, leading to a simpler representation of the dynamic system. Even if it is a robust technique, its application seems limited mostly to the evaluation of the aeroelastic stability.

The third group is represented by identification techniques based on input-output data. The Volterra series method (Lucia et al., 2004), based on the generalization of the impulse response of a system, is one of those techniques. Another approach, the one taken in this work, is characterized by the adoption of various forms of Neural Networks (NN). An example of the application of such an approach is the order reduction of relatively simple aeroelastic systems provided by the application of Discrete Time Recurrent Neural Networks (DTRNN) and Nonlinear AutoRegressive with eXogeneous input (NARX) NN models (Zhang et al., 2010; Yao and Liou, 2012). Instead, the present work introduces the relatively newer approach of Continuous Time Recurrent Neural Networks (CTRNN) to model order reduction. In this way, the resulting ROM is characterized by an equivalent non linear state formulation continuous in time, which can be exploited for both stability and response analyses, carried out much as on a standard set of parametrized differential equations. An immediate advantage of such a choice comes from the avoidance of the sampling time constraint implied in any NN discretized in time, which heavily affects the time steps to be used in any following analysis. As will be shown in Section 7, with the present reduced order modeling technique faster analyses can be carried out, with a much larger time step with respect to that employed in the training phase, which is constrained to smaller values because of the stability issues related to the explicit time scheme used by the CFD solver. If a DTRNN had been employed, the discretization time step would have been fixed, so constraining, a priori, the maximum representable frequency accounted for in the training phase.

Within the CTRNN framework, particular emphasis will be given to the determination of Limit Cycle Oscillations (LCO) of nonlinear aeroelastic systems, which will be computed through two different time domain approaches. The first of them is a trivial direct integration in time, starting from varied trial initial conditions, while the second imposes a periodic boundary solution through a Periodic Collocation Method (PCM) (Epureanu and Dowell, 2003; Manetti et al., 2009a). The PCM embeds an easy extension for the determination of the stability of LCOs through Floquet analyses. It should be a relatively new application of CTRNNs, at least for the computation of periodic aeroelastic solutions.

This work will show how the presented CTRNN based ROM can predict the nonlinear aerodynamic loads generated by a basis of structural motions, not yet assigned, so allowing to obtain a nonlinear aeroelastic system by coupling such a model to any underlying structure. Eventually the adopted PCM scheme will be verified through the computation of LCOs, comparing its results with those of the more common approach represented by direct trial simulations.

2. Continuous Time Recurrent Neural Network (CTRNN)

A neural network is a massively parallel distributed process combining simple processing units, the neurons, which have the natural capability of storing any knowledge accumulated through experience, making it available for later uses. Knowledge is acquired through a *learning process* and is stored in the synaptic connections linking the neurons. Because of their self-learning, fault tolerance, intrinsic nonlinearity, adaptivity and microbiological analogy (Haykin, 2009), neural networks are often used for nonlinear identifications.

Therefore, they are a powerful tool for approximating nonlinear dynamic systems, even when the system structure is unknown and only input-output data are available. They permit a sort of nonlinear generalized *black-box* modeling, thus avoiding the burden of stating a structured parametrization of the related differential equations, as required by other methods (Paduart et al., 2010). In an NN framework, the model is defined only by the structure and connections

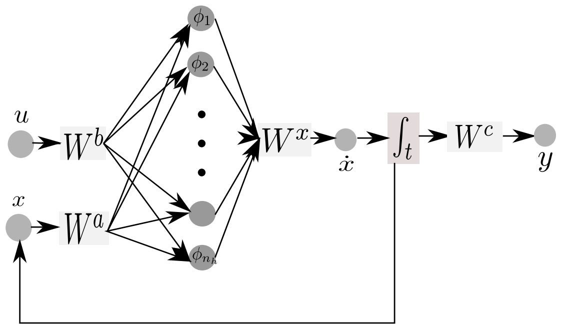


Figure 1: Schematic example of CTRNN.

of the network itself, i.e. the order of the system, the related parameters being determined through experimental/computational models, without any prior knowledge of the system internals. In fact the NN model parameters have no direct relationship to first principles (Nelles, 2001). The advantages of black-box NNs reside in their shorter modeling time and embedded implicit parametrization. So they can be used both for well-and not-so-well-understood processes. Such a feature could nevertheless be seen as a drawback of an NN, since it can lead to a scattered 'monkey-see, monkey do' approach, in place of exploiting the knowledge and physical understanding of a system at hand in guiding any learning process. Neural networks can be adopted both for the modeling of nonlinear functions and nonlinear dynamic systems (Haykin, 2009). In the latter case, such a representation is mostly casted in a recursive form, either in a discrete or continuous time framework. The logical scheme of the CTRNN considered in this work is shown in Figure 1.
The mathematical model is then represented by:

$$\dot{x}_j = \sum_{i=1}^{\mathrm{n}_h} W^x_{ji}\phi_i\left(\sum_{k=1}^{\mathrm{n}_x} W^a_{ik}\, x_k + \sum_{q=1}^{\mathrm{m}} W^b_{iq}\, u_q\right), \qquad j = 1, \dots, \mathrm{n}_x, \tag{1}$$

or more compactly:

$$\dot{\mathbf{x}} = \mathbf{W}^x\mathbf{\Phi}\left(\mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}\right), \qquad \mathbf{y} = \mathbf{W}^c\mathbf{x}, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{\mathrm{n}_x}$ is the network state, $\mathbf{u} \in \mathbb{R}^{\mathrm{m}}$ is the input, $\mathbf{y} \in \mathbb{R}^{\mathrm{p}}$ is the output, $\mathbf{\Phi} : \mathbb{R}^{\mathrm{n}_h} \longrightarrow \mathbb{R}^{\mathrm{n}_h}$ is the set of activation functions, $\mathbf{W}^a \in \mathbb{R}^{\mathrm{n}_h\times\mathrm{n}_x}$, $\mathbf{W}^b \in \mathbb{R}^{\mathrm{n}_h\times\mathrm{m}}$, $\mathbf{W}^x \in \mathbb{R}^{\mathrm{n}_x\times\mathrm{n}_h}$ and $\mathbf{W}^c \in \mathbb{R}^{\mathrm{p}\times\mathrm{n}_x}$ are the matrices containing the network synaptic weights; n_x, n_h, m and p being respectively the state, hidden and input-output space dimensions. The presented model is similar to the one found in (Seyab, 2006), with a slight modification of the bias term, which is not considered in the present work. Such a choice combines the advantage of a significant reduction of the size of the optimization problem with the possibility of a fixed neuron activation point, so leading to symmetrically structured networks.

From Eq. (2) above it is seen that, for a reduced aerodynamic model, it is appropriate to assume a strictly proper output form, whose matrix will be structured as:

$$\mathbf{W}^c = \begin{bmatrix}\mathbf{I}_{\mathrm{p}\times\mathrm{p}} & \mathbf{0}_{\mathrm{p}\times(\mathrm{n}_x-\mathrm{p})}\end{bmatrix}, \tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{0}$ is a null matrix. The structure of Eq. (3) shows that the output of the first p hidden neurons are the output of the network, leading to a direct and easier physical interpretation of the computed results (Haykin, 2009).

A physical description of the quantities involved in the dynamics of the neural network may clarify its meaning. Consider for example an airfoil experiencing pitch and plunge oscillations. If the aim of the CTRNN is the identification of the aerodynamic loads, then the input $\mathbf{u}$ is composed by the pitch and plunge themselves, while the output $\mathbf{y}$ are the nondimensional load coefficients. Because of Eq. (3), the first two component of the state $\mathbf{x}$ will be represented by the CTRNN output, meanwhile the others will have no direct physical meaning, they will increase only the capabilities of the CTRNN in its task of identifying motion-dependent loads. Therefore, in this case the input and output dimensions will be $\mathrm{m} = 2$ and $\mathrm{p} = 2$, while the state dimension n_x should be chosen as small as required to maintain an acceptable computational effort for a wanted precision.

Following a few preliminary numerical experiments the logistic function and its sensitivity to the neuron potential have been found to be an adequate choice for neurons activation. As such they are written as:

$$\phi(v) = \frac{1}{1+\exp(-v)}, \qquad \frac{d\phi}{dv} = \frac{\exp(-v)}{\left(1+\exp(-v)\right)^2}, \tag{4}$$

where v is the neuron potential, its first derivative playing a significant role within the training algorithm.

3. CTRNN training

Without a doubt, the most demanding effort in setting up a meaningful NN is its training. Different strategies have been proposed in the literature, in particular for the DTRNN case, such as the *Back Propagation Through Time* (BTTP) algorithm, for batch learning, and the *Real Time Recurrent Learning* (RTRL) for online applications. An RTRL has also been proposed for the continuous case (Pearlmutter, 1989). Both algorithms have demonstrated good convergence properties within the realm of the applications of interest for the present work. (Haykin, 2009).

It should be nonetheless remarked that no standard procedure has yet been defined for the training of CTRNNs. So, since it has been verified that a simple gradient based learning method, e.g. BTTP and RTRL, is somewhat inefficient (Suykens and Vanderwalle, 1995), improved optimization algorithms should be used, such as the Levenberg-Marquardt (LM) scheme (Marquardt, 1963) or a Genetic Algorithm (GA) (Goldberg, 1989).

