Politecnico di Milano

FACOLTÀ DI INGEGNERIA INDUSTRIALE
Corso di Laurea Magistrale in Ingegneria Aeronautica

Reduced order aeroelastic models through continuous time neural networks

Tesi di Laurea di:
Andrea Mannarino
Matricola 767025

Relatore:
Prof. Paolo Mantegazza

Anno Accademico 2011-2012

Contents

List of Figures

List of Tables

Abstract

During the design and certification phases of an airplane, system stability and well bounded responses to various forcing actions imposed by the regulations must be ensured. Since the modern commercial transport aircrafts operate routinely in a transonic flow regime, the need to simulate such imposed conditions during the first design phases requires the use of methods that can simulate accurately the case-specific aerodynamic conditions. Therefore, nowadays state of the art Computational Fluid Dynamic (CFD) codes are employed. But since the simulation times are still too high, these methods are still little used in practical industrial applications, especially in the study of a coupled structural and aerodynamic response, limiting such simulations only near the various cruise conditions of the aircraft. On the other hand, the aerodynamics schemes employed in standard aeroelastic analyses, i.e. the Doublet Lattice Method, are not capable of simulating accurately the nonlinearities present in a transonic flow field, such as shocks, or possible large flow separations.

In this work a continuous time, neural network based, reduced order modeling techinque is proposed with the aim to reduce the computational burden of such analyses, leading to the generation of models whose simulation is orders of magnitude faster than the standard high fidelity CFD simulations. Such models have the feature of being able to capture the main nonlinearities present in the high order CFD simulation, allowing the analysis of stability issues and forced responses of an aeroelastic system characterized by a nonlinear behaviour, mantaining the same accuracy of the high order system. In the literature numerous applications in reduced order modeling of aeroelastic systems with CFD-based aerodynamics can be found. Different continuous time models have been proposed, but no one is based on neural network design, even if their capability of approximating nonlinear dynamic systems from the universal approximation theorem makes their application attractive. More recently, discrete time recurrent neural network have been employed in the study of the flutter of an aeroelastic typical section in a transonic flow condition. However, continuous time neural networks are a relatively new mathematical structure present today in the literature, both in reduced order modeling and system identification frameworks. Thus such continuous time, neural network based, reduced order modeling technique applied in the study of an aeroelastic system can be considered a new approach, probably never applied before in this field of study.

Particular emphasis will be given to aeroelastic stability analyses, where the nonlinear systems framework, beyond admitting responses asymptotically stable or unstable, is enriched by the presence of persistent periodical neutrally stable solutions, called Limit Cycle Oscillation (LCO). The neural network based Reduced Order Model (ROM) is trained by advanced nonlinear optimization algorithms, since the high nonlinear nature of the problem of interest, employing the modern tools of the Automatic Differentiation (AD) during the training of the network.

The proposed ROM is finally tested in the identification and prediction of the unsteady aerodynamic loads acting on a pitching airfoil and in the simulations of various LCOs of a pitching and plunging aeroelastic typical section and the BACT wing, comparing the results in terms of accuracy and computational time with the CFD results and those present in the literature. The LCO behaviours will be computed through a time-marching scheme and a direct periodic collocation method in the time domain, the latter being a relatively new method in the computation of periodic responses of aeroelastic systems.

Keywords: Computational Aeroelasticity, Reduced Order Model, Limit Cycle Oscillation, Automatic Differentiation, Continuous Time Recurrent Neural Network, Periodic Collocation Method in the Time Domain.

Sommario

Durante le fasi di progettazione e di certificazione di un velivolo, la stabilità e ben definite risposte del sistema devono rispettare i vincoli imposti dalle norme vigenti. Poiché i moderni aeromobili da trasporto commerciale operarano abitualmente in un regime di volo transonico, caratterizzato dalla presenza di onde d'urto, la necessità di simulare queste condizioni durante le prime fasi progettuali richiede l'uso di metodi che devono simulare con precisione le specifiche condizioni aerodinamiche, caso per caso. Pertanto oggi sono impiegati in questo campo i moderni metodi della Fluidodinamica Computazionale, o Computational Fluid Dynamics (CFD), basati su una discretizzazione a volumi finiti del problema aerodinamico. Ma poiché i relativi tempi di simulazione sono ancora troppo elevati, questi metodi sono ancora poco utilizzati in pratiche applicazioni industriali, soprattutto nello studio della risposta accoppiata di struttura e aerodinamica, limitando tali simulazioni solo vicino alle varie condizioni di crociera del velivolo. D'altra parte, i metodi di calcolo aerodinamico impiegati in ben consolidate analisi aeroelastiche, per esempio il Metodo a Reticolo di Vortici instazionario, non sono in grado di simulare con precisione le non linearità presenti in un campo di moto transonico, quali urti o eventuali separazioni di grosse dimensioni.

In questo lavoro una rete neurale a tempo continuo è utilizzata per generare un modello di ordine ridotto del sistema aerodinamico, e viene dunque proposta con l'obiettivo di ridurre l'onere computazionale di tali analisi aeroelastiche, portando alla generazione di modelli la cui simulazione è ordini di grandezza più veloce dei metodi sviluppati allo stato dell'arte dalla Fluidodinamica Computazionale. Tali modelli hanno la caratteristica di essere in grado di catturare le non linearità principali presenti nella simulazione CFD, consentendo l'analisi dei problemi di stabilità e di risposte forzate di un sistema aeroelastico caratterizzato da un comportamento non lineare, mantenendo la stessa precisione del sistema di ordine elevato, caratterizzato da un numero molto maggiore di incognite. In Letteratura si possono trovare numerose applicazioni nella modellazione dei sistemi di ordine ridotto in aeroelasticità con modelli di aerodinamica basati sulla moderna CFD. Diversi modelli a tempo continuo sono stati proposti, ma nessuno è basato su reti neurali, anche se la loro capacità di approssimare sistemi dinamici non lineari dal teorema di approssimazione universale rende la loro applicazione molto attraente. Le reti neurali a tempo continuo sono una nuova struttura matematica oggi presente nella Letteratura, sia per quanto riguarda i modelli di ordine ridotto sia metodi dell'identificazione di sistemi. Dunque una rete neurale ricorrente a tempo continuo, applicata come tecnica di generazione di modelli di ordine ridotto a sistemi aeroelastici, può essere considerata un approccio nuovo, probabilmente mai applicato in questo campo di studio.

Particolare attenzione verrà data all'analisi della stabilità di un sistema aeroelastico, dove nel quadro dei sistemi non lineari, la varietà delle soluzioni di un sistema non forzato è arricchita dalla possibile presenza di una risposta periodica neutralmente stabile, chiamata Ciclo Limite, o Limit Cycle Oscillation (LCO). Il modello di ordine ridotto (ROM) è allenato attraverso algoritmi di ottimizzazione non lineare, scelti data la forte natura non lineare del problema di interesse, utilizzando inoltre i moderni strumenti della differenziazione automatica (AD). Il modello di ordine ridotto proposto è infine messo alla prova nell'identificazione e la previsione di carichi aerodinamici instazionari che agiscono su un profilo alare in beccheggio e nelle simulazioni di cicli limite di una sezione tipica e dell'ala BACT, confrontando i risultati in termini di precisione e tempo di calcolo con la CFD e la Letteratura. I cicli limite sono calcolati mediante un metodo di integrazione numerica e da un metodo di collocazione periodica nel dominio del tempo, considerando quest'ultimo come un metodo relativamente nuovo nel calcolo di risposte periodiche di sistemi aeroelastici.

Parole chiave: Aeroelasticità Computazionale, Modelli di Ordine Ridotto, Cicli Limite, Differenziazione Automatica, Reti Neurali Ricorrenti a Tempo Continuo, Collocazione Periodica nel Dominio del Tempo.

Estratto in lingua italiana

Inizialmente, nel primo capitolo di questo lavoro le diverse tecniche relative alla costruzione di modelli di ordine ridotto, lineari e non, sono presentate, svolgendo una breve ricerca bibliografica sulla stato dell'arte di queste tecniche. In particolare vengono analizzate le caratteristiche di modelli basati sulla serie di Volterra, la quale si basa sulla generalizzazione non lineare della caratterizzazione di un sistema attraverso la propria risposta impulsiva, quelli basati sulla Decomposizione Ortogonale Propria, o Proper Orthogonal Decomposition (POD), la quale invece si basa su una rappresentazione modale dello stato di un sistema non lineare, dunque generalizzando i concetti ben conosciuti ed applicati nell'analisi strutturale di sistemi caratterizzati da un elevato numero di gradi di libertà, e l'approccio del Bilanciamento Armonico, o Harmonic Balance (HB) method, il quale infine permette di risolvere un sistema di equazioni differenziali ordinarie, in generale non lineare, attraverso un approccio alla Fourier, ovvero scomponendo a priori la risposta del sistema nell armoniche principali e andando a risolvere di conseguenza il problema algebrico non lineare risultante dalla sostituzione di tale soluzione nel sistema dinamico originario. Queste tre tecniche, ben conosciute ed utilizzate nell'ambito della generazione di modelli ridotti, sono confrontate con l'approccio seguito in questo lavoro, ovvero quello di modelli di ordine ridotto basati sulle capacità di una rete neurale di approssimare un generico sistema dinamico non lineare. Entrambe le formulazioni a tempo discreto e tempo continuo sono considerate in questa prima parte. Pregi e difetti dei diversi approcci sono discussi e confrontati fra loro, cercando di mettere in evidenza le potenzialità della rappresentazione del sistema attraverso una rete neurale dinamica. Viene dato anche un breve cenno riguardante i metodi della superficie di risposta (*response surface methods*) che sono stati impiegati con successo nella letteratura nell'ambito della stabilità di sistemi aeroelastici non lineari.

Nel secondo capitolo i principali fenomeni non lineari incontrati in analisi aeroelastiche, sia numeriche che sperimentali, sono introdotti. Particolare importanza viene data alle caratteristiche soluzioni di biforcazione di un generico sistema dinamico autonomo e non lineare, ovvero i cicli limite, o Limit Cycle Oscillation (LCO). Una descrizione teorica di questo fenomeno, che nella teoria dei sistemi nonlineari viene definita come biforcazione alla Hopf, viene fornita con brevi cenni all'analisi analitica necessaria per la determinazione di tale condizione su sistemi nonlineari di piccole e grandi dimensioni. Il fenomeno viene poi analizzato da un punto di vista ingegneristico, considerando i diversi casi presenti in Letteratura e cercando di darne una descrizione fenomenologica. I cicli limiti verificati su sistemi aeroelastici sono stati dovuti sia a non linearità presenti nella struttura sia nell'aerodinamica. Viene considerato che nelle condizioni di volo assunte in questo lavoro, ovvero quelle caratteristiche di un regime transonico, il principale effetto di non linearità è senza dubbio introdotto dall'aerodinamica, attraverso le onde d'urto che si sviluppano, si spostano o addirittura compaiono e scompaiono nel campo di moto. Il problema del ciclo limite viene poi analizzato seguendo due filoni. Il primo interpreta la soluzione di ciclo limite come un problema ai valori iniziali. Ovvero data una condizione iniziale, un modello non lineare che contiene al suo interno i diversi attrattori del sistema è in grado di convergere verso la soluzione di ciclo limite, ovviamente attraverso un metodo di integrazione numerica nel tempo. Viene anche specificato che in certi casi tale soluzione viene in qualche modo suggerita al sistema, forzandolo per un certo periodo con un movimento armonico per poi lasciarlo libero di evolvere verso la condizione di ciclo limite. La seconda strada invece definisce il problema della determinazione di un ciclo limite come un problema ai valori al bordo, in cui il modello viene forzato ad una soluzione periodica imponendo la periodicità stessa della risposta e imponendo una componente dello stato, fissando in questa maniera la fase del ciclo limite, dato che il sistema dinamico considerato è autonomo. Tale metodo è chiamato in Letteratura come Collocazione Periodica nel Dominio del Tempo, o Periodic Collocation Method (PCM) e permette di calcolare le soluzioni periodiche di un sistema non lineare a partire da una opportuna soluzione di primo tentativo. Il periodo viene suddiviso in intervalli, la cui ampiezza può variare durante le iterazioni in maniera tale da adattarsi alle non linearità presenti nel sistema. A questo

punto l'espressione implicita del sistema viene collocata sui diversi istanti di tempo considerati nella discretizzazione e pesata dall'ampiezza dell'intervallo temporale corrispondente. Il problema risultante è rappresentato da un sistema algebrico non lineare che viene risolto iterativamente da un opportuno algoritmo di ottimizzazione, nella fattispecie il metodo di Newton-Raphson, nelle incognite rappresentate dall'evoluzione dello stato durante il ciclo limite e dal periodo del ciclo stesso. La stabilità della soluzione di ciclo limite trovata viene infine valutata sfruttando i risultati della teoria di Floquet riguardante i sistemi dinamici periodici, quindi considerando gli autovalori della matrice di monodromia, la quale viene calcolata con un approccio che sfrutta la formulazione del problema precedente. Il metodo della collocazione periodica è molto più rigoroso dal punto di vista matematico rispetto all'integrazione diretta nel dominio del tempo, e può essere trovato in diverse fonti della Letteratura, ma presenta pochissime applicazioni nell'ambito dei sistemi aeroelastici.