In fact the training of a neural network can be viewed as a nonlinear optimization problem, which can greatly benefit from the knowledge of the Jacobian matrix associated to any stationary point. Such a Jacobian is clearly referred to the derivative of the system output with respect to the unknown parameters, the synaptic weights of Eq. (2) in our case, that must be designed by minimizing a given cost function. To such an aim let us define the *state* Jacobian matrix:

$$\mathbf{\Lambda} = \frac{\partial \mathbf{x}}{\partial \theta}, \tag{5}$$

where $\theta = \left(\operatorname{vec}\left(\mathbf{W}^{x}\right)^{\mathrm{T}}, \operatorname{vec}\left(\mathbf{W}^{a}\right)^{\mathrm{T}}, \operatorname{vec}\left(\mathbf{W}^{b}\right)^{\mathrm{T}}\right)^{\mathrm{T}}$, $\operatorname{vec}(\cdot)$ being the operator ordering a matrix into a vector by stacking its columns. Moreover, being associated to a dynamic system, the above Jacobian matrix will change with time. Thus a system of $\mathrm{n}_x \times \mathrm{n}_\theta$ Ordinary Differential Equations (ODEs) must be added and coupled to the original n_x ODEs of Eq. (2) during the network training, with $\mathrm{n}_\theta = 2n_x \cdot \mathrm{n}_h + \mathrm{n}_h \cdot \mathrm{m}$ representing the total number of optimization variables.

Such a training should find out the optimal synaptic weights through the minimization of a given cost function, which, in our case, will be the most common one found in the literature for similar identification problems, i.e. a quadratic function of the Output Error $\mathbf{e}(t)$, defined as:

$$\mathbf{e}(t) = \hat{\mathbf{y}}(t) - \mathbf{W}^{c}\mathbf{x}(t), \tag{6}$$

where $\hat{\mathbf{y}}(t)$ is the output of the parent high order system. Therefore, the cost function reads:

$$F = \frac{1}{2}\sum_{k=1}^{N_t} \mathbf{e}^{\mathrm{T}}(t_k)\mathbf{e}(t_k), \tag{7}$$

where N_t is the number of sampled points. The optimization algorithm will find a minimum of the cost function with respect to the unknown optimal synaptic weights.

It should be noted that, despite its relatively robust convergence properties, the LM solver may prove weak when started from an initial guess point θ_0 far away from the optimal solution. Therefore, a hybrid technique has been implemented, running a few iterations of a global GA (Goldberg, 1989) first, resuming LM when the genetic solution hooks an optimum region, that in this work is represented by a cost function value smaller than a selected threshold. Beside fostering convergence from very rough initial guesses, an added advantage of such a hybrid approach resides in avoiding the calculation of the almost useless initial Jacobian matrices, resuming their costly calculation only when the full advantage of the second order convergence provided by LM is almost assured.

In relation to the refined LM algorithm, it becomes useful to define the vector:

$$\mathbf{E}(\theta) = \left(\mathbf{e}(t_1,\theta)^{\mathrm{T}} \quad \mathbf{e}(t_2,\theta)^{\mathrm{T}} \cdots \mathbf{e}(t_{N_t},\theta)^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{8}$$

with $\mathbf{E} \in \mathbb{R}^{N_t \cdot \mathrm{p}}$, meanwhile the output Jacobian matrix is defined by:

$$\mathbf{J}(\theta) = \frac{\partial \mathbf{E}(\theta)}{\partial \theta}, \tag{9}$$

which has dimensions $\mathbb{R}^{(N_t \cdot \mathrm{p}) \times \mathrm{n}_\theta}$, and can be split into the blocks:

$$\mathbf{J}(\theta) = \left[\mathbf{J}(t_1, \theta)^{\mathrm{T}} \quad \mathbf{J}(t_2, \theta)^{\mathrm{T}} \cdots \mathbf{J}(t_{N_t}, \theta)^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{10}$$

each of dimensions $\mathbb{R}^{\mathrm{p} \times \mathrm{n}_\theta}$, as given by:

$$\mathbf{J}(t_k, \theta) = \frac{\partial \mathbf{e}(t_k, \theta)}{\partial \theta} = -\mathbf{W}^c \boldsymbol{\Lambda}(t_k, \theta), \qquad k = 1, \dots, N_t. \tag{11}$$

Having defined the state Jacobian matrix in Eq. (5), it is easy to compute its dimensions as being given by $\mathbb{R}^{\mathrm{n}_x \times \mathrm{n}_\theta}$ matrix. For a more straightforward computation of its elements, it becomes easier to split it into the following blocks:

$$\boldsymbol{\Lambda} = \left[\frac{\partial \mathbf{x}}{\partial W^x_{ij}} \quad \frac{\partial \mathbf{x}}{\partial W^a_{ij}} \quad \frac{\partial \mathbf{x}}{\partial W^b_{ij}}\right]. \tag{12}$$

Taking the derivative of the state $\mathbf{x}$ with respect of the elements of the synaptic weights matrix $\mathbf{W}^x$ as an example, each sub-matrix will have the following structure:

$$\frac{\partial \mathbf{x}}{\partial W^x_{ij}} = \left[\frac{\partial \mathbf{x}}{\partial W^x_{11}}, \frac{\partial \mathbf{x}}{\partial W^x_{21}}, \dots, \frac{\partial \mathbf{x}}{\partial W^x_{\mathrm{n_x}1}}, \frac{\partial \mathbf{x}}{\partial W^x_{12}}, \frac{\partial \mathbf{x}}{\partial W^x_{22}}, \dots, \frac{\partial \mathbf{x}}{\partial W^x_{\mathrm{n_x}2}}, \dots, \frac{\partial \mathbf{x}}{\partial W^x_{\mathrm{n_x n_h}}}\right]. \tag{13}$$

Remembering the size of each vector/matrix involved and the order they have been sorted with, their assembly will result in a straightforward operation. Defining the single-entry matrix $\mathbf{I}_{ij}$ (Petersen and Pedersen, 2008), as the matrix with a 1 at the position (i, j) and zero elsewhere and using the shorthand notation $\mathbf{z} = \mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}$, the direct computation of the blocks of Eq. (12) from Eq. (2) reads:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^x_{ij}} = \mathbf{I}_{ij}\boldsymbol{\Phi}(\mathbf{z}) + \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^x_{ij}}, \qquad \begin{array}{l} i = 1, \dots, \mathrm{n}_x, \\ j = 1, \dots, \mathrm{n}_h, \end{array} \tag{14}$$

with $\boldsymbol{\Psi}(\mathbf{z}) = \mathrm{diag}_{\mathrm{n}_h}\left[\frac{\mathrm{d}\phi_1}{\mathrm{d}z_1}, \frac{\mathrm{d}\phi_2}{\mathrm{d}z_2}, \cdots, \frac{\mathrm{d}\phi_{\mathrm{n}_h}}{\mathrm{d}z_{\mathrm{n}_h}}\right]$, then:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^a_{ij}} = \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\left(\mathbf{I}_{ij}\mathbf{x} + \mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^a_{ij}}\right), \qquad \begin{array}{l} i = 1, \dots, \mathrm{n}_h, \\ j = 1, \dots, \mathrm{n}_x, \end{array} \tag{15}$$

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^b_{ij}} = \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\left(\mathbf{I}_{ij}\mathbf{u} + \mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^b_{ij}}\right), \qquad \begin{array}{l} i = 1, \dots, \mathrm{n}_h, \\ j = 1, \dots, \mathrm{m}, \end{array} \tag{16}$$

Assembling the different blocks (14), (15) and (16), defining:

$$\mathbf{U} = \left[\mathbf{I}_{ij}\boldsymbol{\Phi}(\mathbf{z}), \quad \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\mathbf{I}_{ij}\mathbf{x}, \quad \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\mathbf{I}_{ij}\mathbf{u}\right], \tag{17}$$

and:

$$\hat{\mathbf{A}} = \mathbf{W}^x\boldsymbol{\Psi}(\mathbf{z})\mathbf{W}^a, \tag{18}$$

we end with a time varying Jacobian matrix governed by the following differential matrix equation:

$$\dot{\boldsymbol{\Lambda}} = \hat{\mathbf{A}}\boldsymbol{\Lambda} + \mathbf{U}. \tag{19}$$

Eventually the coupling of Eq. (2) and Eq. (19) fully defines the *nonlinear state dynamics* of a CTRNN:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x\boldsymbol{\Phi}\left(\mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}\right) \\ \dot{\boldsymbol{\Lambda}} = \hat{\mathbf{A}}\boldsymbol{\Lambda} + \mathbf{U} \\ \mathbf{y} = \mathbf{W}^c\mathbf{x} \\ \mathbf{J} = -\mathbf{W}^c\boldsymbol{\Lambda}. \end{cases} \tag{20}$$

It can thus be seen that both GA and LM require the forward time solution of an ODE system, for the current value of the network synaptic weights. Because of the structure chosen for $\mathbf{W}^c$ in Eq. (3) it is possible to assign the initial condition required by Eq. (20) as being the initial value of the output of a high fidelity simulation, thus defining a physical meaning for the network through the following:

$$\mathbf{x}_0 = \left(\hat{\mathbf{y}}_0^{\mathrm{T}} \quad \mathbf{0}^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{21}$$

where the size of the null vector is equal to $\mathrm{n}_x - \mathrm{p}$. Instead, the initial condition of the state Jacobian matrix is taken as a null matrix, meaning that the initial network synaptic weights reside at a stationary point of $\mathbb{R}^\theta$ (Haykin, 2009).

Since we will take into account only aerodynamic nonlinearities our CTRNN will identify the aerodynamic response computed by a high fidelity CFD code. Then, the thus determined CTRNN model can be coupled to a structure so to produce a complete aeroelastic ROM solver, which should be much more effective in providing precise nonlinear response and stability simulations, at a faster pace than that of corresponding full order high fidelity calculations.