Nel terzo capitolo il metodo di modellazione ridotta sviluppato in questo lavoro di tesi viene dettagliato. Innanzitutto vengono introdotti i modelli analitici utilizzati per rappresentare il moto di un fluido considerato come continuo deformabile, ovvero i modelli di Navier-Stokes e di Eulero. Dato che tali modelli, descritti da sistemi di Equazioni differenziali alle Derivate Parziali (PDE), non sono risolvibili analiticamente per le tipiche applicazioni ingegneristiche, esse vengono risolte numericamente attraverso il metodo dei volumi finiti, appartenenti alle moderne tecniche della CFD. Viene ricavata la forma integrale delle equazioni del moto fluido, che viene poi discretizzata nello spazio, ed inoltre vengono fatte alcune considerazioni su tale formulazione nel caso di un problema con un dominio deformabile, come nelle pratiche applicazioni che vedono coinvolta l'interazione fra il fluido e un corpo, generalmente anch'esso deformabile. Tale formulazione prende il nome di Arbitraria Lagrangiana Euleriana (ALE), ed è sostanzialmente una versione modificata delle equazioni del moto, in cui vengono tenuti in considerazione la velocità dei diversi nodi della griglia computazionale, ed il fatto che il volume della griglia si può modificare solo per effetto di tale velocità, imponendo la cosiddetta Legge di Conservazione Geometrica (GCL). Viene poi considerata la discretizzazione temporale delle equazioni del moto, con un breve riferimento ai vincoli che un metodo di integrazione esplicita deve soddisfare. Viene data una breve descrizione del software libero OpenFOAM, ed in particolare viene descritto il solutore aerodinamico utilizzato in questo lavoro, AeroFoam, il quale si appoggia alle librerie OpenFOAM per la gestione della griglia computazionale, per la gestione dei dati relativi alla soluzione numerica del problema e per la parti di pre e post-processing. Vengono brevemente descritte le pecularità di questo solutore aerodinamico, tra le quali l'utilizzo di tecniche di Multi Grid (MG), utilizzate per accelerare la convergenza alla soluzione stazionaria e del Dual Time Stepping (DTS), che invece permette di utilizzare dei passi di integrazione molto più laschi durante simulazioni instazionarie. Infine viene fornito un esempio di costruzione di un caso test con AeroFoam. La seconda parte di questo capitolo tratta invece nello specifico le tecniche di modellazione ridotta del problema aerodinamico, dato che la discretizzazione numerica attraverso tecniche CFD delle equazioni del moto porta a sistemi dinamici con un numero di stati nell'ordine di $10^5/10^6$. Sono considerate reti neurali ricorrenti a tempo continuo, attraverso una breve presentazione teorica e un confronto con reti ricorrenti a tempo discreto, il quale porta sostanzialmente alla conclusione che le reti a tempo continuo sono indipendenti dalla discretizzazione temporale che caratterizza il segnale di ingresso, mentre quelle a tempo discreto sono notevolmente influenzate da questo fatto, richiedendo che sia la fase di allenamento sia quella di applicazione vengano svolte con lo stesso tempo di campionamento. Per la generalità delle reti neurali a tempo continuo, queste vengono scelte sulla loro controparte a tempo discreto. Dato che nella letteratura nessuna particolare parametrizzazione della rete si è imposta, vengono analizzate tre differenti parametrizzazioni. Per ognuna di queste viene dettagliato il metodo di calcolo dello jacobiano, inteso come variazione dell'uscita della rete rispetto a variazioni dei pesi sinaptici della stessa, in quanto necessario per l'algoritmo di allenamento. Per quest'ultimo è stato scelto un metodo ibrido, che utilizza un algoritmo genetico per la fase di inizializzazione

dei pesi sinaptici e quello di Levenberg-Marquardt per il raffinamento dei risultati. Mentre la prima fase dell'algoritmo non richiede il calcolo dello jacobiano della rete, in quanto necessita solo della valutazione della cifra di merito definita a priori, nella seconda fase tale calcolo è richiesto, in quanto la soluzione ad ogni passo iterativo viene ricavata proprio attraverso la pseudo-inversa della matrice jacobiana. Viene infine considerata la generazione di opportuni segnali di allenamento in grado di eccitare la dinamica della rete necessaria per riprodurre nella fase di generalizzazione i fenomeni di interesse. Vengono discussi diversi tipi di segnale, regolari e non, e il loro metodo di generazione. Ulteriori considerazioni sul contenuto in frequenza e sulle ampiezze caratteristiche di tali segnali vengono fornite.

Nel quarto capitolo i diversi metodo utilizzati al giorno d'oggi nel calcolo delle derivate di funzioni definite in programmi algoritmici sono confrontati. Vengono dunque brevemente presentati i metodi alle differenze finite e la differenziazione simbolica, spesso utilizzati in applicazioni pratiche (soprattutto il primo) e vengono elencati i diversi vantaggi/svantaggi di tali approcci, in termini di accuratezza e complessità computazionale. Vengono poi dettagliati i principali algoritmi associati alla Differenziazione Automatica, o Automatic Differentiation (AD). I due principali modi di calcolo, il cosiddetto *forward mode* e il *reverse mode* vengono introdotti attraverso semplici esempi. Sostanzialmente la differenziazione automatica applica la regola di derivazione di una funzione composta ad una generica funzione definita in un generico codice di programmazione. Dualmente al programma di valutazione della funzione viene creato un programma che invece calcola i differenziali delle operazioni elementari definite nella funzione stessa. E' dunque possibile valutare in parallelo le derivate di una funzione insieme al suo valore, con costi computazionali comunque contenuti. Considerando la regola di derivazione di una funzione composta, nota anche come *chain rule*, il *forward mode* costruisce il valore della derivata partendo da sinistra, mentre l'opposto viene fatto dal *reverse mode*. Quest'ultimo metodo di calcolo si è mostrato particolarmente efficiente nel calcolo delle derivate di una funzione scalare dipendente da un numero elevato di variabili, guarda caso la definizione di una generica cifra di merito utilizzata in problemi di ottimizzazione. E' stato dimostrato che il costo computazionale di questo modo di differenziazione dipende dal numero di variabili dipendenti, e non da qualle indipendente. Di conseguenza il costo computazionale associato ad esso è sempre ben limitato, in particolar modo nei problemi di ottimizzazione. Dato che però questo modo percorre all'indietro la regola della derivazione della funzione composta, una traccia delle operazioni svolte durante la valutazione della funzione deve essere mantenuta in memoria per poter determinare il valore della derivata. Per questo motivo i moderni codici che implementano tali algoritmi permettono sempre di utilizzare la cosiddetta *traccia di valutazione*, dove le operazioni elementari che compongono la definizione della funzione sono registrate passo dopo passo. Tale metodo di calcolo delle derivate permette di ottenere dei risultati liberi dall'errore di troncamento incontrato applicando la tecnica delle differenze finite, con un costo computazionale ben più contenuto di quello necessario per svolgere le stesse operazioni con la tecnica della differenziazione simbolica. Questi modi vengono poi generalizzati al caso di funzioni a valori vettoriali. Vengono poi ripresi i concetti di rappresentazione locale di una generica funzione continua attraverso la serie di Taylor espansa attorno ad un punto di riferimento. Attraverso le tecniche fornite dalla AD il calcolo dei coefficienti di Taylor di vettori (con il *forward mode*) e matrici (con il *reverse mode*) risulta essere particolarmente efficiente. Viene poi proposto un metodo esplicito di integrazione nel tempo di sistemi di equazioni ordinarie, che utilizza proprio la rappresentazione in serie di Taylor del termine noto per avanzare la soluzione dal passo corrente a quello successivo. Alcuni semplici esempi vengono proposti per dimostrare le capacità di questo integratore di trattare sistemi lineari, non lineari e altamente non lineari. In particolare viene considerata l'influenza del termine di ordine più elevato nella serie di Taylor sull'accuratezza della soluzione, confrontata con quella ottenuta con un integratore standard. Infine, una procedura di allenamento della rete neurale a tempo continuo basata su tale metodo di integrazione viene dettagliata, permettendo così di sfruttare gli algoritmi della AD nel metodo presentato nel capitolo precedente, con lo scopo di

migliorarne l'efficienza.

Nel quinto capitolo il metodo di modellazione ridotta sviluppato viene messo alla prova considerando un problema di identificazione del modello di aerodinamica. Un profilo alare viene fatto oscillare in beccheggio in una condizione di volo transonica, in cui almeno un'onda d'urto è presente nel campo di moto. Tale condizione di lavoro è ovviamente non lineare per la presenza di un onda d'urto mobile sulla superficie del profilo. Il modello numerico di ordine elevato, rappresentato dal solutore aerodinamico AeroFoam viene tarato attraverso delle simulazioni aerodinamiche stazionarie, confrontando i risultati ottenuti con quelli presenti in Letteratura e di conseguenza scegliendo il numero di celle computazionali più adatto considerando il compromesso fra accuratezza e tempi di calcolo. Vengono poi considerati due diversi tipi di segnali di allenamento, e quest'ultimo viene svolto su tutte e tre le parametrizzazioni della rete neurale descritte in dettaglio nel terzo capitolo. I risultati dell'allenamento vengono confrontati in termini di tempo di calcolo e di accuratezza, rappresentata dal valore raggiunto dalla cifra di merito considerata per la convergenza al termine dell'allenamento. Per il calcolo della matrice jacobiana utilizzata dall'algoritmo di ottimizzazione di Levenberg-Marquardt vengono considerati tre tipi di calcolo: quello attraverso le differenze finite, quello attraverso lo sviluppo analitico di tale matrice e quello basato sugli algoritmi della differenziazione automatica. Questi tre approcci vengono confrontati sulla base del tempo di calcolo. La fase di generalizzazione invece considera degli ingressi di beccheggio puramente sinusoidali, di differente ampiezza e frequenza, e le prestazioni delle tre reti vengono confrontate in termini di accuratezza. Infine la capacità di generalizzazione viene provata attraverso una validazione incrociata, ovvero alle reti allenate con un certo segnale viene dato in ingresso il segnale di allenamento opposto e viceversa. Viene inoltre discusso il delicato argomento della scelta dell'ordine del modello dinamico, fornendo delle considerazioni euristiche.

Nel sesto capitolo un modello di ordine ridotto di un campo di moto transonico instazionario viene messo alla prova in due simulazione aeroelastiche, dove la dinamica stessa dell'aerodinamica è accoppiata con la dinamica strutturale. E' di particolare interesse lo studio di cicli limite dovuti alla nonlinearità introdotta dalle onde d'urto presenti nel campo di moto, che subiscono degli spostamenti relativamente grandi, se riferiti alla corda del profilo, ed inoltre compaiono e scompaiono dal campo di moto a seconda della particolare posizione del profilo. Vengono considerati come modelli aeroelastici la classica sezione tipica, spesso impiegata per validazione di codici CFD, e l'ala BACT, la quale presenta diversi dati di confronto nella Letteratura. Viene dunque inizialmente considerata la fase di allenamento, in cui vengono utilizzati i segnali introdotti nel terzo capitolo. Tutte e tre le parametrizzazioni di rete neurale vengono considerate, confrontando i risultati dell'allenamento in termini di tempo computazionale richiesto e ordine di convergenza della cifra di merito considerata. Le reti neurali ottenute dalla fase di allenamento vengono poi messe alla prova nella fase di generalizzazione, in cui due segnali di ingresso mai visti durante l'allenamento vengono forniti alle diverse reti, confrontando il grado di fedeltà con i risultati ottenuti con la CFD. Fatto questo, il modello ridotto di aerodinamica viene accoppiato con il modello di dinamica strutturale di un profilo alare rigido connesso a terra da degli elementi deformabili. Vengono considerate come riferimento le risposte aeroelastiche ottenute dal solutore aerodinamico AeroFoam, in cui il problema è risolto interfacciando le relative discretizzazioni del modello strutturale ed aerodinamico e risolvendo in iterativamente le equazioni della dinamica strutturale e quelle dell'aerodinamica, dato che il movimento strutturale entra come condizione al contorno al problema aerodinamico, ed i carichi aerodinamici entrano come forzanti nel problema strutturale. Dai dati presenti in letteratura, vengono considerati dei casi in cui il sistema aeroelastico è sottoposto a cicli limite, dove la risposta è caratterizzata da delle oscillazioni periodiche non smorzate. Vengono fatte delle simulazioni dei modelli aeroelastici con modelli di ordine ridotto di aerodinamica. La condizione di ciclo limite viene in qualche modo forzata con l'approccio dettagliato nel secondo capitolo. Vengono comunque scelte delle ampiezze e frequenze diverse da quelle del ciclo limite di interesse. Mostrato che i modelli ridotti

sono in grado di riprodurre tale condizione con un sufficiente livello di accuratezza, viene studiato l'inviluppo di ampiezze e frequenze del ciclo limite al variare della rigidezza torsionale della sezione tipica. I risultati ottenuti vengono confrontati con quelli della CFD e quelli presenti nella Letteratura. Tale inviluppo viene inoltre calcolato attraverso il metodo di collocazione periodica nel dominio del tempo, descritto nel dettaglio nel secondo capitolo. Tale metodo si è mostrato particolarmente efficace nel calcolo di condizioni di ciclo limite, anche con periodi di primo tentativo relativamente distanti dalla reale soluzione. Questo approccio può essere considerato una novità nell'ambito del calcolo di risposte periodiche di sistemi aeroelastici ridotti, dato che al momento le tecniche più utilizzate sono quelle della simulazione diretta nel dominio del tempo e recentemente anche il bilanciamento armonico nel dominio della frequenza. Vengono infine tratte alcune conclusioni sulle capacità dei modelli ridotti nel riprodurre la condizione di ciclo limite, condiderando inoltre la loro capacità di simulazione senza forzare tale condizione.

Chapter 1

Introduction

1.1 Reduced Order Modeling

In recent years, the huge increase in computational power has allowed to solve problems once impossible to approach. Nowadays, systems of Partial Differential Equations (PDE), representative of complex physics, can be discretized and solved with numerical algorithms on massive computational systems. Simulations of systems with millions (or even tens of millions) of unknowns has become relatively commonplace [1].

However, such massive simulations require an adequate computational power to be afforded. Moreover, a direct numerical simulation of the related system of PDE is often insufficient to understand a physical phenomenon in depth. Reduced Order Models (ROMs) are a set of theoretical and numerical structures that permits to reconstruct and represent a generic nonlinear system with a small number of Degree Of Freedom (DOF). They thus allow a simpler formulation of a nonlinear problem, gaining both a deeper understanding of the physical phenomenon and an uttermost saving of simulation time. Of course, a simpler representation of a system allow also an effective formulation of the various analysis that have to be performed. In brief, a ROM permits to represent the nonlinear, dynamic behaviour of a system, once it is able to capture the main fundamentals dynamic embedded in the system itself.

ROMs have progressively gained space in the analysis of physical system, since nowadays numerical simulations present two main problems:

- As previously said, direct simulation can provide a detailed response history of the field variables, but such results may not help the user in gaining an increased level of understanding of the essential physics of the problem;

- In the absence of massive computational resources, the simulation of large-scale problems remains unpracticable to be used in various design phases, such as stability evaluation, control system design and optimization.

Considering the aeronautical world, an example of direct numerical simulation of a PDE system is of course the Computational Fluid Dynamics (CFD): it permits an accurate simulation of a turbulent, viscous flow field in the three-dimensional space; it is very useful for a large span of the experimental results, but its computational cost is still too high to be used during a design phase of an aircraft, since there is the need of many fast simulation. Moreover the analyses that has to be performed often do not require a point valued solution available in the whole domain: for example the load evaluation over a body immersed in a fluid stream requires only the knowledge of the solution on the body itself.