4. Tackling the CTRNN training with Automatic Differentiation (AD) tools

In practical engineering applications, it is relatively simpler to code response functions than their derivatives with respect to any parameter of interest. Nevertheless, accurate values of a function derivatives often play a central role in model *validation* (sensitivity analysis) and *optimization.* The adoption of Automatic Differentiation (AD) tools permit to obtain an automatic determination of any derivative required to build a Jacobian matrix on the base of the mere availability of the related function written in a computer code, so avoiding their analytic evaluation, which can be cumbersome. A sound illustration of such a point can be found in (Griewank and Walther, 2008; Griewank et al., 1996).

The two main basic algorithms used by AD to compute derivatives are the *forward* mode, which computes the function value and its derivatives in parallel, and the *reverse* or *adjoint* mode, which requires a first evaluation of the function, followed by the evaluation of its derivatives through a step by step backward run over the definition of the function itself, as for the Backpropagation Algorithm for NNs (Griewank and Walther, 2008). It should be clear that, while the forward mode follows the same path for a number of times equal to the number of partial derivative desired, the reverse mode permits to compute all of the partial derivatives of a function in a single shot, i.e. following the reverse path once.

Moreover, by exploiting the properties of a Taylor series in the approximation of a function, an AD is directly applicable also to the training of the neural network previously presented. In fact the CTRNN model can be written as a general parametrization of a nonlinear system:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \theta\right), \qquad \mathbf{y} = \mathbf{W}^c \mathbf{x}, \tag{22}$$

so that the related *network sensitivity* can be defined as the variation of the network output against θ. Then, differentiating Eq. (22) with respect to θ, one obtains the following ordinary differential matrix problem:

$$\dot{\mathbf{x}}_\theta = \mathbf{f}_\mathbf{x} \mathbf{x}_\theta + \mathbf{f}_\theta, \qquad \mathbf{y}_\theta = \mathbf{W}^c \mathbf{x}_\theta, \tag{23}$$

with $\mathbf{f}_\mathbf{x} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbb{R}^{\mathrm{n}_x \times \mathrm{n}_x}$, $\mathbf{f}_\theta = \dfrac{\partial \mathbf{f}}{\partial \theta} \in \mathbb{R}^{\mathrm{n}_x \times \mathrm{n}_\theta}$, $\mathbf{x}_\theta = \dfrac{\partial \mathbf{x}}{\partial \theta} \in \mathbb{R}^{\mathrm{n}_x \times \mathrm{n}_\theta}$ and $\mathbf{y}_\theta = \dfrac{\partial \mathbf{y}}{\partial \theta} \in \mathbb{R}^{\mathrm{p} \times \mathrm{n}_\theta}$. The above system will be integrated along with the system dynamics in Eq. (22), starting from a given initial condition $\mathbf{x}_\theta(0) = \dfrac{\partial \mathbf{x}(0)}{\partial \theta}$. The related system of ODEs is composed by $\mathrm{n}_x + \mathrm{n}_x \times \mathrm{n}_\theta$ equations and can be integrated numerically through any of the well known standard ODE solvers. Nonetheless, typically, its size grows very quickly and since such an integration must be performed at each iteration of the training algorithm, this part constitutes the real burden of a network training.

Consider the following initial value problem:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)), \qquad \mathbf{x}(0) = \mathbf{x}_0, \tag{24}$$

with f differentiable d-times in time. Defining $\mathbf{z}(t) = \mathbf{f}(\mathbf{x}(t))$, the Taylor coefficients of the solution can be computed through the following iterative formula:

$$\mathbf{x}_{i+1} = \frac{\mathbf{z}_i}{i+1}, \tag{25}$$

which derives from the application of the forward AD mode to Eq. (24). The usual procedure is to calculate $\mathbf{z}_0 = \mathbf{x}_1$ from $\mathbf{x}_0$, then plugging it into $\mathbf{x}_0 + \mathbf{x}_1\, t$ of $\mathbf{z}(t)$ so to obtain $\frac{1}{2}\mathbf{z}_1 = \mathbf{x}_2$ and so on. In this way one has to perform d sweeps through the evaluation algorithm for $\mathbf{f}$ with the degree of the Taylor arithmetic growing by one at each time (Griewank and Walther, 2008).

Along with the solution of the ODE system, the analyst may desire to compute the *Jacobian* matrix, $\mathbf{B} = \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{x}_0}$, solution of the following initial value problem:

$$\dot{\mathbf{B}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{B} = \mathbf{A}\mathbf{B}, \qquad \mathbf{B}(0) = \mathbf{I}, \tag{26}$$

with $\mathbf{I}$ the identity matrix. Clearly, because of its implicit dependence on $\mathbf{x}$, the time varying Jacobian matrix must be integrated along with the state dynamics itself. The expression of the Taylor elements of $\mathbf{B}$ are then obtained applying the chain rule to Eq. (25):

$$\mathbf{B}_{i+1} = \frac{\mathrm{d}\mathbf{x}_{i+1}}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\frac{\mathrm{d}\mathbf{z}_i}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\sum_{j=0}^{i}\frac{\partial \mathbf{z}_i}{\partial \mathbf{x}_j}\frac{\mathrm{d}\mathbf{x}_j}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\sum_{j=0}^{i}\mathbf{A}_{i-j}\mathbf{B}_j, \tag{27}$$

where the matrices $\mathbf{A}_j$ are computed through the AD reverse mode (Griewank and Walther, 2008; Seyab, 2006). In practice, a more straightforward approach can be used. In fact after defining a normalized time $\tau = t/\Delta t$, where Δt is the adopted integration step, the ODE system defined in Eq. (24) is transformed to:

$$\frac{\mathrm{d}}{\mathrm{d}\tau} = \frac{1}{\Delta t}\frac{\mathrm{d}}{\mathrm{d}t} \rightarrow \frac{\mathrm{d}\hat{\mathbf{x}}(\tau)}{\mathrm{d}\tau} = \hat{\mathbf{x}}' = \Delta t\, \mathbf{f}(\hat{\mathbf{x}}(\tau)), \tag{28}$$

with $\hat{\mathbf{x}}(\tau) = \mathbf{x}(t/\Delta t)$. The solution and its Jacobian matrix trajectories can now be expressed as:

$$\hat{\mathbf{x}}(\tau) = \hat{\mathbf{x}}_0 + \hat{\mathbf{x}}_1\tau + \hat{\mathbf{x}}_2\tau^2 + \cdots + \hat{\mathbf{x}}_d\tau^d + \mathcal{O}\left(\tau^{d+1}\right), \tag{29}$$

$$\mathbf{A}(\tau) = \mathbf{A}_0 + \mathbf{A}_1\tau + \mathbf{A}_2\tau^2 + \cdots + \mathbf{A}_d\tau^d + \mathcal{O}\left(\tau^{d+1}\right), \tag{30}$$

meanwhile the Taylor series elements of the Jacobian matrix are assembled by Eq. (27). Starting from the initial values $\mathbf{x}_0$ and $\mathbf{B}_0$, the solution at each time step $t_{i+1} = t_i + \Delta t$ is computed explicitly as follows:

$$\mathbf{x}(t_i + \Delta t) = \sum_{j=0}^{d}\hat{\mathbf{x}}_j, \tag{31}$$

and:

$$\mathbf{B}(t_i + \Delta t) = \sum_{j=0}^{d}\mathbf{B}_j, \tag{32}$$

with the maximum Taylor series degree d being adjusted so to obtain an accurate solution of Eq. (24).

However, like for any explicit numerical integrators, for stability reasons, Taylor expansion methods are forced to adopt excessively small step sizes, wherever the system is stiff. There are nonetheless *A-stable* implicit variants that overcome this limitation (Griewank et al., 1996). It will nevertheless be seen that such a stability problem will not affect what presented in this work significantly, since the training phase is based on signals computed by CFD simulations, having a very fine resolution in time.

In this work, the above Taylor series elements will be computed with the support of the package ADOL-C (Automatic Differentiation by OverLoading in C++) (Griewank et al., 1996), making it much simpler the evaluation of the any order derivatives of the easily coded vector-valued functions. Moreover, contrarily to the source transformation approach, the generation of an intermediate source code can be avoided by exploiting the C++ operator overloading, thus saving run time memory. It is noted that ADOL-C permits also the computation of directional (Lie) derivatives, gradients of any Taylor coefficient with respect to any independent variable.

It is worth noting that Eq. (22) is *nonautonomous.* In such a case ADOL-C permits the computation of the Jacobian matrix only with respect to the state variables, a limitation that can be overcome by simply adding the differential equations modeling constant parameters, so resulting in the augmented system:

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \theta\right), \qquad \dot{\theta} = \mathbf{0}, \qquad \mathbf{y} = \mathbf{W}^{c}\mathbf{x}, \tag{33}$$

with the value of θ being the one available at the beginning of the iterated optimization, which remain constant in the computation of the CTRNN response. While it is true that the size of the ODE system is augmented, the added differential equations are trivial. Moreover, since the CTRNN represents a ROM, the size of n_{θ} will usually be small, thus the overall integration procedure will suffer only for a very small added cost.

At this point let us rewrite Eq. (34) as being autonomous by defining the extended state variable $\mathbf{z}$:

$$\mathbf{z} = \left\{\mathbf{x}^{\mathrm{T}} \qquad \theta^{\mathrm{T}}\right\}^{\mathrm{T}}, \tag{34}$$

so that Eq. (22) now reads:

$$\dot{\mathbf{z}} = \mathbf{f}\left(\mathbf{z}\right), \qquad \mathbf{y} = \left[\mathbf{W}^{c} \quad \mathbf{0}_{\mathrm{p}\times\mathrm{n}_{\theta}}\right]\mathbf{z} = \hat{\mathbf{W}}^{c}\mathbf{z}. \tag{35}$$

Therefore, the forward and reverse AD modes may be easily employed for the computation of the Taylor series elements presented in Eqs. (25) and (27) respectively, afterward computing the solution of Eq. (35) through Eqs. (31) and (32).