Reduced order modeling can be used both for linear and nonlinear systems. As an example, a straightforward order reduction in the linear case is of course the use of *normal modes* in

structural engineering [2]: they permits an accurate representation of the movement and the load acting on the structure using a relatively small set of DOFs. An order reduction for a nonlinear system is even more attractive, with the possibility of capturing physical phenomena characteristic of such systems, e.g. a possible Limit Cycle Oscillation (LCO) behaviour [3], mantaining a low number of DOF.

A ROM should provide a sufficiently accurate description of the system's dynamic with a computational cost much lower that that required by a direct numerical simulation, providing the dynamic response can be straightforwardly understood. Such model order reduction can be interpreted as a projection of the full order system, characterized by a high number of DOFs, onto a much smaller space, which encapsulate most, if not all, of the system's fundamental dynamics [1].

Even if it inherits a lot of procedures from system identification theory, it should not be mistaken that reduced order modeling is well different from system identification. In fact, while system identification techniques are adopted when an unknown system has to be modelled, starting from an unknown system structure or with a fixed structure but with unknown parameters (in this case is more correct to talk about parameter estimation), reduced order models start from a system which has been already modelled mathematically, tipically through a system of PDE, and the successive spatial discretization has produced a very large system of Ordinary Differential Equations (ODE). Typically such spatial discretization is performed considering the geometry of the problem, without taking into account the efficiency of the basis on which the solution is approximated. ROM methods can be always classified as projection methods, because of the dynamical system is projected onto a subspace of the original phase space, but in this case the basis on which the solution is discretized is chosen to be more efficient, since such a subspace is composed of basis functions which inherit the own characteristics of the overall solution. The basic idea of a ROM representation of a system is given in figure 1.1.

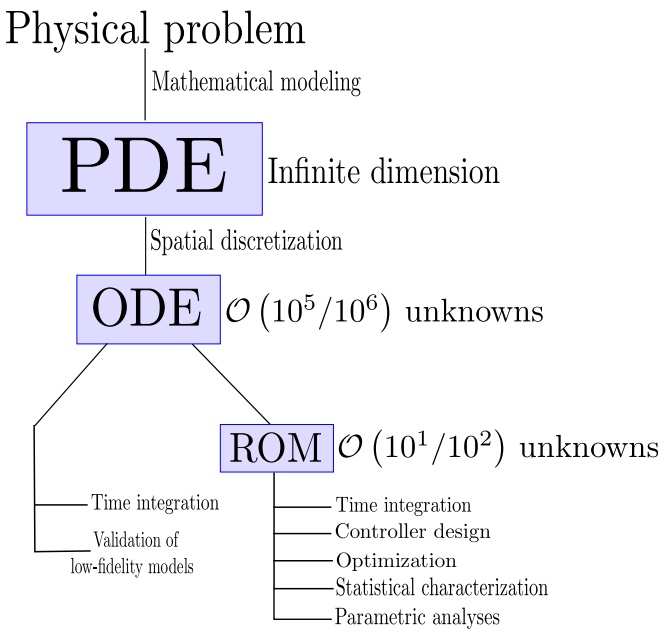


Figure 1.1: Flow chart of a reduced order analysis

1.1.1 Aeronautical problems

There are a lot of examples of reduced order models in the aeronautical world, especially for linear models. So the use of normal modes has just been cited as an effective technique to reduce the size of a structural system. Regarding the aerodynamic representation, panel methods [4]

has been extensively used for load estimation of steady/unsteady, incompressible and linearized compressible flows. Panel methods are of course reduced order models, since they capture the physical behaviour of a continuous system with a small number of unknowns.

Aeroelasticity is the well known multi-disciplinary field that studies the structural behaviour under the interaction of inertial, structural and aerodynamic forces. Examples of aeroelastic reduced order models are those based on simple physical principles, such the representation of a wing as a cantilevered elastic beam, represented by the first bending and torsional normal modes coupled with a nonlinear aerodynamic model based on a look-up table constructed by means of experimental data, theoretical formulas, panel methods or a few CFD simulations [3]. These models permit fast simulations of the system, useful during a control law design for example, with the accuracy of the results that depends on the structural and aerodynamic modeling.

Nonlinearities can be present in an aeroelastic system due to both structural, control and aerodynamic modeling. The resulting aeroservoelastic behaviour cannot be accurately predicted by standard linear analysis methods. Structural nonlinearities can arise for example from worn hinges of control surfaces, freeplay, hysteresis, material behaviour, or also from geometric characteristics, like high aspect ratio wings. Aerodynamic nonlinearities occur prevalently in the transonic flight regime or at large angle of attack, if the viscosity of the fluid is considered. The sources of those nonlinearities are shock waves with large motion and massively separated flows.

Modern airplanes that present their cruise at transonic flow regimes requires special attention: potential based methods are not adequate when strong shocks are present in the flow field. Therefore in such situations the adoption of the Euler's model is required. In case of very strong shocks even this latter flow model should not be employed, since it neglects the effect of thermal conductivity which can be important. In the viscous case the situation is even more complex, with the possibility of large flow separation. The flow behaviour can be accurately predicted by high fidelity CFD codes. However, their direct coupling with a structural solver for the determination of the aeroelastic response can be quite expensive from the computational point of view, due to the application of a multistep time-domain scheme used in the integration of the flow dynamics equations, which requires a lot of time, also with the modern high-performance computers. Moreover, many design and control applications in aeroelasticity may require relatively simple models to predict the system dynamics with as little computational (and experimental) effort as possible, especially in the conceptual and preliminary design phases of an aircraft. Reducing the complexity of the physical model is certainly a non-trivial task, in particular for the case of strongly nonlinear or even chaotic systems.

The solution proposed here, aimed to reduce the design computational cost, is the realization of a ROM that can predict the intrinsic unsteady, nonlinear behaviour of a flow past an aeronautical structure in terms of loads, permitting then, with the coupling of the aerodynamic model to the structural dynamics, a fast simulation of the resulting nonlinear aeroelastic system.

1.2 Different approaches to nonlinear reduced order modeling

Different methods for modeling nonlinear systems with a few DOF have been developed during the last years. In the field of aeroelasticity in particular, a great effort has been made in order to reduce the computational cost relative to high-fidelity, nonlinear aeroelastic simulations. The ROM proposed in the literature are able to capture the nonlinear flow characteristics with a greater computational efficiency than a full CFD simulation. Those dynamic system has been used to perform transonic flutter analysis, flutter analysis at high angle of attack, gust response, statistical characterization of the aeroelastic system and aeroservoelastic control law design [1]. However, most of the ROM proposed for predicting the aerodynamic behaviour are *dinamically linear* models [3]: this means that the ROM behaves like a linear system near the reference solution, with aerodynamic loads varying linearly with respect to the structural motion. Those models can be useful for flutter investigations, under the assumption of small structural vibrations,

but they are all but general aeroelastic model, since their response is no more reliable when the amplitude of the structural motion is large.

Aerodynamic reduced order modeling can be approached from three distinct ways. The first approach characterizes the aerodynamic flow field in terms of a relatively small number of global modes, a mode being intended here as a mean distribution of the variables that characterizes some gross motion of the flow [3]. This approach is the one followed by the Proper Orthogonal Decomposition (POD) modeling, which considers the aerodynamic model as a generic dynamic system, characterized by modes that describe the physical behaviour of the systems itself, like in the linear case. Considering these informations, a small set of dominant eigenmodes can be retained, constructing a model of reduced dimension. In this way, a typical CFD model, which can have from 10^4 to 10^6 or more DOF, may be reduced to a model containing only a hundred, or even tens of modes, remaining capable to accurately describe the pressure on an oscillating aerodynamic surface. The second approach follows the one proposed by the system identification theory: it does not explicitly rely on a modal description of the system, but appeals to the idea that only a small set of input, i.e. structural modes, and a correspondingly small number of output, i.e. modal loads, are dominant in the system dynamic evolution. This approach is the one followed by the Volterra series method and the dynamically driven recurrent neural network method proposed here. The third approach instead does not consider a proper model order reduction: a solution's structure is imposed a priori, for example, considering only periodic solutions, a Fourier series expansion becomes the most natural choice. Then, substituting the imposed shape in the full order problem, a nonlinear, algebraic system for the Fourier's coefficients is obtained and then solved with standard nonlinear solver. This is the Harmonic Balance (HB) method, which has proved itself very efficient in the computation of periodic responses of nonlinear systems with respect to the brute force full order simulation. Also a fourth approach can be considered, but it is less employed in nonlinear reduced order modeling with respect to the other methods. Such a method is based on a accurate interpolation of the full order system over various sampling points in the parameter space. The basis of this method are briefly exposed in this chapter.

A comparison of the different ROM approaches proposed in the literature will be given in the following paragraphs, along with the modeling technique that will be adopted in this work. It should finally be mentioned that all of the techniques here presented can be efficiently used also to build up analytic models from experimental data.

1.2.1 Volterra theory

The idea of a time-domain, linear, unsteady aerodynamic model has been estensively used in the literature regarding aircraft and helicopter aeroelasticity, by means of input-output models such as the *indicial response* [5], or a state-space representation [6, 7, 8, 9]. These models have been proved to be sufficiently accurate and robust to predict the aerodynamic loads deriving from small changes of the related boundary conditions.

Modeling the aerodynamic nonlinear response, such the one coming from a transonic flow has been found a challenging problem. Solutions as the *transonic indicial response* [10] have been proposed but more general approaches have been developed also. One of these is the *Volterra theory* of nonlinear systems. In practice this approach is a nonlinear Green's function method that provides a natural and intuitive extension of well understood linear input-output formulation into the nonlinear domain. In particular, Green functions can be interpreted as a linear subset of a much broader nonlinear Volterra functional space [1].

It is a nonlinear generalization of the concept of impulse response of a physical system: defining $u(t)$ a generic, scalar input function of the time t, for $t \geq 0$, the response of the system $\mathbf{w}(\mathbf{t})$ is given by the following linear combination of nonlinear terms:

$$\mathbf{w} = \mathbf{w_0} + \int_0^t \mathbf{h}_1(t-\tau)u(\tau)d\tau + \int_0^t \int_0^t \mathbf{h}_2(t-\tau_1, t-\tau_2)u(\tau_1)u(\tau_2)d\tau_1 d\tau_2 + \dots \tag{1.1}$$

where $\mathbf{w}_0$ is the initial state of the system, meanwhile the functions $\mathbf{h}_i$ are defined as *kernels* of the Volterra series. Their identification is based on measuring the response of the system to a set Dirac delta input at the time τ_i, $i = 1, n$, where n is the order of the series. For aeronautical applications, this identification is carried out from a small number of CFD simulations, where the input can be the imposed movement of the body and the output are the aerodynamic loads. These input should be carefully chosen, because of the general nonlinear behaviour of the system. Of course, for a linear system, only $\mathbf{h}_1$ has to be computed, and the other terms vanish away. Some analytical expressions have been derived for *1st* and *2nd* order kernels, as shown in [11]. Also a discrete time version of this method can be designed for numerical applications [1].

This formulation permits an analytical approach to the problem, but its accuracy has shown to be very sensitive to the choice of the input amplitude and to the computational time step used for discretizing (1.1), even in the linear case [12]. Higher order kernels do not seem to improve the performances of the nonlinear identification.

Sometimes, in order to reduce the computational cost of the CFD aerodynamic simulation, linearized model are considered. Such a linearization entails the evaluation of reference equilibrium solutions using a nonlinear model, but then the dynamic response varies linearly to the application of a generic input, which should be of small amplitude in order to provide acceptable results. In the literature these models are also called dinamically linear models [3]. These linearized models can be then transformed in state-space form. The identification of such models can be carried out through CFD analyses, computing the response of the aerodynamic system to an imposed input. Various strategies have been followed in the literature, for example a structural mode at a time is excited with an input chosen between a step, a ramp, a sinusoid, a gaussian pulse or a blended step [13]. These time domain responses are then transformed into the frequency domain obtaining the well known Generalized Aerodynamic Force (GAF) coefficient matrix. A good feature of the Volterra series approach is the ability to characterize the aerodynamic linearized system using a small number of CFD analyses. Once characterized, the model (1.1) can be efficiently used to obtain a prediction of the aerodynamic response without costly repeated uses of a CFD solver. Resulting linear ROMs can be computed efficiently in a state-space form with the Eigenvalue Realization Algorithm (ERA) [1].

Volterra series show some similarities with neural network modeling, and this fact will be highlighted in the following sections.

1.2.2 Proper Orthogonal Decomposition (POD)

The proper orthogonal decomposition technique, also known as Principal Component Analysis (PCA) or Karhunen-Loéve (KL) expansion can be interpreted as a generalized nonlinear version of the modal representation of a linear system. It is also referred as an empirical spectral method [3, 1]. It is used for determining efficient bases for approximating dynamic systems with a few DOF. This technique has been introduced for the study of the turbulent coherent structures [14]. As in the linear case, the system state $\mathbf{x}$ can be expressed as a modal expansion of the type:

$$\mathbf{x} = \mathbf{\Phi q} \tag{1.2}$$

where $\mathbf{\Phi} = [\phi_1\, \phi_2\, \cdots\, \phi_N]$ is the modal matrix and $\mathbf{q}$ is the related amplitude vector. The difference is that in this case the dynamic system is nonlinear, governed for example by the following autonomous dynamic relation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{R}(\mathbf{x}) \tag{1.3}$$

with the function $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^N$.

The basis vector is computed to maximize the following cost function:

$$J = \max_{\varphi} \frac{< (\mathbf{x}, \varphi)^2 >}{(\varphi, \varphi)} = \frac{< (\mathbf{x}, \phi)^2 >}{(\phi, \phi)} \tag{1.4}$$

where the operator $(\cdot,\cdot)$ denotes the inner product in the related space and $< \cdot >$ is a time average operator.

In practice, the POD basis is derived from a set of observations the system responses. Samples, or snapshots of the system behaviour are used to compute an appropriate set of basis functions to represent the system variables. It is important to remark that the basis function so obtained are not only appropriate, but optimal [15, 1]. The need of obtaining samples from the full order system can be viewed as both a strength and a weakness of the method: its strength derives from the fact that the sampling of various system responses the ROM can be efficiently tuned so to obtain an high-fidelity behaviour with a small number of DOFs. Its weakness is of course related to the need of performing various runs of the full order model in order to construct the set of snapshots, and a possible lack of model robustness to changes in parameters governing the system behaviour.