5. Limit Cycle Oscillations

Limit Cycle Oscillations (LCOs) are a peculiar phenomenon encountered in the analysis of nonlinear dynamic systems. In the context of the related theory, an LCO is one of the simplest dynamic bifurcation, *"a first stop on the road to chaos"* (Dowell et al., 2004). Without focusing too much on bifurcation analysis, as there is a vast supporting literature on such a subject, e.g. in general (Seydel,

1988) and for aeroelastic LCOs (Morton and Beran, 1999; Stanford and Beran, 2013), let us consider a few details pertaining to the aeroelastic case.

Both the aerodynamic (Morton and Beran, 1999; Zhang et al., 2010; Yao and Liou, 2012) and structural (Manetti et al., 2009a; Epureanu and Dowell, 2003; Li et al., 2012) nonlinearities can lead to such a characteristic behaviour. So an advantage of using theoretical models resides in the possibility of analysing each of the several possible physical phenomena leading to an LCO separately.

An LCO response can be determined by following two main lanes. The first is a brute force, repeated direct numerical integration of the full-order/high-fidelity nonlinear system, since the computational power available nowadays makes it straightforward, especially in view of its large grain massive parallelization. Nevertheless, such an approach remains computationally very expensive, especially during the preliminary design phase, where many computational simulations are needed to evaluate any required performance over a large set of configurations. The other approach is based on enforcing a periodic solution of the nonlinear system under consideration, either through a direct periodic collocation in time (Manetti et al., 2009b; Epureanu and Dowell, 2003; Borri and Mantegazza, 1987) or through a harmonic balance in the frequency domain (Thomas et al., 2002; Lucia et al., 2004; Thomas et al., 2010; Kholodar et al., 2004), the time domain approach being the one followed in this paper.

Time marching integrations are largely used also for determining the flutter condition through numerical experiments involving nonlinear aerodynamics (Morton and Beran, 1999). Today such methods are mainly used to validate the results of stability changes obtained with a Hopf bifurcation analysis at varying dynamic pressures, which can provide any stable/unstable response but only stable LCOs.

Calling λ any bifurcation parameter of interest, we have seen that our ROM can be associated to a set of explicit nonlinear ODEs of the type:

$$\dot{\mathbf{x}} = \mathbf{R}\left(\mathbf{x};\lambda\right) \quad \text{for} \quad t > 0, \qquad\qquad \mathbf{x}\left(t=0\right) = \mathbf{x}_0, \tag{36}$$

whose stable LCOs are much more amenable to an easy and relatively efficient determination by means of time simulations, without much concern about the choice between explicit and implicit integration methods because of the relatively small size of $\mathbf{x}$ associated to a ROM model.

Nevertheless, a periodic LCO trajectory can be enforced *a priori* onto Eq. (36) and its solution computed both in the time and frequency domain. The frequency domain approach exploits Fourier series, with unknown period, typically, considering only a few first fundamental components, leading to the well known Describing Function method (Gelb and Vander Velde, 1968; Manetti et al., 2009a) and numerically generalized HB formulations (Thomas et al., 2002; Hall et al., 2002; Lucia et al., 2004).

Here we will follow the alternative approach of casting a periodic solution directly in the time domain, through the enforcement of a collocated periodic discretization (Epureanu and Dowell, 2003; Borri and Mantegazza, 1987). In such a view it is simpler to reformulate Eq. (36) with its more general fully implicit form:

$$\mathbf{F}\left(\mathbf{x}, \dot{\mathbf{x}}, \lambda\right) = \mathbf{0}, \tag{37}$$

which is assumed to admit an LCO with unknown period T. Then, to impose such a periodic solution, we divide its period T in N time intervals, thus $N+1$ time nodes. In view of the adoption of an adaptive discretization in time, the related intervals Δt can be non-uniform, hence they are be defined by a vector α such that:

$$\Delta t_i = \alpha_i T, \qquad i = 1, ..., N. \tag{38}$$

At each time interval, Eq. (37) can be time weighted and collocated at the corresponding instant, for example with the Mid-Point Rule (MPR), reading:

$$\mathbf{F}_{\mathrm{MPR}_i} = \mathbf{F}\left(\frac{\mathbf{x}(t_{i+1}) + \mathbf{x}(t_i)}{2}, \frac{\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i)}{\Delta t_i}\right)\Delta t_i = \mathbf{0}. \tag{39}$$

The MPR is a second order, symplectic, energy preserving integrator, thus it can be efficiently adopted for solving Eq. (37). Hovever, if Eq. (37) is stiff or is a set of Differential Algebraic Equations (DAE), there is the possibility that the MPR converges toward a ringing solution, possibly jeopardizing convergence. In order to solve this problem, Ref. (Manetti et al., 2009b) adopts a hybrid method, mixing MPR with a second order Backward Difference Formula (BDF2) whose collocation gives:

$$\mathbf{F}_{\mathrm{BDF}_i} = \mathbf{F}\left(\mathbf{x}\left(t_i\right), \frac{\rho_{1_i}\mathbf{x}\left(t_{i+1}\right) + \rho_{2_i}\mathbf{x}\left(t_i\right) + \rho_{3_i}\mathbf{x}\left(t_{i-1}\right)}{\Delta t_i}\right)\Delta t_i = \mathbf{0}, \tag{40}$$

where:

$$\rho_i = \frac{\alpha_i}{\alpha_{i+1}}, \qquad \rho_{1_i} = 1 + \frac{\rho_i}{\rho_{i+1}}, \qquad \rho_{2_i} = \rho_i + 1, \qquad \rho_{3_i} = \frac{\rho_i^2}{\rho_i + 1}, \tag{41}$$

with ρ_i allowing the use of differing collocation steps so to adapt the time mesh to the nonlinear solution gradients. The mentioned hybridization consists in a linear combination of the two methods, as given by:

$$\beta\mathbf{F}_{\mathrm{MPR}_i}\left(\mathbf{x}\left(t_i\right), \mathbf{x}\left(t_{i+1}\right)\right)\Delta t_i + \left(1-\beta\right)\mathbf{F}_{\mathrm{BDF}_i}\left(\mathbf{x}\left(t_{i-1}\right), \mathbf{x}\left(t_i\right)\mathbf{x}\left(t_{i+1}\right)\right)\Delta t_i = \mathbf{0}, \tag{42}$$

so that when the coefficient β tends to 0, the dissipation of any unwarranted high frequency content of the solution is provided by BFD2, so solving any MPR ringing problem, albeit at the cost of some loss of precision (Manetti et al., 2009b). The discretization of Eq. (39) leads to $N \cdot n$ nonlinear algebraic equations in $n \cdot (N+1) + 1$ unknowns: the $N+1$ vector solutions $\mathbf{x}(t_i)$ and the period T, where of course n is the dimension of $\mathbf{x}$. Therefore $n+1$ more equation must be added for the mathematical closure of the problem. Since we are looking for periodic solution, n equations are introduced by imposing the periodicity:

$$\mathbf{x}(t_1) = \mathbf{x}(t_{N+1}). \tag{43}$$

Moreover, since the system is autonomous, the time origin is also unknown and must be defined, fixing in this way the phase reference of the computed limit cycle. That is done by imposing a single element, say the k-th, of the vector solution $\mathbf{x}$ at an arbitrary collocation time point, say the j-th, so providing the last equation needed for the closure of the problem:

$$x_k(t_j) = C, \tag{44}$$

where $C \in \mathbb{R}$. The resulting system of nonlinear algebraic equations is eventually solved with a nonlinear solver, such as Newton-Raphson. In practical applications the efficiency of the solution is usually enhanced by adopting a continuation on the number of time interval used N, starting from a low number of points and then increasing them using an interpolation of the previous solution as an initial guess. Such an approach will be here called Periodic Collocation Method (PCM). Further continuations strategies can be applied to varying system parameters, as in the case of an aeroelastic problem (Manetti et al., 2009b; Epureanu and Dowell, 2003).

After ordering the collocated unknowns as:

$$\mathbf{Z} = \left(\mathbf{x}^{\mathrm{T}}(t_1), \mathbf{x}^{\mathrm{T}}(t_2), \ldots, \mathbf{x}^{\mathrm{T}}(t_{N+1}), T\right)^{\mathrm{T}}. \tag{45}$$

The block structure of the Jacobian matrix, required by a Newton-Raphson solver, is shown below in Eq. (46):

$$\mathbf{J} = \begin{bmatrix} \mathbf{A}_1^1 & \mathbf{A}_1^2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{b}_1^1 \\ \mathbf{A}_2^1 & \mathbf{A}_2^2 & \mathbf{A}_2^3 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{b}_2^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_{i-1}^k & \mathbf{A}_i^k & \mathbf{A}_{i+1}^k & \cdots & \mathbf{0} & \mathbf{b}_i^k \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{N-1}^N & \mathbf{A}_N^N & \mathbf{A}_{N+1}^N & \mathbf{b}_N^N \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{46}$$

with its blocks given by:

$$\begin{aligned} \mathbf{A}_{i-1}^k &= (1-\beta)\,\frac{\rho_3}{\alpha_k T}\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{MPR, k}}, \\ \mathbf{A}_i^k &= \beta\left(\frac{1}{2}\,\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\bigg|_{\text{MPR, k}} - \frac{1}{\alpha_k T}\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{MPR, k}}\right)\Delta t_i + (1-\beta)\frac{\rho_2}{\alpha_k T}\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{BDF, k}}\Delta t_i, \\ \mathbf{A}_{i+1}^k &= \beta\left(\frac{1}{2}\,\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\bigg|_{\text{MPR, k}} + \frac{1}{\alpha_k T}\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{MPR, k}}\right)\Delta t_i + (1-\beta)\left(\frac{1}{2}\,\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\bigg|_{\text{BDF, k}} + \frac{\rho_1}{\alpha_k T}\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{BDF, k}}\right)\Delta t_i, \end{aligned} \tag{47}$$

and:

$$\mathbf{b}_i^k = \mathbf{F}\left(\mathbf{x}(t_i), \dot{\mathbf{x}}(t_i)\right) - \left[\beta\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{MPR, k}} \dot{\mathbf{x}}_{\text{MPR}}^k + (1-\beta)\,\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\bigg|_{\text{BDF, k}} \dot{\mathbf{x}}_{\text{BDF}}^k\right]\alpha_k, \tag{48}$$

where the subscript MPR and BDF indicates that the collocated functions and derivatives refer to the use of an MPR and BDF formula respectively. Such an approach is quite efficient, since the Jacobian matrix is usually easy to compute and *sparse*, therefore requiring a limited number of operations.