It has been shown in [14, 3], that taking a set of responses of the system at different time steps or from different input, let us say $M << N$ samples, it is possible to construct the two-point correlation matrix $\mathbf{Z}$:

$$\mathbf{Z}_{i,j} = (\mathbf{x}_i, \mathbf{x}_j); \tag{1.5}$$

being $\mathbf{x}_i$ and $\mathbf{x}_j$ the evaluation of the system state at t_i and t_j if the simulation is carried out in the time domain. The eigenvalue problem associated to this matrix:

$$\mathbf{Z}\phi = \lambda\phi \tag{1.6}$$

produces the optimal set of basis vectors, $\mathbf{\Phi} = [\phi_1|\phi_2|...|\phi_M]$ representing the nonlinear system. The eigenvalue λ_i represents some form of energy contribution of the mode ϕ_i to the system response. In practice, fewer than M modes are retained to simulate the system behaviour, and these are selected considering the amplitude of the module of the eigenvalue λ_i. However, it has been observed that the range of validity of the ROM decreased as the number of retained mode is decreased [1].

This approach can also be applied in the frequency domain, where the response needed for constructing the two-point correlation matrix is retained only for a finite set of frequencies.

It is worth to be remarked that the POD approach to reduced order modeling is strictly related to the Singular Value Decomposition (SVD) of rectangular matrices [16]. Remembering briefly what SVD means, consider a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ with rank d. It can be decomposed by means of the following relation:

$$\mathbf{U}^H\mathbf{X}\mathbf{V} = \begin{bmatrix} \mathbf{\Sigma}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{\Sigma} \in \mathbb{R}^{n\times m} \tag{1.7}$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$ are two orthogonal matrices, composed by the left and right singular vectors of $\mathbf{X}$ and $\mathbf{\Sigma}_d = \mathrm{diag}_d\left(\sigma_1, \sigma_2, \cdots, \sigma_d\right) \in \mathbb{R}^{d\times d}$, with σ_i the i-th singular value of $\mathbf{X}$. The superscript H is the transpose and conjugate operator, which reduces to the classical transposed operator T for real-valued problems. Left and right singular vectors can be computed by:

$$\mathbf{X}^H\mathbf{u}_i = \sigma_i^2\mathbf{u}_i \quad \text{and} \quad \mathbf{X}\mathbf{v}_i = \sigma_i^2\mathbf{v}_i \qquad i = 1, ..., d \tag{1.8}$$

Such a link lies in the fact that the approximating POD basis should contain as much informations or energy as possible. With this consideration, the approximation of the snapshot vector $\mathbf{x}_i$ through the single vector ϕ can be interpreted as the following constrained optimization problem:

$$\max_{\phi} F = \sum_{j=1}^{M} \left(\mathbf{x}_i, \phi\right)^2 \qquad \text{such that:} \quad (\phi, \phi) = 1 \tag{1.9}$$

Using the Lagrange multipliers method, it can be derived that the solution of 1.9 is given by the following eigenvalue problem:

$$\mathbf{X}\mathbf{X}^H\phi = \sigma^2\phi \tag{1.10}$$

which is indeed equal to 1.6. The singular value analysis yields that the vector ϕ_1 is a singular eigenvector of 1.10 and the relative singular value is equal to the functional value defined in 1.9. Iterating such precedure, the vectors ϕ_i, $i = 1, ..., d$ can be computed:

$$\max_{\phi} F = \sum_{j=1}^{M} (\mathbf{x}_i, \phi)^2 \qquad \text{such that:} \quad (\phi, \phi) = 1 \quad \text{and} \quad (\phi, \phi_j) = 0, \quad j = 1, ..., i-1 \tag{1.11}$$

with the corresponding functional value equal to σ_i^2. It is now clear that for every $d \leq M$ the approximation of the columns of $\mathbf{X}$ by the first d singular vectors $\{\phi_i\}_{i=1}^{d}$ is optimal in the least squares sense among all rank d approximations to the column of $\mathbf{X}$. Moreover, this approach leads to a practical determination of a POD basis of rank d [16].

At this point the representation (1.2) can be inserted in the original nonlinear system (1.3), that can be solved by a projection technique:

$$\mathbf{F} = \frac{d\mathbf{x}}{dt} - \mathbf{R}(\mathbf{x}) \tag{1.12}$$

is the dynamic residual, which can be forced to vanish after weighting it with each of the M modes:

$$\mathbf{\Phi}^T \mathbf{F} = \mathbf{\Phi}^T \left(\frac{d\left(\mathbf{\Phi q}\right)}{dt} - \mathbf{R}(\mathbf{\Phi q}) \right) = \mathbf{0} \tag{1.13}$$

assuming that the modes are normalized such that $\mathbf{\Phi}^T \mathbf{\Phi} = \mathbf{I}$, eq. (1.13) results in a subspace projection of the full order system:

$$\frac{d\mathbf{q}}{dt} = \mathbf{\Phi}^T \mathbf{R}(\mathbf{\Phi q}) \tag{1.14}$$

If one is interested to a steady-state solution of problem (1.14) the equilibrium solution satisfies the following system of nonlinear, algebraic equations:

$$\hat{\mathbf{R}} = \mathbf{\Phi}^T \mathbf{R}(\mathbf{\Phi q}) = \mathbf{0} \tag{1.15}$$

An extension of the subspace projection method is the so called direct projection, which utilizes higher order terms of the function $\mathbf{R}\left(\mathbf{x}\right)$, for example considering its Taylor series expansion:

$$\begin{aligned} \mathbf{R}\left(\mathbf{x}_0 + \Delta\mathbf{x}\right) = \mathbf{R}\left(\mathbf{x}_0\right) &+ \sum_{i=1}^{N} \left.\frac{\partial \mathbf{R}}{\partial x_i}\right|_{\mathbf{x}_0} \Delta x_i \\ &+ \frac{1}{2} \sum_{i,j=1}^{N} \left.\frac{\partial^2 \mathbf{R}}{\partial x_i \partial x_j}\right|_{\mathbf{x}_0} \Delta x_i \Delta x_j \\ &+ \frac{1}{6} \sum_{i,j,k=1}^{N} \left.\frac{\partial^3 \mathbf{R}}{\partial x_i \partial x_j \partial x_k}\right|_{\mathbf{x}_0} \Delta x_i \Delta x_j \Delta x_k \\ &+ \mathcal{O}\left(|\Delta\mathbf{x}|^4\right) \end{aligned} \tag{1.16}$$

The following pre-multiplication by $\mathbf{\Phi}^T$ and post-multiplication by $\mathbf{\Phi}$ will produce a set of nonlinear ODEs, with the linear portion in state-space form. However the numerical computation of the Jacobians can be burdensome if not adequate derivative computational technique is not adopted.

This technique has also proved to be very efficient in model sensitivity analysis. Assuming that the general model (1.3) depends on a some set of parameters, here called $\theta \in \mathbb{R}^{n_\theta}$, this can be explicitly expressed as:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{R}\left(\mathbf{x}; \theta\right) \tag{1.17}$$

Having defined $\hat{\mathbf{R}}$ in (1.15), the Jacobian matrices with respect of $\mathbf{q}$ and θ can be computed:

$$\mathbf{J} = \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{q}} \qquad \mathbf{Y} = \frac{\partial \hat{\mathbf{R}}}{\partial \theta} \tag{1.18}$$

Then, if (1.15) holds, the following relation is also true:

$$\mathrm{d}\hat{\mathbf{R}} = \mathbf{J}\,\mathrm{d}\mathbf{q} + \mathbf{Y}\,\mathrm{d}\theta = \mathbf{0} \tag{1.19}$$

and consequently:

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\theta} = -\mathbf{J}^{-1}\mathbf{Y} \tag{1.20}$$

remembering the transformation (1.2):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\theta} = -\mathbf{\Phi}\mathbf{J}^{-1}\mathbf{\Phi}^{T}\frac{\partial \mathbf{R}}{\partial \theta} \tag{1.21}$$

This is the sensitivity dynamic equation, that can be formulated is one wishes to analyze the changing behaviour of the system with the modification of the system's parameters. Of course, as usual, the inverse of the Jacobian matrix $\mathbf{J}$ is not computed directly: first it is factorized and then repeatedly used in the evaluation of (1.21). Moreover, the computation of the Jacobian matrices can be quite a burden for the method proposed, but since the model has reduced dimensions, or if more sophisticated techniques are used [17], these terms can be efficiently obtained.

POD ROMs have been effectively used also for linear modeling: the reference equilibrium solution is computed with a CFD solver with accelerating techniques, which are briefly described in chapter 3 toward the steady state value; then the governing equations are linearized for periodic disturbances of small amplitude, placed in the frequency domain form, and solved with an another CFD run. This solution is thus gathered for a range of different frequencies to form the set of POD's snapshots, computing then the resulting linear ROM model [1]. A further model order reduction can be obtained through the well known *balanced* reduction, described briefly in [3] and detailed in [15].

POD representation of nonlinear dynamic system has been shown to be very effective and maybe today is the most robust method for reduced order modeling in computational aerodynamic and aeroelasticity [1]. The basis used by this formulation is optimal, and thus permits to develope a model which, even with a few DOF captures the system dynamics accurately over a range of frequencies and forcing input [15]. It should be noted that for linear systems, of the type: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, the POD method presents the same features of a subspace iteration scheme [2, 18] or the Arnoldi method [18] in the computation of the eigenvalues of a matrix: it projects linearly the system states on a small basis characterized however by a high precision, because of the subspace is composed by basis functions which inherit already special characteristics of the overall solution. Of course also this method presents the need of an accurate design of the input in order to represent the general nonlinear behaviour of the system. If no physical interpretation is considered during the input design, the results can be inaccurate. Similarly to the Volterra series, in this case the assembling of the correlation matrix needs the collection of the samples derived from the numerical simulation of the full-order nonlinear problem, and this is of course the most expensive part of the method. Moreover, the collection of the samples, or *snapshots* is not straightforward, no standard rules are defined for doing it, having as a consequence a problem-dependent formulation, making the use of this method difficult for non-expert designers.

1.2.3 Harmonic Balance Method (HB)

Assuming that the solution of (1.3) is periodic in time, it can be expressed in form of the Fourier series:

$$\mathbf{x} = \sum_{m=-\infty}^{\infty} \hat{\mathbf{x}}_m \exp\left(j\omega_0 m\, t\right) \tag{1.22}$$

where ω_0 is the fundamental frequency, defined as $\omega_0 = \frac{2\pi}{T}$, with T the period of the harmonic response, and j is the imaginary unit, such that $j = \sqrt{-1}$. Once known that (1.3) exhibit periodic solutions, it is sufficient to computed such solution only in a time interval which corresponds to its period.

The HB technique, also known as time-Galerkin method, consists in assuming the solution of (1.3) in the form of a truncated Fourier series, with a predetermined number of harmonics, here referred with N_H. Then, this expression is substituted in the original ODE, and algebraically manipulating the results to collect terms belonging to the same frequency. Any resulting term with a frequency not in the Fourier expansion is dropped. Each harmonic is then balanced by requiring that equal frequency terms on each side of the equation satisfy the equality independently. This balancing results in a system of coupled, generally nonlinear, algebraic equations which is then solved for the Fourier coefficients $\hat{\mathbf{x}}$ [19].

If one wants to remain the time domain for computing the solution of (1.3), the HB method permits to recast the problem in a steady-state form that accounts for the underlying time periodicity of the solution and can be solved with a pseudo-time integration using standard acceleration techniques [20].

In practice, the truncated Fourier series results in:

$$\mathbf{x} = \sum_{m=-N_H}^{N_H} \hat{\mathbf{x}}_m \exp\left(j\omega_0 m\, t\right) \tag{1.23}$$

Substituting the definition (1.23) into (1.3) and integrating over a period, a system for the Fourier coefficients is obtained:

$$\mathbf{A}\mathbf{X} - \mathbf{R}(\mathbf{X}) = \mathbf{0} \tag{1.24}$$

where

$$\mathbf{A} = \mathrm{diag}(-j\, N_H, ..., j\, N_H) \qquad \mathbf{X} = \begin{pmatrix} \hat{\mathbf{x}}_{-N_H} \\ \vdots \\ \hat{\mathbf{x}}_{N_H} \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R}\left(\hat{\mathbf{x}}_{-N_H}\right) \\ \vdots \\ \mathbf{R}\left(\hat{\mathbf{x}}_{N_H}\right) \end{pmatrix} \tag{1.25}$$

Problem (1.24) can be solved with a nonlinear algorithm such as Netwon-Raphson technique. Typically, a reduced set of Fourier modes is required in order to capture the nonlinear behaviour of the high-fidelity system. It has been found out that if the body motion is small, a single harmonic is often sufficient in order to obtain accurate results [3]. During the analysis of a LCO, the period T is unknown: the problem can be resolved retaining the system (1.24), just fixing a component of a single Fourier coefficient, for example $\hat{x}_{1_{N_H}}$ and then solving for the period along with the remaining coefficients [19, 21]. Such a procedure will be explained extensively in chapter 2, where a periodic collocation method in the time domain is presented. However, it can be remarked by now that such a method is analogous to an approach based on describing functions, and this implies a sort of filtering of the high frequency of the system [22], which is however often provided by other parts of the system itself, although it can be difficult to be checked analytically [23].

The evaluation of the so called *harmonic fluxes* is computationally expensive, in the order of $\mathcal{O}\left(N\, N_H^3\right)$ for Euler system in the fluid dynamics case, and are not easy to extend to turbulent, viscous flows [19].