Once a converged solution has been obtained, the *monodromy* matrix $\mathbf{Q}$ is computed to investigate the LCO stability within the framework of the *Floquet* theory (Jordan and Smith, 2007). Therefore, let us consider the linearization of Eq. (37), around an LCO solution, i.e. for $\mathbf{F}|_{\mathrm{LCO}} = 0$ at any time:

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathrm{LCO}} \Delta\mathbf{x} + \left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\mathrm{LCO}} \Delta\dot{\mathbf{x}} = \mathbf{0}, \tag{49}$$

which is a first order linear ordinary differential system with periodically varying coefficients. Following a standard literature approach, e.g. see (Jordan and Smith, 2007), we write the above equation as:

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{P}(t)\Delta\mathbf{x}(t), \qquad \text{with} \quad \mathbf{P}(t) = -\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}^{-1}\right|_{\mathrm{LCO}} \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathrm{LCO}}, \tag{50}$$

with $\mathbf{P}(t+T) = \mathbf{P}(t)$ for all t. Then we combine all of its independent solutions in a unique matrix $\mathbf{X}$, called the *fundamental* matrix of the system, defined as:

$$\mathbf{X} = \left[\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \cdots, \Delta\mathbf{x}_n\right], \tag{51}$$

where $\mathbf{X}$ is the solution of:

$$\dot{\mathbf{X}} = \mathbf{P}\mathbf{X} \quad \text{for} \quad t > 0, \qquad \mathbf{X}(0) = \mathbf{I}. \tag{52}$$

According to Floquet theorem (Jordan and Smith, 2007), the fundamental matrix satisfies:

$$\mathbf{X}(t+T) = \mathbf{Q}\mathbf{X}(t) \qquad \forall\, t, \tag{53}$$

so that $\mathbf{Q} = \mathbf{X}(T)$ and its eigenvalues are called the characteristic numbers of Eq. (50), which are independent constants. Their values characterize the stability of any obtained LCO solution. Within the presented PCM approach $\mathbf{Q}$ is simply determined by solving the following linear system (Manetti et al., 2009b):

$$\begin{bmatrix} \mathbf{A}_2^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_2^2 & \mathbf{A}_3^2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_{i-1}^k & \mathbf{A}_i^k & \mathbf{A}_{i+1}^k & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{N-1}^N & \mathbf{A}_N^N & \mathbf{A}_{N+1}^N \end{bmatrix} \begin{Bmatrix} \mathbf{X}(t_1) \\ \mathbf{X}(t_2) \\ \vdots \\ \mathbf{X}(t_i) \\ \vdots \\ \mathbf{Q} \end{Bmatrix} = - \begin{Bmatrix} \mathbf{A}_1^1 \\ \mathbf{A}_1^2 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{Bmatrix}. \tag{54}$$

An LCO will be stable if all of the modulus of the eigenvalues of $\mathbf{Q}$ are lower than one, taking into account that there is always an eigenvalue with modulus equal to one (Manetti et al., 2009a; Jordan and Smith, 2007). So, if at least one eigenvalue has a modulus higher than one the LCO will be unstable. It is worth remarking that if any of the highest eigenvalue is significantly close to one, a finer period discretization should be use as a check, to avoid attributing an unstable behaviour just because of the inadequate precision of any found solution.

6. Aerodynamic models

In the following examples the Euler option of the in-house CFD solver AeroFoam (Romanelli, 2012) will be adopted for the training of our CTRNNs. Such a solver is supported by the open source tool OpenFOAM (VV.AA., 2010) for the handling of the computational grid data, numerical solutions and the pre/post-processing phases. It is a density-based Euler/Reynolds-Averaged Navier-Stokes (RANS) solver, formulated in an Arbitrary-Lagrangian-Eulerian framework. Accordingly, it can treat efficiently problems with moving grids, permitting the simulation of aeroservoelastic applications. The Euler flow model has been chosen in the present analyses because the resulting LCOs will be enhanced by the large movements of shock waves over the wing surface. Such an effect would have been smeared out if RANS simulation had been used out, with a costlier computational effort. Furthermore, this choice allows to carry out more meaningful comparisons with other results available in the literature (Zhang et al., 2010; Yao and Liou, 2012; Thomas et al., 2002). In such a view an inviscid flow model has been selected. Nevertheless, RANS simulations are currently under testing and will be presented in future works. It is based on a state of the art Finite Volume, cell-centered solver, which can treat both structured and unstructured grids. The convective fluxes of the RANS model are discretized by the classical Roe's approximated Riemann solver, which is a first order, monotonic scheme, blended by a centered approximation, i.e. Lax-Wendroff, producing a high-resolution scheme, completed by the entropy fix of Harten and Hyman and the van Leer flux limiter. Viscous and conductive fluxes are treated without any approximation. The time discretization can exploit first and second order accurate solutions, within a wide choice of explicit multi-step Runge-Kutta schemes, up to 5 stages. Options for local time-stepping, residual smoothing, dual time-stepping for unsteady problems and multi-grid to speed-up the convergence of the simulation are also available. Convergence analyses and capabilities of the solver in aeroelastic applications can be found in (Romanelli et al., 2012, 2010).

7. Sample applications

7.1. Two degrees of freedom typical section

The CTRNN is here applied to a plunging and pitching NACA 64A010 profile, at $M_\infty = 0.8$, in plain air (Thomas et al., 2002; Yao and Liou, 2012). A schematic representation of the system is depicted in Figure 2, with the related non-dimensional equations of motion given by Eq. (55):

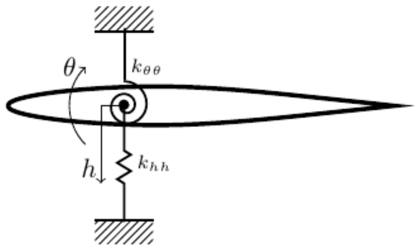


Figure 2: Two degrees of freedom typical section.

$$\begin{bmatrix} 1 & x_\theta \\ x_\theta & r_\theta^2 \end{bmatrix} \begin{Bmatrix} h''/b \\ \theta'' \end{Bmatrix} + \begin{bmatrix} (\omega_h/\omega_\theta)^2 & 0 \\ 0 & r_\theta^2 \end{bmatrix} \begin{Bmatrix} h/b \\ \theta \end{Bmatrix} = \frac{(V^*)^2}{\pi} \begin{Bmatrix} C_L \\ 2\, C_{M_{EA}} \end{Bmatrix}, \tag{55}$$

where a $.'$ indicates the differentiation with respect to the structural non dimensional time $\tau_s = \omega_\theta t$, x_θ and r_θ are the non dimensional static unbalance and moment of inertia respectively and ω_h and ω_θ the uncoupled natural circular frequencies of the mechanical system. The two degrees of freedom are the airfoil plunge h and pitch θ and the bifurcation parameter is represented by the reduced velocity, defined as:

$$V^* = \frac{V_\infty}{\omega_\theta\, b \sqrt{\mu}}, \tag{56}$$

being b the semi-chord and $\mu = \dfrac{\mathrm{m}}{\pi \rho_\infty b^2}$ the fluid to mass ratio. The model can be written in the compact matrix form:

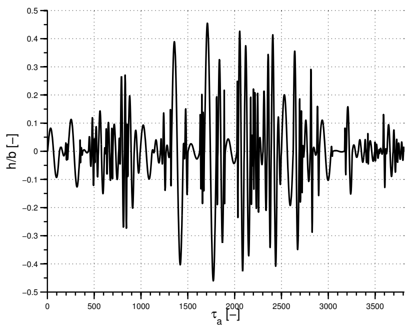
$$\mathbf{M}\mathbf{u}'' + \mathbf{K}\mathbf{u} = \mathbf{f}. \tag{57}$$

The parameters of the case here considered are reported in Table 1.

x_θ	r_θ^2	ω_h/ω_θ	μ
0.25	0.75	0.5	75

Table 1: Adimensional parameters of the typical section case.