Reference [19] proposes also a simpler and more efficient formulation: first the solution $\mathbf{x}(t)$ of (1.3) is *collocated* at $2\, N_H + 1$ instant of times, evenly distributed about the periodic orbit, and this terms are collected in the following array:

$$\mathbf{X}^* = \left(\mathbf{x}^T(0), \mathbf{x}^T\left(\frac{T}{2\, N_H + 1}\right), ..., \mathbf{x}^T\left(\frac{2\, N_H\, T}{2\, N_H + 1}\right)\right)^T \tag{1.26}$$

Then a Fouries transform operator $\mathbf{E} \in \mathbb{R}^{N\, N_H \times N\, N_H}$ relates $\mathbf{X}$, defined in (1.25) to $\mathbf{X}^*$:

$$\mathbf{X} = \mathbf{E}\mathbf{X}^* \tag{1.27}$$

the operator E has a blocked structure. Problem (1.3) is then recasted in the frequency domain:

$$j\,\omega\,\mathbf{x} = \mathbf{R}\,(\mathbf{x}) \tag{1.28}$$

Defining the following elements:

$$\mathbf{N} = \mathrm{diag}\,(-N_H, \ldots, N_H) \qquad \mathbf{R}^* = \begin{pmatrix} \mathbf{R}\,(\mathbf{x}(0)) \\ \mathbf{R}\left(\mathbf{x}\left(\dfrac{T}{2\,N_H+1}\right)\right) \\ \vdots \\ \mathbf{R}\left(\mathbf{x}\left(\dfrac{2\,N_H\,T}{2\,N_H+1}\right)\right) \end{pmatrix} \tag{1.29}$$

The array $\mathbf{R}^*$ can be related with the array $\mathbf{R}$ of eq. (1.24) with same previous blocked operator $\mathbf{E}$:

$$\mathbf{R} = \mathbf{E}\mathbf{R}^* \tag{1.30}$$

Substituting these definitions in (1.28), the following nonlinear, algebraic problem is obtained:

$$j\,\omega\,\mathbf{N}\mathbf{E}\mathbf{X}^* = \mathbf{E}\mathbf{R}^* \quad \text{equivalently} \quad j\,\omega\,\mathbf{E}^{-1}\mathbf{N}\mathbf{E}\mathbf{X}^* = \mathbf{R}^* \tag{1.31}$$

Where $\omega = \frac{2\pi}{T}$. In [19] a pseudo time τ has been introduced, by which (1.31)is integrated forward in the pseudo time toward the fully developed solution:

$$\frac{\partial \mathbf{X}^*}{\partial \tau} + j\,\omega\,\mathbf{E}^{-1}\mathbf{N}\mathbf{E}\mathbf{X}^* = \mathbf{R}^*\,(\mathbf{X}^*) \tag{1.32}$$

This approach allows the time dependent solution of (1.32) to be computed with the existing acceleration algorithms toward the steady-state [20], that need far fewer iterations than a full order time-accurate integration method. In this way, the evaluation of the fluxes has a computational cost in the order of $\mathcal{O}\,(N\,N_H)$ [19], bounding the cost of the numerical scheme. However it should be noted that such an approach has been developed for a known period T, e.g in the analysis of a flow past the front stage rotor of a high-pressure compressor [24]. A different fomulation of the problem, based on the periodic collocation in the time domain of the solution, permits to treat explicitly the period as an unknown, and it is presented in chapter 2.

HB is not a reduced order modeling technique in a strict sense, since it solves the full-order nonlinear problem, but the computational time is reduced from hours to seconds for time-periodic problems. It does not reduce the number of variable resulting from the spatial discretization and does not provide a model that is a compact representation of the full order system. Thus HB does not involve a compression of the spatial data, preserving the parametric relationships present in the system without loss of fidelity. However it should be noted that with respect to the other methods presented before, the HB method requires a re-formulation of the governing equations of the system, and this can be interpreted as an its weakness. Nonetheless, this technique does indeed yield an efficient and low order representation of the temporal variations of complex systems experiencing periodic behaviour in time, presenting also a great accuracy [1].

HB has been used for many years for studying the behaviour of time-periodic systems of ODE, but in the last years it has been applied to the aeronautical environments, especially for analyzing nonlinear, internal flow problems, in turbomachines. High fidelity numerical simulations of a fluid flow through a transonic turbomachine are of primarily interest to designers of aeronautical jet engines. Such results can be obtained with the modern CFD algorithms today available, but the time required for such simulations is the main limit of this approach. Moreover, a time-accurate solution of a fully developed flow periodic in time, i.e. the flow through a rotor-stator, is often inefficient, since it is necessary to step through many disturbance periods before a fully developed solution is reached [1]. HB has permitted to reduced the computational time taking advantage

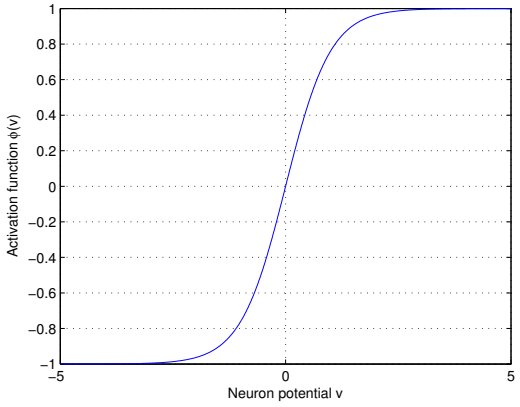
of the time-periodic nature of the flow, assuming the more natural solution that one can take in the frequency domain.

Thus, HB method is an efficient tool for reduced order modeling, but it is limited to the study of time-periodic problems. So, its application to aeroelasticity is confined at the moment to the study of flutter and LCO; it has not been applied for the computation of a generic aeroelastic response yet.

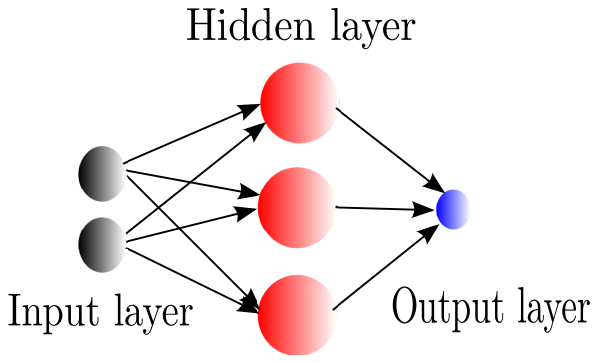
1.2.4 Recurrent Neural Networks (RNN)

A neural network is a massively parallel distributed process made up of simple processing units (the neurons) that have a natural capability to store the knowledge accumulated through experience, making it available for later uses. Knowledge is acquired through a *learning process*, and it is stored in the synaptic connections linking the neurons. Neural network modeling has been widely used in the nonlinear branch of the system identification because of their ability of self-learning, fault tolerance, intrinsic nonlinearity, adaptivity and microbiological analogy [25].

Neural networks are a powerful tool for approximating nonlinear dynamic systems, also when the system itself is unknown and the only data available is the input-output relation, thus they permit a sort of *black-box* modeling of any nonlinear system. In neural network modeling, the model structure is fixed by the structure and the connections present in the network itself, meanwhile the parameters are determined through experimental/computational models. Thus, for building black-box models no or very little prior knowledge is exploited. The model parameters have not direct relationship to first principles [26]. The advantages of black-box modeling are the short modeling time, a little expertise required by the analyst and can be used for both understood and not understood processes. The second feature mentioned is also one of the drawbacks of such approach, since a little understanding of the physical process is given by this modeling technique. However, in chapter 3 a way will be shown to obtain a physical meaning from the output of a neural network.



(a) Squashing function



(b) Simple neural network with one hidden layer

A general network is composed by an input layer, where the input data is stored and passed to the computational units, represented by the neurons, which receive a linear combination of the input, where the coefficients of the combination are called *synaptic weights* of the network, and use this input as the argument of a nonlinear function, usually a squashing function, as in the example figure 1.2a. If the network is used to approximate a nonlinear function, then this computational layer should be a *hidden* layer, meaning that the output of the neurons should not be the direct output of the network. Thus the output of the network is usually a combination of the output of each neuron, typically linear. An example of such a network is given in figure 1.2b. Because of the *universal approximation theorem*, a neural network with one hidden layer is sufficient for approximating a nonlinear function with arbitrary accuracy, ensuring that the

number of neurons is adequate [25].

The modeling of nonlinear dynamic systems is more interesting. In such a case the network should be able to learn the evolution path of a general system from the input-output data pairs only. The *memory* is inserted in the process through a time delay applied to the hidden neurons, defining thus the *state* of the network. Basically, a network state is an hidden neuron which is connected to an other hidden neuron. Such a network is defined as *recurrent* neural network. The input layer is now composed by the set of the input data, function of the time now, and the set of the delayed output of the hidden neurons.

A fully connected recurrent neural network feedbacks all the hidden neurons output to the input layer, meanwhile a partially connected recurrent neural network feedbacks only a small fraction of such output, as shown in figure 1.2, represented with the so called *Elman* neural model [25].

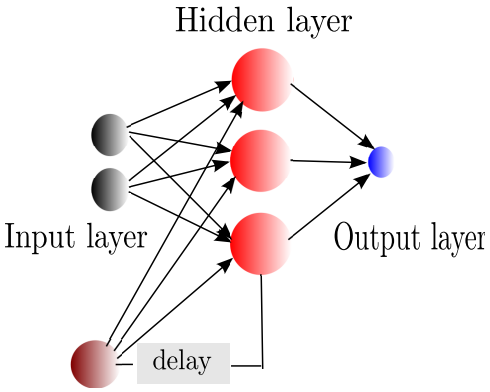


Figure 1.2: Partially recurrent neural network

The resulting internal dynamics realizes a nonlinear state space model without information about the true system states. The internal states can be therefore interpreted as an artificial tool employed for the realization of the desired dynamic input/output behaviour. Such models with internal dynamics are most frequently based on a Multi Layer Perceptor (MLP) network architecture [26, 25]. Such structured recurrent scheme leads to faster training and fewer stability problems. Nevertheless, the number of states is related to the number of neurons in the hidden layer, and this fact restricts the flexibility of such tool. However, in chapter 3, a generalization of such network formulation will be proposed, trying to remove the above mentioned limitation. Such recurrent networks can be formalized by the following discrete state-space model:

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{\Phi}(\mathbf{W^a}\mathbf{x}_n + \mathbf{W^b}\mathbf{u}_n) \\ \mathbf{y}_n = \mathbf{W^c}\mathbf{x}_n \end{cases} \tag{1.33}$$

or by a continuous model:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{\Phi}(\mathbf{W^a}\mathbf{x}(t) + \mathbf{W^b}\mathbf{u}(t)) \\ \mathbf{y}(t) = \mathbf{W^c}\mathbf{x}(t) \end{cases} \tag{1.34}$$

where $\mathbf{x}$ represent the state of the network, $\mathbf{u}$ the input, $\mathbf{y}$ the output, $\mathbf{W^a}$, $\mathbf{W^b}$ the synaptic weights between the hidden and the input layer, $\mathbf{W^c}$ the synaptic weights between the hidden and the output layer and $\mathbf{\Phi}$ the set of nonlinear activation function, defined as:

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(v) \\ \vdots \\ \phi_{n_x}(v) \end{pmatrix} \tag{1.35}$$

if n_x is the dimension of the state space. Several parametrizations of recurrent neural networks, for both discrete and continuous formulation have been proposed in the literature [27, 28, 25, 29, 30] but no standard formulation is available nowadays for the representation of nonlinear dynamic systems. Different parametrizations will be considered in chapter 3, enhancing their differences in the training and generalization phases.

Like an human brain, in order to behave like the nonlinear dynamic system of interest, the neural network must gain its experience over training signals chosen a priori, which should be able to excite the main nonlinearities present in the system. The choice of the model input is tipically realized by a trial-and-error approach with the help of any available prior knowledge. In the fields of structural mechanics, aerodynamics, and other branches of physics, the influence of the singular variables on the system output is usually quite clear, thus the relevant model input can be chosen by available insight into the physics of the process [26]. The training of neural network can be performed in two main ways:

- *Unsupervised* training: only the system input are available. The principal tool employed for this kind of training is the Principal Component Analysis (PCA) [25], which permits to discard non-relevant input with a low computational demand;

- *Supervised* training: both the system input and output are available for the training of the network. The resulting training process can be interpreted as an optimization problem where the error, defined as the difference between the real system output and the network output, is minimized varying the synaptic weights of the network.

After the training phase, if the training signal has excited all the nonlinearity, frequencies and amplitudes of interest, the resulting neural network should be able to *generalize* the knowledge learned into the behaviour of the full order nonlinear system. Thus for generalization is meant the capacity of the neural network to reproduce the full system output for an input data sample not contained in the training set.

Different approaches to the system identification of nonlinear aeroelastic systems with discrete time recurrent neural network have been adopted: for example the utilization of a recurrent neural network with Radial Basis Function (RBF) as activation functions has been able to predict the limit cycle oscillation in a transonic flow of the *NACA 64A010A* airfoil and the Benchmark Active Control Technology (BACT) wing [31]. Such recurrent RBF network can be interpreted as a nonlinear version of the well known AutoRegressive model with eXogenous input (ARX) identification technique, in fact in different sources of the literature this kind of neural network is also defined as Nonlinear AutoRegressive model with eXogenous input (NARX). Instead, in [32] its has been presented an aerodynamic system identification technique based on the rigorous mathematical theory of the Support-Vector-Machine (SVM), always related to a discrete time recurrent neural network, which has been able to predict different limit cycle oscillation conditions for various Mach numbers.

However, the first approach discussed just above can be very costly from the computational point of view, because to train the synaptic weights of the network to behave like the nonlinear system of interest, the RBF parameters, such that the function center and the function spread (or radius), must be trained as well. Different algorithms are present in the literature for predicting these function parameters from the input data, the most known one being the K-means clustering algorithm [25]. Moreover, a NARX model provides only a one-step ahead prediction of the response, rather than optimizing the simulation error. However, once the neuron centers and spreads have been computed, the resulting training algorithm is reduced to a linear least squares problem, as shown in [31, 33], therefore this phase is characterized by a very small computational time. The second approach is well known for static neural network, but a small number of applications have been presented so far in the contest of dynamics systems. Another consideration is that both approaches are formulated in a discrete-time domain, with no applications in the continuous domain.

The difference between the discrete time model and the continuous one will be explained in this thesis, but a very important property of a recurrent neural network described in the state-space form as (1.33) or (1.34) is that it is a universal approximator of all nonlinear dynamic systems [25, 29, 34]. This feature makes recurrent neural networks a robust tool for reduced order modeling.

Of course they present also some drawbacks. In fact there is no mathematical theorem that guarantee the achievement of the global minima of the error during the training phase, thus there is the possibility of converging toward a local minima, which causes poor performances. Moreover, too long training data sets may lead to over-fitting a specific training with an ensuing poor generalization capability. Thus, in order to obtain a functional recurrent neural network, a good parametrization of the network itself and a careful design of the training signal is required.

Some aspects regarding neural network are very similar to Volterra series, discussed previously. Each of them involves the characterization of a system through an input-output mapping, and there is a direct relationship between the synaptic weights of the network and the kernels used in the Volterra representation [1]. The major difference between the two approaches is the training effort. As previously mentioned, the first kernels of a Volterra series are known analytically with no need of a training period. Moreover the definition of a kernel as a generalization of the impulse response of a system made the series approach adapts to physical interpretation. However recurrent neural networks have no the disadvantage related to the input amplitude limitations required for convergence of the Volterra series and moreover, the need of higher order terms, which can be computationally expensive is no more required.

Comparing recurrent neural networks to the others method presented (POD and HB), it is straightforward to note that the only design parameter required to neural network is the number of states used (n_x), meanwhile the choice of *snapshots* in POD design can be non-trivial and made the solution snapshot-dependent. If opportunely trained, a neural network can be a general approximator of a dynamic system, thus can be used to predict the response to a general input, instead, HB method is typically employed for analyzing the stability of the time-periodic system, with no regarding to dynamic response [3, 19].

1.2.5 High order interpolation of the high-fidelity model

In order to reduce the computational cost of simulations related to high-fidely models, a high order interpolation of the solution at different sampling points may be exploited, obtaining smooth models which can be employed in fast analysis. Different applications of such a method are available in the literature, both for large linear [35] and nonlinear [36, 37] systems. In particular, the latter references are related to aeroelastic problems, where the computation of the stability properties (in the specific, the *flutter boundary*) of the system is carried out coupling the structural system, described in modal form, with the aerodynamics, described by a CFD model. Such nonlinear dynamic system can be represented by:

$$\dot{\mathbf{x}} = \mathbf{R}\left(\mathbf{x}, \mu\right) \tag{1.36}$$

where $\mathbf{x}$ is the system state and μ represents a set of parameters, tipically the dynamic pressure, the Mach number, the altitude, etc. The stability of the systems is evaluated through the computation of the eigenvalues of the linearized system near a trim condition:

$$\mathbf{R}\left(\mathbf{x}_0, \hat{\mu}\right) = 0 \longrightarrow \mathbf{A} = \left.\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right|_{\mathbf{x}_0, \hat{\mu}} \tag{1.37}$$

and checking the well known stability rule (asymptotic stability is related to a negative real part of all the eigenvalue). Because of the presence of an aerodynamic model described through a CFD scheme, the resulting eigenvalue problem $\mathbf{Ap} = \lambda \mathbf{p}$ is very large, therefore very expensive. In [36], the Jacobian matrix is scomposed in structural and aerodynamic part, and only the

structural eigenvalue sub-problem is retained of interest, considering only the interaction with the aerodynamic system.

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_s \\ \mathbf{p}_a \end{pmatrix} \qquad \mathbf{A} = \begin{bmatrix} \mathbf{A}_{ss} & \mathbf{A}_{sa} \\ \mathbf{A}_{as} & \mathbf{A}_{aa} \end{bmatrix} \qquad \mathbf{p}_s \in \mathbb{R}^{2\,n_{\text{struct}}} \quad \mathbf{p}_a \in \mathbb{R}^{n_a} \tag{1.38}$$

The resulting eigenvalue problem can be written as:

$$\mathbf{S}(\lambda)\mathbf{p}_s = \lambda \mathbf{p}_s \tag{1.39}$$

Where $\mathbf{S}(\lambda)$ is the Schur complement of $\mathbf{A}$, and it is equal to:

$$\mathbf{S}(\lambda) = \mathbf{A}_{ss} - \mathbf{A}_{sa}\left(\mathbf{A}_{aa} - \lambda \mathbf{I}\right)^{-1}\mathbf{A}_{as} = \mathbf{A}_{ss} + \mathbf{S}^c(\lambda) \tag{1.40}$$

The size of the nonlinear eigenvalue problem written in 1.39 is equal to $2\,n_{\text{struct}} + 1$ (the structural variables plus the eigenvalue), instead of the $n_a + 2\,n_{\text{struct}} + 1$ unknowns of the original one. The computational saving is considerable, because of $n_a >> 2\,n_{\text{struct}}$. Since the eigenvalue problem 1.39 is nonlinear, it has to be solved with a iterative technique, i.e. Newton-Raphson. However, the evaluation of the coupling term $\mathbf{S}^c(\lambda)$ is very costly, since the aerodynamic is represented by a very large system. The solution proposed in [36] avoid such large cost, approximating the Schur complement defined in 1.40: the coupling term is evaluated on different sampling points selected in the parameter space described by μ, then its extension over all the domain is performed interpolating the interaction matrix over such few samples, where it is evaluated through the high-fidelity model, obtaining the following nonlinear eigenvalue problem:

$$\left(\mathbf{A}_{ss} + \hat{\mathbf{S}}^c(\lambda)\right)\mathbf{p}_s = \lambda \mathbf{p}_s \tag{1.41}$$

which is far away cheaper than the previous version, since the full interaction term has not to be evaluated at each iteration of the Newton-Raphson algorithm. In this particular case, the interaction term $\mathbf{S}^c$ is interpolated through a Kriging technique, which is a response surface method. Without considering the details, that can be found extensively in [36, 37], such a predictor is composed by two main terms: a low-order regression model and a random normally distributed signal, characterized by a covariance that depends on the variance of the input samples and the two-point correlation matrix constructed by the samples itself. It is important to note that the mean square error of the prediction vanishes at a sampled location, interpolating the exact system response [38]. When using such a response surface method we are representing how the process typically behaves and the interpolated data is the most consistent prediction with the estimated typical behaviour. A final issue that can be considered is how to choose the samples. In the literature there are different efficient algorithms which permit to place the samples where some cost function is maximized. We can cite, for example, the risk-based sampling, the latin hypercube and the expected improvement algorithm [36, 38].

Other types of interpolation strategies can be used, for example Radial Basis Functions (RBFs) are severally adopted in different applications, an example can be found in [35], because of their high smoothness and accuracy, permitting a reliable interpolation of the data computed by the high-fidelity model over all the parameter space of interest. Other candidates as high-dimensional interpolators are low-order polynomials and moving least squares approaches.

1.3 Thesis outline

In the following chapters, the work is organized as follows. In chapter 2 the typical nonlinear stability aeroelastic problems encountered in practical application are described, with particular emphasis to the Limit Cycle Oscillation (LCO) phenomenon. Such a problem will be approached from two ways, one that considers the LCO as an initial value problem, the other considering

instead the LCO solution of a boundary value problem associated to the nonlinear aeroelastic system. Chapter 3 introduces the aerodynamic modeling employed in this work, summarizing the analytical formulation of the Navier-Stokes and Euler flow models, and then considering the numerical formulation adopted to obtain physically meaningful solutions in transonic flows. Considerations about the size of numerical model suggests that a low order representation of the aerodynamic system should be required if high accurate load predictions are needed during the conceptual and preliminary design phases of an airplane. A continuous time recurrent neural network is proposed as reduced order model of such computational costly numerical system. Different parametrizations are presented with the associated analytical computation of the Jacobian matrix needed for the nonlinear training algorithm adopted. A series of possible training input are proposes, using physical considerations for their design. In chapter 4 the tool Automatic Differentiation (AD) is introduced, comparing its efficiency with the other methods employed in the computation of function derivatives in computer programs. A time-marching explicit integration scheme, based on Taylor series expansion is proposed, exploiting the functionalities of the AD algorithms in the computation of Taylor series vectors and matrices of vector-valued function. Some simple examples are presented, with the aim of setting up the various parameters involved in such a method. Finally an AD technique is employed in the network training algorithm through the integration scheme proposed, applied in the computation of the network response and its Jacobian matrix. In chapter 5 the reduced order model is tested in the identification of the unsteady aerodynamic loads acting on a pitching airfoil. The performances of the different parametrizations are compared in terms of computational time and training accuracy. In chapter 6 instead the aerodynamic reduced order model is applied in the prediction of various aeroelastic LCOs experienced by a pitching and plunging typical section and the BACT wing, both working in a transonic regime. Also in this chapter the different network parametrizations are compared in accuracy. The LCO behaviour will be computed though the two different approaches derived in 2. Since their efficiency, the computation of the LCO envelope has proved to be much more fast than the one computed by the CFD solver considered. Finally, in chapter 7 some conclusions about the reduced order model technique are considered, summarizing the strength and the weaknesses shown during the analyses by such a methodology, and pondering over some future developments that can be employed in further applications.

Chapter 2

Nonlinear aeroelasticity

The determination of the aeroelastic behaviour is a demanding problem owing to the capturing of system nonlinearities, which come from structural and aerodynamic modeling, playing a key role in the phenomenon called Limit Cycle Oscillation (LCO). To capture the effects of aerodynamic nonlinearities on aeroelastic behaviour, time integration methods based on the Transonic Small Disturbance (TSD) method, full Navier-Stokes and Euler equations have been developed. These methods can provide accurate approximations of the system behaviour, but in general they require very large computation times owing to the large number of variables and the characteristic integration times required to establish steady flow stability properties [1]. Direct methods formulated from the nonlinear Hopf bifurcation theory have been developed to compute flutter and LCO onset speeds of aeroelastic systems without a time-marching method, but they become computational expensive when the system becomes large [39].

This chapter considers some nonlinear aeroelastic problems studied up to now, with particular emphasis the stability analysis and thus to LCOs. In section 2.1 the basis of LCOs is given in a general form, with no consideration to aeroelastic applications. Then the problem is specialized to an aeroelastic system, discussing the main characteristics of such a phenomenon from the structural point of view, giving a short list of references of the problems encountered in real applications related to LCO. Then, section 2.1.1 analyses the LCO problem from a mathematical point of view as an initial value problem, so that given the initial condition, it is possible to compute the system evolution. Instead, in section 2.1.2, the LCO problem is formulated as a boundary value problem, where the boundary conditions are represented by the imposition of the solution periodicity and the imposition of the time origin. The solution proposed is formulated in the time domain, and the problem is solved through a periodic collocation in time, even if a frequency domain approach might also be possible.

2.1 Limit cycle oscillations

The change of the qualitative character of a solution of a nonlinear ODE with a varying parameter is known as *bifurcation*. This occours where a linear stability analysis yields an instability. The connection between the nonlinear and linear behaviour is through the *implicit function theorem*: the solution can be continued smoothly except where the system Jacobian matrix becomes singular, whereas new solutions are developed.

Limit cycle oscillations are a peculiar phenomenon encountered only in the analysis of nonlinear dynamic systems. In the context of the nonlinear system theory, an LCO is one of the simplest dynamic bifurcation, *"a first stop on the road to chaos"* [3].

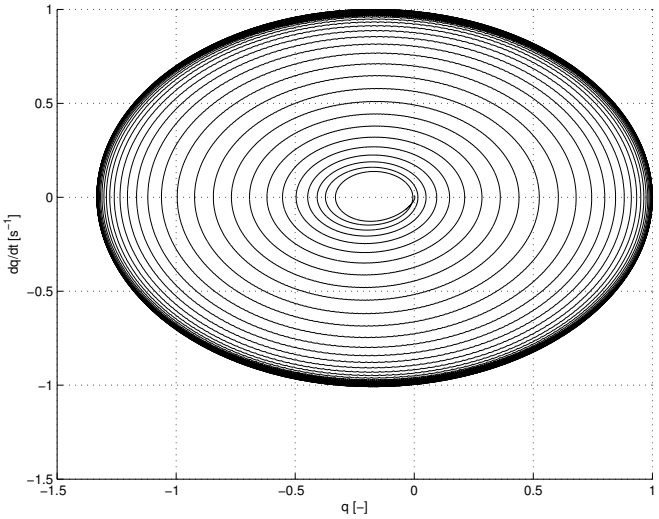


Figure 2.1: Example of a LCO orbit in the phase space

In the specific, an LCO is a closed orbit which can appear after a *Hopf bifurcation point* [40, 41, 39, 3, 42, 43, 44] of the dynamic system of interest, and here is described as follows. Given an autonomous nonlinear dynamic system, governed by the following model:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \dot{\mathbf{x}} = \mathbf{R}(\mathbf{x}; \lambda) \tag{2.1}$$

where the symbol λ represents a set of parameters, called *bifurcation parameters*, for a reason that will be clearer after the next few lines. $\mathbf{R}$ is a nonlinear function such that $\mathbf{R}(\mathbf{x}) : \mathbb{R}^N \longrightarrow \mathbb{R}^N$. The point at which the system (2.1) loses stability to time-oscillatory distrurbances is the Hopf bifurcation point. The stability exchange occours when a complex pair of eigenvalues of the system Jacobian matrix has a vanishing real part, while the real component of the other eigenvalues is negative.

The system (2.1) is characterized by a set of *equilibrium solutions*, $\tilde{\mathbf{x}}$, each of them satisfying:

$$\mathbf{R}(\tilde{\mathbf{x}}; \lambda) = \mathbf{0} \tag{2.2}$$

The linearized (local) stability study of an equilibrium solution consist in the analysis of the set of eigenvalues α of the system Jacobian matrix, $\mathbf{J}$, evaluated at the equilibrium point:

$$\mathbf{J}(\mathbf{x}; \lambda)|_{\tilde{\mathbf{x}}} = \left.\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right|_{\tilde{\mathbf{x}}} \tag{2.3}$$

The Jacobian matrix can be then used to perform a linear stability analysis of the equilibrium solution. For example consider the linearization of (2.1) near $\tilde{\mathbf{x}}$: $\mathbf{x} = \tilde{\mathbf{x}} + \Delta\mathbf{x}$; the linearized dynamics is governed by:

$$\Delta\dot{\mathbf{x}} = \mathbf{J}|_{\tilde{\mathbf{x}}} \Delta\mathbf{x} \tag{2.4}$$

since $\tilde{\mathbf{x}}$ is a time indipendent state, $\mathbf{J}|_{\tilde{\mathbf{x}}}$ is a constant matrix and its eigenvalues govern the growth rates of a linearized solution:

$$\Delta\mathbf{x} \propto \sum_{j=1}^{N} \mathbf{X}_j \exp(\alpha_j t) \tag{2.5}$$

Where α_j is the j-th eigenvalue of the Jacobian matrix evaluated at the equilibrium point, with $j = 1, \dots, N$, and $\mathbf{X}_j$ the associated eigenvector, which describes the character of the exponentially growing or decaying linearized solution. In the case that the linearized system eigenvalues are degenerate, i.e. $\alpha_j = \alpha_{j+1}$, two cases can be distingued. The first is that the Jacobian matrix can be brought to diagonal form, thus two linearly independent eigenvectors $\mathbf{X}_j$ and $\mathbf{X}_{j+1}$ are associated to the degenerate eigenvalue. The second possibility is that the Jacobian matrix can be brought to Jordan form, which has zeros everywhere except for the repeating eigenvalues on the diagonal and some elements equal to one directly above it. In such a case, relation 2.5 should be modified as:

$$\Delta\mathbf{x} \propto \sum_{j=1}^{N} \sum_{i=1}^{m_j} \mathbf{X}_j \, t^{i-1} \exp(\alpha_j t) \tag{2.6}$$

where m_j is the geometric multiplicity of the eigenvector $\mathbf{X}_j$.