The training signal is designed without considering any coupled structure, so that the generated trained response is derived from purely aerodynamic simulations, with imposed boundary conditions capable of catching all of the amplitudes and frequencies of the generalized forces associated to pitch/plunge motions. Such training signals are generated by imposing the maximum amplitude and reduced frequency allowed in the reduced order model representation, therefore their value is taken randomly during its generation. No steady state signals have been considered in this work since we are only interested in computing unsteady solutions, such as LCOs. An example of the adopted training


(a)

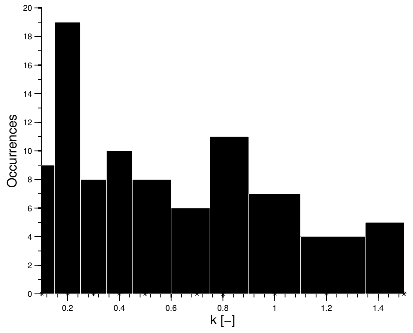

(b)

Figure 3: Sample of signal considered in the training phase. Figure 3(a) shows the related time history, while Figure 3(b) illustrates the frequency range covered by the signal.

signal is shown in Figure 3(a), while the frequency range excited is reported in Figure 3(b). The signals are expressed as a function of the aerodynamic reduced time $\tau_a = V_\infty t/c$, while the reduced frequency is in this case defined by $k = \omega c/V_\infty$.
The training data, composed by $N_t = 3500$ samples, is generated by AeroFoam in about 24 [h], with a physical time step of $\Delta t = 2 \cdot 10^{-3}$ [s]. The adopted two dimensional mesh is of a structured C-type, with 12200 cells, therefore 48800 unknowns. All of the computations have been carried out on a single processor of a computer, with an Intel® Core™ 2 Duo CPU with a frequency of 2.93 GHz. A convergence analysis enables a comparison of the accuracy and the computational time required for the training procedure of the CTRNN. For an easier interpretation of the following converged results, we call A the CTRNN characterized by $\mathrm{n}_x = 2$, $\mathrm{n}_h = 4$, so that $\mathrm{n}_\theta = 24$, B the network with $\mathrm{n}_x = 3$, $\mathrm{n}_h = 5$, $\mathrm{n}_\theta = 40$ and C the network with $\mathrm{n}_x = 5$, $\mathrm{n}_h = 8$, $\mathrm{n}_\theta = 96$. Of course all the networks considered have the same number of inputs (h, θ); therefore, m = 2. These three examples have been chosen since they present some difference in their behaviour. The convergence procedure on the order of the CTRNN has been performed starting from the smaller model possible (n_x, $\mathrm{n}_h = 2$) and then increasing the model order until a satisfactory accuracy has been obtained. Then the maximum number of generations allowed to the GA has been fixed to 200, starting from random synaptic weights, while the maximum number of iterations of the LM algorithm has been limited to 300, with a converged cost function threshold value set at 10^{-4}. Both the input and the output have been normalized, so to appropriately weight the fitting errors. The related trained results are resumed in Tables 2, 3 and 4, which compare the overall performances associated to the different ways of computing the Jacobian matrix within the LM phase, i.e: finite differences, analytical and the AD based procedure previously presented.
They clearly show that finite differences are quite inefficient with respect to ana-

Jacobian matrix computational method	Computational time	Converged F
Finite differences	6 [h]	$\mathcal{O}(10)$
Analytic	3 [h] 20 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	2 [h] 58 [min]	$\mathcal{O}\left(10^{-1}\right)$

Table 2: Convergence properties for the CTRNN A ($\mathrm{n}_x = 2$, $\mathrm{n}_h = 4$, $\mathrm{n}_\theta = 24$).

Jacobian matrix computational method	Computational time	Converged F
Finite differences	Not converged	$[-]$
Analytic	4 [h] 03 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	3 [h] 51 [min]	$\mathcal{O}\left(10^{-2}\right)$

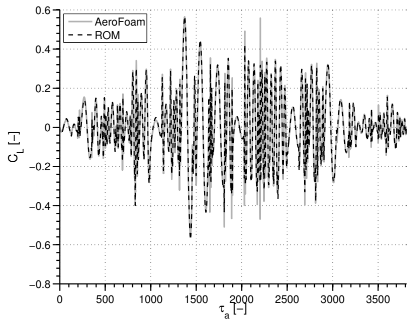
Table 3: Convergence properties for the CTRNN B ($\mathrm{n}_x = 3$, $\mathrm{n}_h = 5$, $\mathrm{n}_\theta = 40$).

Jacobian matrix computational method	Computational time	Converged F
Finite differences	Not converged	$[-]$
Analytic	5 [h] 34 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	5 [h] 12 [min]	$\mathcal{O}\left(10^{-2}\right)$

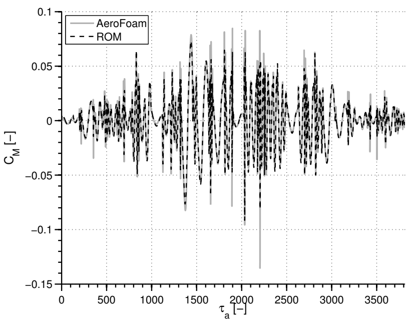
Table 4: Convergence properties for the CTRNN C ($\mathrm{n}_x = 5$, $\mathrm{n}_h = 8$, $\mathrm{n}_\theta = 96$).

lytical derivatives, up to missing convergence within the fixed maximum number of iterations allowed. The AD training based on a Taylor series of degree $d = 6$ provides results close to those of its analytic counterpart, even with somewhat shorter execution times, probably because of the high efficient AD algorithms employed, especially in the evaluation of the Jacobian matrix dynamics, since its computation is not carried out explicitly but the function evaluations are exploited with this aim by the AD tool directly (Griewank and Walther, 2008). Comparing the results of networks A, B and C, it can be seen that the second CTRNN is the one showing the best trade-off between accuracy and required computational time, an outcome of the training results obtained with network B is shown in Figure 4.

It can be seen that, because of the wide frequency content of the training signal the computational training time is quite high. In fact the literature presents significantly shorter times for similar trainings including a structural coupling (Zhang et al., 2010; Yao and Liou, 2012). That should come without any surprise, since the structure acts both as a signal enhancing filter at its characteristic frequencies and as a low pass filtering beyond them. The latter action in particular is of great help in cancelling the high frequency part of the training, which is the one affecting the most of the not so low matching error and needed training signal length. Therefore the direct identification of the aeroelastic system should reduce the training time, and different test are currently

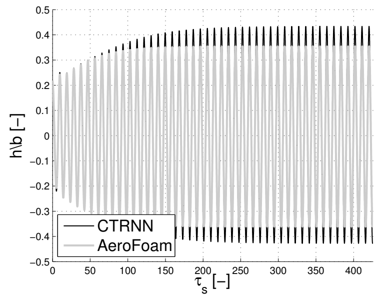


(a)



(b)

Figure 4: Sample of CTRNN output at the end of the training phase. Figures 4(a) and 4(b) 2) show the time histories of lift and aerodynamic moment coefficients respectively.



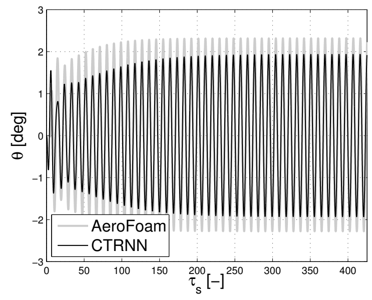


Figure 5: LCO motion obtained by the CTRNN, $V^* = 0.739$. The maximum reduced frequency considered in the training phase is $k = 1.5$.

under consideration to prove this fact. Nevertheless we accept to pay the related simulation costs in order to be able to exploit the freedom afforded by a well trained CTRNN in making it possible an easier a posteriori coupling to a wider range of structures.

Then we couple the aerodynamic ROM to the structure, so to build the nonlinear aeroelastic system represented by Eq. (58).

$$\mathbf{M}\mathbf{u}'' + \mathbf{K}\mathbf{u} = \mathbf{f}, \qquad \mathbf{f} = \mathbf{B}\mathbf{W}^c\mathbf{x}, \qquad \dot{\mathbf{x}} = \mathbf{W}^x\mathbf{\Phi}\left(\mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}\right), \tag{58}$$

with $\mathbf{B} = \dfrac{(V^*)^2}{\pi}\mathrm{Diag}\left(1,\, \dfrac{1}{2}\right)$. Setting our bifurcation parameter to $V^* = 0.739$, we can then compare the LCO obtained with AeroFoam and CTRNN-B, in both amplitudes and frequency. The related results are shown in Figure 5.

As it can be clearly seen, the LCO frequency is correctly predicted to be $k_{\mathrm{LCO}} = 0.2077$, while both the plunge (overestimated) and pitch (underestimated) show a rough 10% error with respect to CFD-based simulations. It

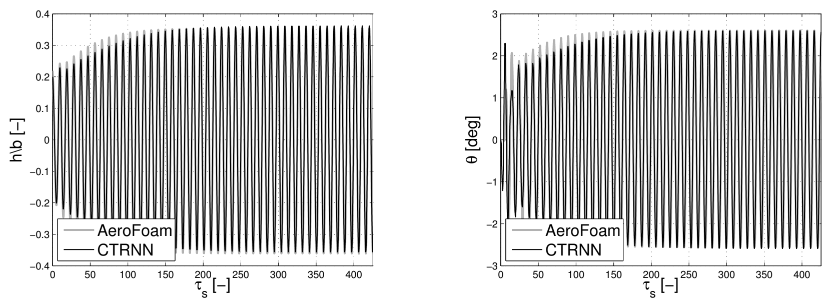


Figure 6: LCO motion obtained by the CTRNN at $V^* = 0.739$. The maximum reduced frequency considered in the training phase is $k = 3$.

should be remarked also that neither the network A nor C have been able to produce better results.