Introducing the notation $\mathbb{R}e[\cdot]$ to indicate the real part of a number, if $\mathbb{R}e[\alpha_j] < 0$, $\forall j$, then the equilibrium point is stable to small perturbations. Instead, if exist a j such that $\mathbb{R}e[\alpha_j] > 0$, then the equilibrium point is unstable, resulting in a time dependent behaviour of the system. As λ changes, the onset of the instability occurs when there is at least an eigenvalue $\hat{\alpha}$ with null real part: $\mathbb{R}e[\hat{\alpha}] = 0$, with the other eigenvalues real part strictly negative. At this point, there are two possible classes of behaviour that will tipically occur as the parameter λ is changed:

- A single real eigenvalue passes through zero, this is a *stationary bifurcation*;
- A complex conjugate pair of eigenvalues passes through the imaginary axis in the complex plane, this is a *Hopf bifurcation.*

The presence of a pair of purely imaginary eigenvalues implies that a Hopf bifurcation can only occour in systems of dimension two or higher.

As stated in [41], if the pair $(\tilde{\mathbf{x}}^*, \lambda^*)$ satisfies the following three conditions, then from the equilibrium point will emerge a branch of unsteady solutions with limit-cycle behaviour:

- $\mathbf{R}(\tilde{\mathbf{x}}^*; \lambda^*) = \mathbf{R}^* = \mathbf{0}$;
- $\mathbf{J}(\tilde{\mathbf{x}}^*; \lambda^*)$ has a pair of purely imaginary eigenvalues (with no other eigenvalue having vanishing real part): $\alpha_\pm = \pm j\Psi$, with $\Psi \neq 0$;
- The transversality condition is satysfied: $\left.\dfrac{d\mathbb{R}e[(\alpha_\pm(\lambda))]}{d\lambda}\right|_{\lambda=\lambda^*} \neq 0$

If these conditions are fullfilled and there are no other eigenvalue pairs with positive real parts, the Hopf-point is a point of stability exchange [39].

For $\lambda > \lambda^*$ there is at least one exponentially growing solution to the linearized system (2.5). The long time solution of the full nonlinear system will be clearly affected by nonlinearity. These may saturate the oscillation growth, as in the case of a stable LCO, or may further enhance the growth, taking the state far away from the initial one even for λ close to λ^*, this is the case of an unstable LCO. The behaviour of the nonlinear system (2.1) near the bifurcation point can be represented by the so called *normal forms* [44], which are basically a set of dynamical equations for the amplitudes of the unstable eigenvectors. In the case of a Hopf bifurcation, the possible normal forms are:

- *Supercritical*: the nonlinearities are saturating, a constant equilibrium point solution is unstable and a new, stable solution develops. The orbit descripted in this case is a circle, the limit cycle;
- *Subcritical*: the nonlinearities are destabilizing, a constant equilibrium point solution is unstable and also unstable limit cycles develop.

Returning to the bifurcation point analysis, if N is small, the point can be tracked through the examination of all the eigenvalues of $\mathbf{J}$ with the varying parameter λ. This can be the case of a ROM, where the set of DOF is relatively small, in the order of hundreds or even tens, and the previous approach can be carried out with a small computational cost. With the proposed procedure, the Hopf-point solution is specified to be an oscillatory perturbation to the equilibrium solution $\mathbf{x}^*$ of the form $\mathbf{x} = \mathbf{x}^* + \epsilon \mathbf{X} \exp(\alpha t)$, the right hand side $\mathbf{R}$ can thus be expressed through a Taylor series expansion:

$$\mathbf{R}(\mathbf{x}; \lambda) = \mathbf{R}(\mathbf{x}^*; \lambda^*) + \epsilon \mathbf{J}(\mathbf{x}^*; \lambda^*) \mathbf{X} \exp(\alpha t) + \mathrm{o}\left(\epsilon^2\right) \tag{2.7}$$

with ϵ being a vanishing small parameter and $\mathbf{X}$ a coefficient vector. With an asymptotic series expansion analysis of (2.1), the following equations are obtained:

$$\begin{aligned} \mathbf{R}(\mathbf{x}^*; \lambda^*) &= \mathbf{0} \quad && \mathcal{O}\left(\epsilon^0\right) \\ \mathbf{J}^* \mathbf{X} &= \alpha \mathbf{X} && \mathcal{O}\left(\epsilon^1\right) \end{aligned} \tag{2.8}$$

with $\mathbf{J}^* = \mathbf{J}(\mathbf{x}^*; \lambda)$. The meaning of the previous relations is clear, at order zero the pair $(\mathbf{x}^*, \lambda^*)$ is an equilibrium solution for (2.1), meanwhile at the first order the eigenvalue problem associated to $\mathbf{J}^*$ represents the behaviour of the system near the equilibrium solution. The coefficient vector $\mathbf{X}$ is thus the eigenvector related to the eigenvalue β of $\mathbf{J}^*$. Instead, if N is not small, as in practical full order problems, a more sophisticated analysis as been proposed in [39]. The eigenvector has the general form $\mathbf{X} = \mathbf{X}_1 + j\,\mathbf{X}_2$, and at the Hopf bifurcation point $\alpha_\pm = \pm j\,\Psi$, thus expliciting the block structure of (2.8), the following two conditions are obtained:

$$\begin{aligned} \mathbf{J}^* \mathbf{X}_1 + \Psi \mathbf{X}_2 &= \mathbf{0} \\ \mathbf{J}^* \mathbf{X}_2 - \Psi \mathbf{X}_1 &= \mathbf{0} \end{aligned} \tag{2.9}$$

Moreover, at the Hopf point $\mathbf{X}_1$ and $\mathbf{X}_2$ are constrained by the following relations:

$$\begin{aligned} \mathbf{p}^T \mathbf{X}_1 &= 0 \\ \mathbf{p}^T \mathbf{X}_2 &= 1 \end{aligned} \tag{2.10}$$

with $\mathbf{p}$ a constant, not null, arbitrary vector. Thus at the Hopf point, the following nonlinear, algebraic system is satysfied:

$$\begin{cases} \mathbf{R}^* & = \mathbf{0} \\ \mathbf{J}^* \mathbf{X}_1 + \Psi \mathbf{X}_2 & = \mathbf{0} \\ \mathbf{J}^* \mathbf{X}_2 - \Psi \mathbf{X}_1 & = \mathbf{0} \\ \mathbf{p}^T \mathbf{X}_1 & = 0 \\ \mathbf{p}^T \mathbf{X}_2 - 1 & = 0 \end{cases} \tag{2.11}$$

in the unknowns $\mathbf{x}^*$, Ψ, $\mathbf{X}_1$, $\mathbf{X}_2$, λ. System (2.11) provides directly the Hopf bifurcation point. In principle, this system can be solved by the standard Newton's method, hovewer in [39] the authors have proposed an alternative approach in order to exploit the block structure of the Jacobian matrix of system (2.11). In this work, the first, direct approach will be used, since the size of the present ROM test problems are amenable for such a solution.

Let us now consider the LCO problem associated to aeroelastic systems, i.e. to the nonlinear interaction between the structural and aerodynamic loads acting on the affected aircraft components. This behaviour can be associated to the formation of large vortical flow structures, as in the case of low speed, high angle of attack flow regime or associated with complex shock wave motion in the transonic flow regime. However, LCOs of aeroelastic systems appear to prevail more in transonic than in subsonic flows. Therefore it has been thought that, at least for some configurations, the source of the nonlinearity that leads to an LCO resides in the aerodynamic [39]. Of course, structural nonlinearities can also lead to LCO whether the flow is transonic or not [23]. One of the advantages of studying theoretical models is that each of the several possible physical phenomena that may lead to LCO can be studied separately. Considering an inviscid transonic flow model, such as the one described by full potential and Euler models, with the latter being used in the following chapters, the principal nonlinear physical effect of interest is the relatively large motion of a shock wave as the amplitude of the wing pitch motion becomes sufficiently large. This in turn leads to a motion of the center of pressure. This movement may lead to an LCO rather then the catastrophic exponentially growing oscillations predicted by time-linearized aerodynamic models. Thus a nonlinear aerodynamic model allows large and more general shock motions, including the possible appearance and disappearance of a shock during a cycle of the wing motion [19, 39].

A limit cycle oscillation is a characterized by sustained periodic oscillations that neither increase nor decrease in amplitude over time for a given flight condition. It differs from the classical linear flutter in its tendency toward limited amplitude oscillations rather than diverging one [45]. Hovewer, the high-frequency and large-amplitude motion that results from LCO can cause the loss of the structural integrity in different parts of an airplaine, for example the wing, or in certain cases the fatigue damage is accumulated, like for aeronautical panels. Therefore, the prediction of this behaviour is of uttermost importance in the design of a modern airplane.

Different airplane models have experienced LCO, between them there are the *F-111*, the *F-16*, and the *F-18* models. All the three of them have experienced an LCO due to external stores, i.e. missiles, which is a case well studied in the literature, see [46]. More recently, the presence of a LCO involving in the first wing bending mode of the *Boeing 747-8* has delayed its certification. Such a phenomenon can induce different problems during the flight of the aircraft: first of all structural issues such as fatigue. The sustained and undamped oscillations can reduce the fatigue life of certain components of the aircraft, causing an early substitution of the component or a more frequent inspection. All of these solutions add an additional cost to the airplane manteinance. Second, if the stores under consideration are missiles or weapons in general, the resulting LCO can cause the lock of these elements or endenger the separation. Third, for a manned aircraft, the pilot workload is surely increased, augmenting the level of fatigue and stress of the pilot himself.

From the point of view of an aeroelastic analysis, the bifurcation parameter λ can be defined as the dynamic pressure $q = \frac{1}{2}\rho V_\infty^2$, since the change of the stability properties of an aeroelastic system depends primarily on this parameter. Another parameter is the Mach number, $M_\infty = \frac{V_\infty}{c_\infty}$, where c_∞ is the speed of sound. An aeroelastic stability analysis with varying M_∞ can enhance the presence of the celebrated *transonic dip*, qualitatively shown in figure 2.2.

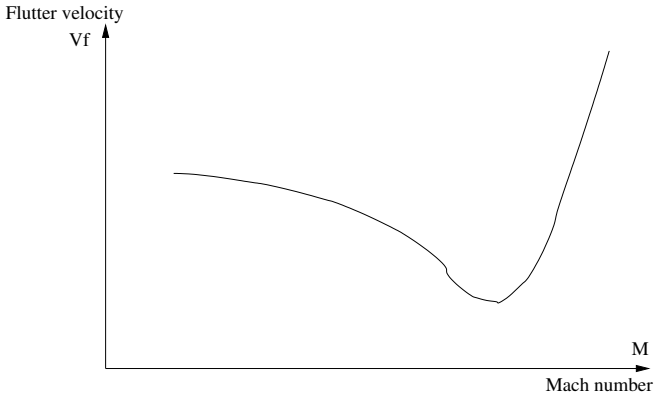


Figure 2.2: Example of transonic dip

It has been said that an LCO is characteristic of nonlinear systems. It is not possible to have a bifurcation point in a linear system, since the equilibrium state, if exists, is unique, thus no movement is allowed in the state space from one equilibrium point to an another. More specifically, the physical sources of nonlinearity in an aeroelastic problem can be of multiple types: control surface freeplays, geometric-structural nonlinearities, such as plate-like structures or wing with low aspect ratio. But the main source of nonlinearity reported in the literature [3, 19, 39, 45] are of the aerodynamic type: as said in transonic flows, large shock motions may lead to a nonlinear relationship between the structure motion and the resulting loads acting on it. Another source is a separated flow region, with the same consequence as the previous case.

A LCO can be *stable*, if following to a small distrubance the motion returns to the same LCO after a settling time, or *unstable*, if the consequence of a disturbance is the motion of the state from the previous LCO to a stable LCO, which *attracts* the system state. In this case there is also the possibility of obtaining diverging solutions. The strenght of the nonlinearities influences the response of the system. For example, in the case of weak shocks are present in the flow field, the dynamic response tends to an unstable divergent behaviour rather than a bounded response. Instead, in presence of strong nonlinearities, the response will be well bounded, with the characteristic LCO behaviour, as shown in figure 2.3. Thus, these finite amplitude oscillations can in some cases replace what would be the rapidly growing and destructive oscillations of a classical linear flutter behaviour.

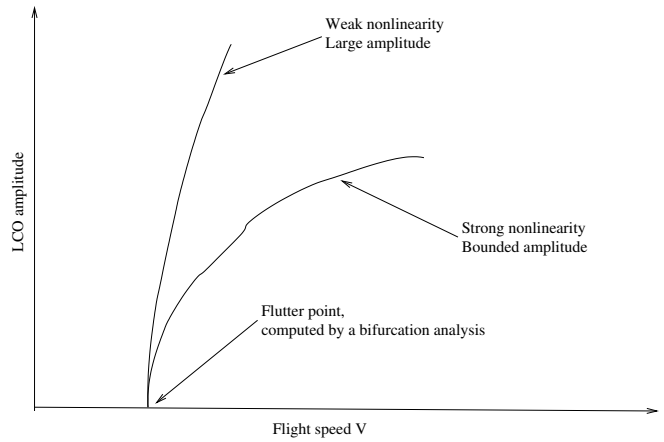


Figure 2.3: Comparison of LCOs with weak and strong nonlinearities

Although linear techniques can be used to predict the LCO frequency, they cannot consistently predict where within the flight envelope the onset of the oscillation will occur. However, numerical linearized models can be effectively applied in order to compute accurately classical aeroelastic phenomena, such as flutter boundary and gust response [3].