In order to obtain a CTRNN capable of adequately predicting LCO amplitudes and frequency, the frequency range of the training signal has been widened, so to take into account reduced frequencies up to $k = 3$, with quite a sizeable increase of the training time. However the move proves to be an appropriate one, as it can be verified from Figure 6 which shows a precise matching with the reference solution.
In fact it can be seen that the already good frequency match is safeguarded and the correct reference LCO amplitudes $h/b = 0.34\,[-]$, $\theta = 2.6\,[\text{deg}]$ are fully recovered. The aeroelastic ROM response has been obtained by an implicit, A-L stable method, with tunable numerical dissipation (Masarati et al., 2001), using a time step $\Delta t = 8 \cdot 10^{-3}\,[\text{s}]$, much larger than the one adopted during its training. Therefore we have proved that with the proposed CTRNN formulation, correct responses can be obtained even with larger time steps than the one adopted in the training phase. So the aeroelastic simulation time ends in a considerable time saving, the full parent CFD simulation requiring about 40 [h] to develop an LCO solution, against a ROM time of only 22 [s].

Furthermore, it should be remarked that also the dynamic order of the network is relevant in the prediction of a correct LCO, as it can be seen from Figure 7, whereas network A has not been able to correctly reproduce the LCO amplitudes and frequency. This result can be ascribed to too a low order of the model, while networks B and C seems to capture a correct LCO, because their higher number of states and neurons is more adequate to represent such a nonlinear response.
Finally the PCM option has been verified at the same $V^* = 0.739$ on the base of a reference to the two following cases:

- Case A: varying h_0/b, fixed $T_0 = 0.09\,[\text{s}]$,
- Case B: fixed $h_0/b = 0.34$, varying T_0,

where h_0/b is the initial value of the first state and T_0 the initial guess of the

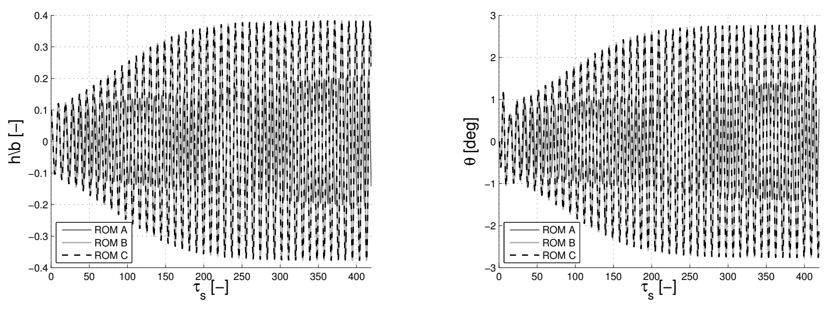


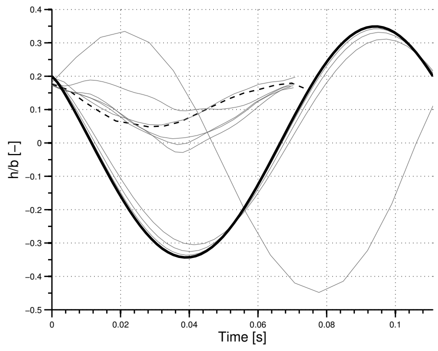
Figure 7: LCO motion obtained by CTRNNs with different dynamic order, $V^* = 0.75$. As can be seen, increasing its order, the network recovers the CFD-predicted LCO response.

unknown period. The results, obtained with a number of time intervals increasing from 16 to 66 and $\beta = 0.4$, are reported in the Tables 5 and 6.

Case A			
h/b imposed	LCO amplitude h/b	LCO amplitude θ [deg]	LCO period [s]
0.1	0.1	0.7	0.1111
0.15	0.34	2.6	0.1111
0.2	0.34	2.6	0.1111
0.3	0.34	2.6	0.1111
0.33	0.34	2.6	0.1111

Table 5: LCO sensitivity analysis with respect to the value of the fixed h/b variable.

It should be noticed that the PCM shows a significant robustness against initial guesses significantly far away from the converged solutions, in terms of both amplitude and LCO period. An example of such an assertion can be seen in Figure 8, with the converged solution being the thickest line, while the thinner ones represent intermediate Newton-Raphson iterations and the initial guess is represented by the dashed line.
At this point we are able to vary the reduced velocity to determine the LCO envelope diagrams, shown in Figures 9 and 10.
The results obtained through the direct time integration of the ROM and the PCM are compared to those obtained through the fine AeroFoam model, along with analyses presented in (Yao and Liou, 2012), plotted here with a dashed line. The envelope computed by the PCM exploits a continuation method using the converged result at the previous reduced velocity as the initial guess for the new reduced velocity. The good agreement provided by the different methods can be spotted with a simple glance. Finally let us consider the computational time required by the CFD-based and ROM-based analyses, presented in Tables 7 and 8.



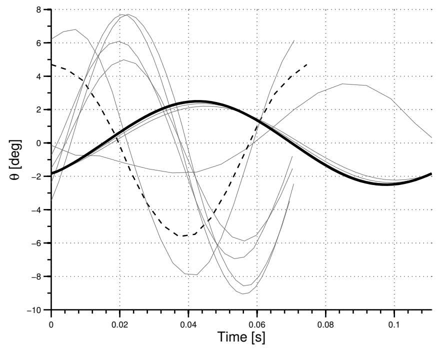
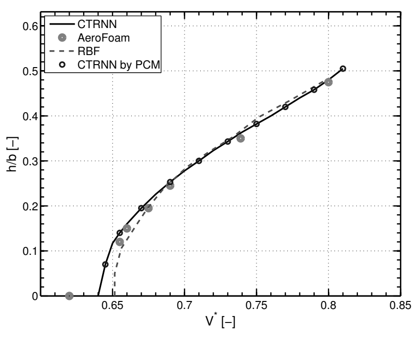


Figure 8: LCO computation by PCM. The first component of the state $h_0/b = 0.2$ is maintained fixed, while the initial guess on the LCO period is $T_0 = 0.075$ [s].



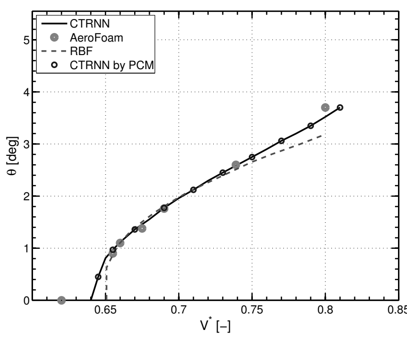
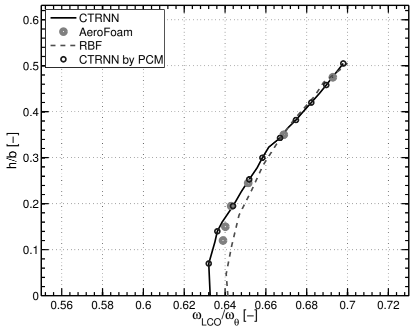


Figure 9: Amplitudes trends for the typical section test case.



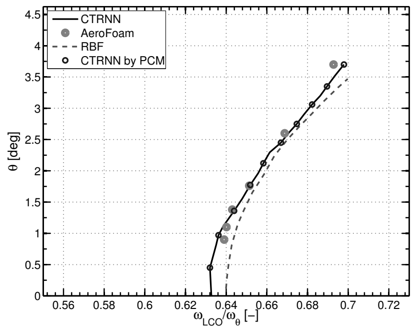


Figure 10: Frequencies trends for the typical section test case.

Case B			
Initial guess of T	LCO amplitude h/b	LCO amplitude θ [deg]	LCO period [s]
0.05		Not converged	
0.08	0.34	2.6	0.1111
0.1	0.34	2.6	0.1111
0.15	0.34	2.6	0.1111
0.16	0.34	2.6	0.1112
0.163	0.345	2.6	0.1113
0.175		Unstable LCO	
0.2		Unstable LCO	

Table 6: LCO sensitivity analysis with respect to initial guess of the period.

Computational time required for seven time accurate simulations	1 [h] 12 [min] + 42 [h] 42 [min] + 42 [h] 02 [min] + 41 [h] 00 [min] + 41 [h] 00 [min] + 39 [h] 10 [min] 38 [h] 40 [min]
Total time required	244 [h] 06 [min]

Table 7: Required computational time for the CFD-based generation of the LCOs envelope.

It should be remarked that the PCM has been employed in the computation of 10 points of the envelope diagram, but required the same time as 20 simulations through a direct time integration. It should nonetheless be noted that most of the overall time spent for the PCM calculations is to be ascribed to the determination of the very first LCO, initialized with rough estimates of the converged solution. Proceeding to the determination of the following LCO by varying the bifurcation parameter in a continued mode required far shorter solution times, thus making the PCM a viable alternative to direct time integration. Nevertheless the two methods should not be seen as competitors but as complementary tools in the most awkward phase of the search of a first solution, to be followed by much more effective continued scans over the range of the bifurcation parameters of interest. It should be remarked also that Table 6 shows that two unstable LCOs are predicted by the PCM. Such a result has been dismissed

Generation of the training input-output data pair by AeroFoam	24 [h] 01 [min]
Training time	3 [h] 50 [min]
Envelope simulation time (20 points)	20 [min]
Total time required	28 [h] 11 [min]

Table 8: Required computational time for the ROM-based generation of the LCOs envelope.

x_θ	r_θ^2	ω_h	ω_θ	μ	c	l
0.0035	1.036	21.01 [rad/s]	32.72 [rad/s]	4284	0.4064 [m]	0.8128 [m]

Table 9: BACT wing data, reported from (Wasnak, 1996).

by increasing the discretization points, up to 160, so confirming the comment previously made about the higher precision needed for a reliable determination of Floquet characteristic values.

7.2. BACT wing

The other test case considers the wing of the Benchmark Active Control Technology (BACT) (Wasnak, 1996; Rivers et al., 1992). It is a rigid, rectangular wing with a NACA 0012 airfoil section, pitching around its mid chord axis, which is mounted on a device called Pitch And Plunge Apparatus (PAPA), designed to permit the motions suggested by its name. The BACT wing has a dynamic behaviour very similar to the classical two DOFs typical section previously shown in Figure 2. The main differences are related to a more complex, three dimensional, aerodynamic behaviour and to some high frequency structural modes, brought in by a not truly rigid wing and PAPA. Because of the significant frequency separation between the rigid pitch/plunge motions and those higher modes, it is nevertheless viable to approximate the BACT wing as a two DOFs system. Such an assumption has been verified through more accurate investigations of the vibration properties of the BACT-PAPA structure, whereas the lowest frequency of any deformable mode was found to be more than six times higher than the ones of the rigid pitch/plunge modes (Wasnak, 1996). The parameters defining the BACT system are presented in Table 9, where, calling l the wing span, it is possible to notice a mass ratio, $\mu = \frac{\mathrm{m}}{\pi \rho_\infty b^2\, l}$, typical of heavy wing sections of the turbomachine kind (Bisplinghoff et al., 1955).
The bifurcation point has been computed through a linear Nastran $\mathrm{p}-k$ flutter analysis, based on a doublet lattice aerodynamics at $M_\infty = 0.8$. A check of the obtained results against the corresponding flutter test (Rivers et al., 1992) shows a calculated reduced flutter velocity of $V_f^* = 0.64$, somewhat higher than its experimental counterpart of $V_{f_{\mathrm{exp}}}^* = 0.595$ and a matching of the related reduced frequency at $k_f = \omega_f b / V_f = 0.0193$. It should be remarked that the accurate estimation of the bifurcation frequency, combined with the not so bad flutter velocity provided by a linear flutter analysis, is often of great help in effectively starting any following determination of an envelope of possible LCOs.

Then, because of the mentioned similarities between the present and previous numerical verifications, we can proceed to illustrate the obtained results with a somewhat faster pace. Thus, the adopted training signal, shown in Figure 11, is, once more, a hybrid sinusoidal time history, with amplitude and frequency range content adapted to the case under consideration. Once more, a structured C-type mesh has been adopted, with $5 \cdot 10^4$ cells and $25 \cdot 10^4$ unknowns. The training CFD simulation is carried out through a fifth order RK

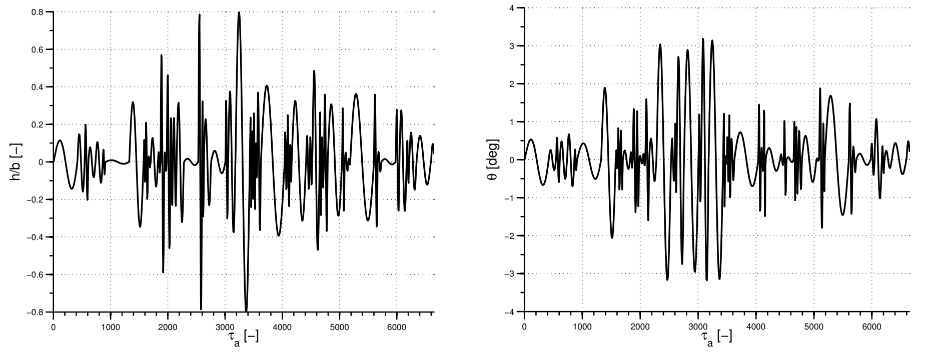


Figure 11: Example of training signal considered in the BACT test case, the maximum reduced frequency considered in this case is $k = 0.3$.

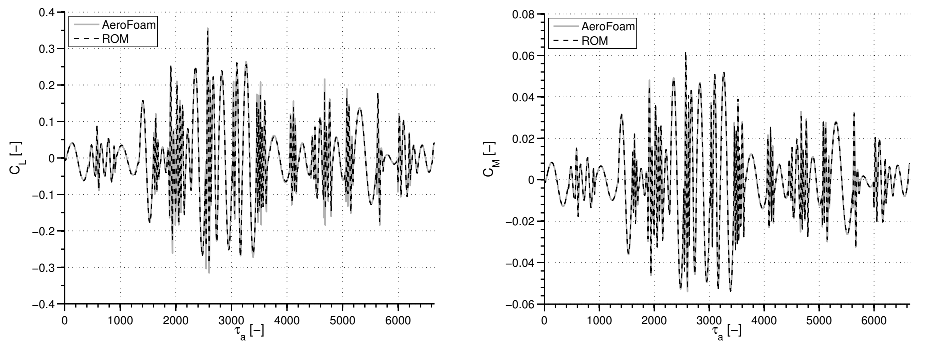


Figure 12: Example of training results relative to the BACT test case.

scheme (maximum CFL = 3), with a physical time step of $\Delta t = 2.5 \cdot 10^{-3}$ [s] on a signal length of 5 seconds, i.e. $N_t = 2000$ samples. Using both of the available computer processors, the computational time required for the training is 26 [h] 13 [min] . After a convergence analysis performed in the same way of the previous section, the selected dynamic order of the adopted CTRNN is $\mathrm{n}_x = 3$ and $\mathrm{n}_h = 5$, which guarantees a sufficient level of accuracy to maintain a limited training time. Also in this case the input dimension is equal to m = 2, since the input is again represented by the pitch and plunge DOFs. It should be noted that even if the high-fidelity aerodynamic model is much more complex than the previous test case, a CTRNN of similar order will be able to capture its essential nonlinear behaviour. This fact is promising in view of much more realistic applications of the presented ROM. The results of the related training are displayed in Figure 12. A comparison between the different trainings is shown in Table 10. A very good fit has been obtained, with a small error for both

Jacobian matrix computational method	Computational time	Converged F
Analytic	3 [h] 58 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	3 [h] 34 [min]	$\mathcal{O}\left(10^{-2}\right)$

Table 10: Convergence properties of the present CTRNN for the BACT wing case ($n_x = 3$, $n_h = 5$, $n_\theta = 40$).

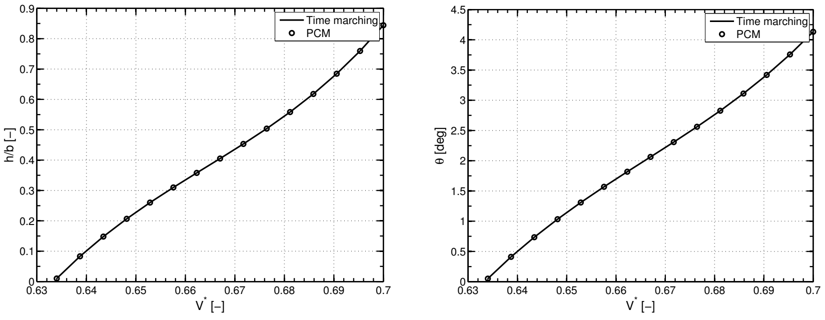


Figure 13: Amplitudes trends for the BACT wing test case.

of the coefficients, C_L and C_M, followed by the evaluation of the related LCO envelopes, mutually verified through continued PCM and simulated time marching response solutions. Figures 13 and 14 depict the related LCO trends, while Table 11 shows the computational time required for the ROM determination and the evaluation of LCO envelopes.

8. Concluding remarks

The paper has detailed the development of a nonlinear aeroelastic Reduced Order Models, built upon input-output data generated by high fidelity CFD

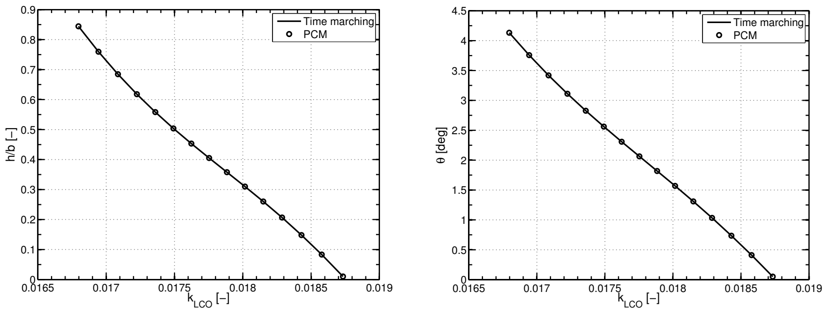


Figure 14: Frequencies trends for the BACT wing test case.

	Time marching	Periodic collocation method
Generation of the training input-output data pair by AeroFoam	26 [h] 23 [min]	
Training time	3 [h] 34 [min]	
Envelope simulation time (15 points)	12 [min]	16 [min]
Total time required	30 [h] 09 [min]	30 [h] 13 [min]

Table 11: Required computational time for the ROM-based generation of the LCOs envelope.

simulations. The proposed ROM is structured as a Continuous Time Recurrent Neural Network, trained with a procedure based on Automatic Differentiation. It is thus possible to simulate any aeroelastic response of interest without being constrained to a small time step size except where required for numerical stability and precision reasons. Moreover, a CTRNN allows the determination of Limit Cycle Oscillations through the application of an adaptive Periodic Collocation Method in the time domain, permitting an easy Floquet analysis of their stability. The same freedom in time step adaption also makes it easier to determine LCOs by continued simulations in time, allowing control of the related integration error to avoid the masking of possible LCOs caused by any undue numerical damping. The efficiency of the presented ROM procedure has the potential of making it possible a more extensive adoption of nonlinear aeroelastic analyses in early stages of aircraft design, maintaining adequate accuracies, close to the solutions provided by the far more costlier high fidelity CFD calculations.