An LCO response can be obtained following two main lines. 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. However, such approach remains computationally very expensive, especially during the conceptual and preliminary aircraft design phases, where fast simulation are needed in order to evaluate the performance of a large set of configurations required by the regulations. The other approach is based to an enforcement of a periodic solution of the nonlinear system under consideration through a direct periodic collocation in the time domain. This approach is surely less expensive from a computational point of view, but requires an a priori estimation of a tentative solution for the LCO of interest.

2.1.1 LCO as an initial value problem

Time-marching integration is the primary method for determining the flutter point in numerical experiments involving transonic aerodynamics [39]. Today such methods are mainly used to validate the results of stability changes obtainied with an Hopf bifurcation analysis with varying dynamic pressure.

System (2.1) is a first order, nonlinear, dynamic model, and with the assignment of initial conditions, it can be integrated forward in time:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{R}\left(\mathbf{x}; \lambda\right) & t > 0 \\ \mathbf{x}\left(t = 0\right) = \mathbf{x}_0 \end{cases} \tag{2.12}$$

The integration can be performed with explicit or implicit schemes. While explicit methods are the first choice for their semplicity of implementation, they are limited on the maximum time step by stability issues. The latter methods permit larger time integration steps, with the burden of solving a nonlinear algebraic problem at each time step. This fact don't trouble the analysis if the system has a small size or $\mathbf{R}$ has a simple analytic expression.

Specializing to aeroelastic applications, tipically if the analysis point is beyond the bifurcation point the initial condition can be even homogenous: the numerical perturbations introduced by the discretization method automatically induce the LCO behaviour. However, tipically a small not null initial mode velocity is given as initial condition, in order to give a perturbation that will seed the final system response.

A good criteria to find out an LCO with direct simulation is to compute the linear bifurcation point in terms of frequency and characteristic modes as explained previously and then follow these results in the simulation. In certain cases, the mode computed by the bifurcation analysis is given as initial condition, in terms of position and velocity. In other cases the system can be forced for a certain time interval with the bifurcation mode at a frequency near to the Hopf one, then the system is left free to move and the LCO solution should develop. Such an approach can be found in [47].

For example a time-marching method can integrate the solution to either a steady state or a stable LCO to determine if a bifurcation point has been crossed. Bracketing a single flutter point may require several time-integration solutions, each consisting of hundred of thousand explicit time step iterations [19]. However such simulations can be easily parallelized and therefore they can be very cheap in term of required time. It is clear that the computation of the bifurcation point is the basis for a time marching simulation, since without the knowledge of the existence of a bifurcation point the analysis of different LCO conditions can be very difficult.

2.1.2 LCO as a boundary value problem

Since LCO is a periodic (*neutrally stable*) solution of a nonlinear system as (2.1) it can be computed both in the time and frequency domain. The frequency domain approach can be exploited using a Fourier series. Typically, considering only one fundamental harmonic the resulting approximation has proved adequate, leading to the well known Describing Function method [22, 23]. An improvement of the solution can be obtained by considering a small set of harmonics, in the order of three/four. Different examples are available in the literature [19], proving that the effect of higher order harmonics is often negligible and with the consideration of a single harmonic the results obtained are qualitatively correct and have a fair quantitative accuracy. An interesting work which focuses on LCOs of aeroelastic systems with multiple lumped nonlinearities can be found in [23]. In such article an incremental complexity approach is considered in order to identify limit cycles and evaluate their stabilities. First a solution based on dual-input describing function is computed, considering a single harmonic. Then the stability of the LCO is evaluated through an extension of the single-input describing function "quasi-static" method, leading to an approach quite similar to the well-estabilished existing methods used to evaluate linear flutter conditions directly. Finally, in case that higher harmonics are required in the analysis, an extended harmonic balance method based on a numerical minimization in the frequency domain is adopted and the stability of the computed solution is evaluated through a Floquet analysis.

On the other hand, the time domain approach can be followed casting the problem in the time domain and solving with a periodically collocated discretization [21, 48]. The latter is described here, since has been proved as being more general and flexible. Although it leads to a larger number of equations than the Fourier series approach, it results in a largely sparse system of equations, whose solution can be more efficient than any harmonic approximation.

First let us recast system (2.1) in the following implicit form:

$$\mathbf{F}\left(\mathbf{x}, \dot{\mathbf{x}}\right) = \mathbf{0} \tag{2.13}$$

Suppose that $\mathbf{F} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ admits an LCO with period T. In order to impose a periodic solution it is necessary to divide the period T in N time intervals, thus $N+1$ time nodes. The time intervals Δt can be non-uniform, hence they can be defined by a vector α such that:

$$\Delta t_i = \alpha_i T, \qquad i = 1, ..., N \tag{2.14}$$

For each time interval, (2.13) can be time-weighted collocated at the corresponding instant, for example with the Mid-Point Rule (MPR), reading:

$$\mathbf{F}_{\mathrm{MPR}_i} = \mathbf{F}\left(\frac{\mathbf{x}(t_{i+1}) + \mathbf{x}(t_i)}{2}, \frac{\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i)}{\Delta t_i}\right)\Delta t_i = \mathbf{0} \tag{2.15}$$

The MPR is a symplectic, energy preserving, second-order integrator, thus can be efficiently adopted for solving problem (2.13). Hovewer, if system (2.13) is stiff in time or it is composed by a set of Differential Algebraic Equations (DAE), there is the possibility that the MPR converge toward a ringing solution, which can jeopardize the convergence. In order to solve this problem, in [49] an hybrid method that mixes the characteristics of the MPR with the second-order Backward Difference Formula (BDF2) has been proposed:

$$\mathbf{F}_{\mathrm{BDF}_i} = \mathbf{F}\left(\mathbf{x}\left(t_i\right), \frac{\rho_{1_i}\mathbf{x}\left(t_{i+1}\right) + \rho_{2_i}\mathbf{x}\left(t_i\right) + \rho_{3_i}\mathbf{x}\left(t_{i-1}\right)}{\Delta t_i}\right)\Delta t_i = \mathbf{0} \tag{2.16}$$

where:

$$\rho_i = \frac{\alpha_i}{\alpha_{i+1}} \qquad \rho_{1_i} = 1 + \frac{\rho_i}{\rho_{i+1}} \qquad \rho_{2_i} = \rho_i + 1 \qquad \rho_{3_i} = \frac{\rho_i^2}{\rho_i + 1} \tag{2.17}$$

The term ρ_i permits the adaptivity of the numerical grid, which can be very useful when the correct solution is tracked and the related refinement is required. The hybridization is carried out by means of the parameter $\beta : 0 \leq \beta \leq 1$:

$$\beta \mathbf{F}_{\mathrm{MPR}_i}\left(\mathbf{x}\left(t_i\right), \mathbf{x}\left(t_{i+1}\right)\right) \Delta t_i + \left(1-\beta\right) \mathbf{F}_{\mathrm{BDF}_i}\left(\mathbf{x}\left(t_{i-1}\right), \mathbf{x}\left(t_i\right) \mathbf{x}\left(t_{i+1}\right)\right) \Delta t_i = \mathbf{0} \tag{2.18}$$

With the coefficient β which tends toward 0, the integration method gains the properties of the BFD method which can be summarized in a good dissipation of the high frequency content of the solution. This added dissipation worsen the precision, but avoid the problems introduced by a mere application of the MPR [49]. Discretization (2.15) leads to $N\,n$ equations in $(N+1)\,n+1$ unknowns: the $N+1$ vector solutions $\mathbf{x}(t_i)$ and the period T. 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 for imposing the periodicity:

$$\mathbf{x}(t_1) = \mathbf{x}(t_{N+1}) \tag{2.19}$$

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

$$x_k(t_j) = C \tag{2.20}$$

where $C \in \mathbb{R}$.

The resulting system of nonlinear algebraic equations, obtained by the evaluation of (2.13) at the collocation points, can be finally solved with an iterative nonlinear solver, such as Newton-Raphson techinque.

Thus the boundary value problem can be formulated as follow:

$$\begin{cases} \mathbf{F}\left(\mathbf{x}, \dot{\mathbf{x}}; T\right) = \mathbf{0} \\ \mathbf{x}(0) = \mathbf{x}(T) \\ x_k(0) = C \end{cases} \tag{2.21}$$

Periodic solutions can be obtained also with $(n+1)\,N$ unknown, since $\mathbf{x}\left(t_{N+1}\right)$ can be directly assembled onto $\mathbf{x}\left(t_1\right)$, but such approach has not be implemented by the authors in [23] and can be considered as a future development of such method, analysing the possible influence on its convergence properties. In practical applications solution efficiency is usually enhanced by adopting a continuation on the number of time interval N used, starting from a low number of points and then increasing them using an interpolation of the previous solution as initial guess. This is called Periodic Collocation Method (PCM), and allows the adaption of variable time step sizes to better fit the nonlinear terms present in (2.13). Further continuations are then possible if the system depends on one or more parameters, as in the case of an aeroelastic problem [49, 21].

Following the approach given in [49], in order to make easier the assignment of the state component consistently on the base of rough estimates of the LCO amplitude only, such component can be written at the first time node as:

$$x_k(t_1) = C - \sum_{i=1}^{N} \frac{x_k(t_i)}{n} \tag{2.22}$$

With such assignment it is not necessary to take care of any non-null average value of the k-th component of the state.

Since the Newton-Raphson algorithm needs the computation of the Jacobian matrix, it is shown here its structure, which is composed by separated blocks, if the unknowns are ordered in the following order:

$$\mathbf{X} = \left(\mathbf{x}^T(t_1), \mathbf{x}^T(t_2), \ldots, \mathbf{x}^T(t_{N+1}), T\right)^T \tag{2.23}$$

The associated Jacobian matrix of (2.13) reads as:

$$\mathbf{J} = \begin{bmatrix} \mathbf{A}_1^1 & \mathbf{A}_1^2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{b}_1^1 \\ \mathbf{A}_2^1 & \mathbf{A}_2^2 & \mathbf{A}_2^3 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{b}_2^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_{i-1}^k & \mathbf{A}_i^k & \mathbf{A}_{i+1}^k & \cdots & \mathbf{0} & \mathbf{b}_i^k \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{N-1}^N & \mathbf{A}_N^N & \mathbf{A}_{N+1}^N & \mathbf{b}_N^N \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{2.24}$$

where the blocks have the following expressions:

$$\begin{aligned} \mathbf{A}_{i-1}^k &= (1-\beta)\,\frac{\rho_3}{\alpha_k T}\,\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{MPR, k}} \\ \mathbf{A}_i^k &= \beta\left(\frac{1}{2}\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{MPR, k}} - \frac{1}{\alpha_k T}\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{MPR, k}}\right)\Delta t_i + (1-\beta)\frac{\rho_2}{\alpha_k T}\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{BDF, k}}\Delta t_i \\ \mathbf{A}_{i+1}^k &= \beta\left(\frac{1}{2}\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{MPR, k}} + \frac{1}{\alpha_k T}\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{MPR, k}}\right)\Delta t_i + (1-\beta)\left(\frac{1}{2}\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{BDF, k}} + \frac{\rho_1}{\alpha_k T}\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{BDF, k}}\right)\Delta t_i \end{aligned} \tag{2.25}$$

and

$$\mathbf{b}_i^k = \mathbf{F}\left(\mathbf{x}(t_i), \dot{\mathbf{x}}(t_i)\right) - \left[\beta\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{MPR, k}}\dot{\mathbf{x}}_{\text{MPR}}^k + (1-\beta)\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{BDF, k}}\dot{\mathbf{x}}_{\text{BDF}}^k\right]\alpha_k \tag{2.26}$$

where the subscript MPR means that the unknown at the corresponting time node is approximated through the MPR, the opposite is valid for the subscript BDF. The complete Jacobian matrix of (2.13) is obtained eventually appending to (2.24) the derivatives of (2.22) with respect to the collocated unknowns. Such a method has proved to be particularly efficient, since the Jacobian matrix is usually easy to compute and moreover it is *sparse*, therefore requiring a limited number of operations. Remember that the terms $\dfrac{\partial \mathbf{F}}{\partial \mathbf{x}}$ and $\dfrac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}$ have to be computed: depending on the specific problem this can be done by finite differences or by exploiting the related analytic expression. However it is usually easy to include the Jacobian matrix computation in a numerical routine.

Once a converged solution has been obtained, the *monodromy* matrix $\mathbf{Q}$ is computed in order to investigate the LCO stability within the framework of the *Floquet*'s theory [40, 43]. Let us consider again (2.13), it can be linearized around the LCO solution such as:

$$\mathbf{F}|_{\text{LCO}} + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{LCO}}\Delta\mathbf{x} + \left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{LCO}}\Delta\dot{\mathbf{x}} = \mathbf{0} \tag{2.27}$$

Since the LCO solution is an equilibrium solution for (2.13), equation (2.27) can be rewritten in the following form:

$$\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}\right|_{\text{LCO}}\Delta\dot{\mathbf{x}} + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{LCO}}\Delta\mathbf{x} = \mathbf{0} \tag{2.28}$$

which is a first order periodic ordinary differential system. Such a dynamic system can be written in the standard form, well known thorugh the literature [43]:

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{P}(t)\Delta\mathbf{x}(t) \qquad \text{with} \quad \mathbf{P}(t) = -\left.\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}^{-1}\right|_{\text{LCO}}\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\text{LCO}} \tag{2.29}$$

Where $\mathbf{P}(t)$ satisfies of course the periodicity condition $\mathbf{P}(t+T) = \mathbf{P}(t)$ for all t, since the Jacobian matrices evaluated at the LCO are both periodic. Let us combine all the independent

solutions of the above time periodic ODE in an unique matrix $\mathbf{X}$, called the *fundamental* matrix of the system, defined as:

$$\mathbf{X} = [\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \cdots, \Delta\mathbf{x}_n] \tag{2.30}$$

where $\Delta\mathbf{x}_i$ is the solution of (2.28) to an homogeneous initial condition but the i-th term, which is taken unitary. Therefore, (2.28) can be read as:

$$\begin{cases} \dot{\mathbf{X}} = \mathbf{P}\mathbf{X} \\ \mathbf{X}(0) = \mathbf{I} \end{cases} \tag{2.31}$$

Thanks to Floquet's theorem [43], the fundamental matrix evaluated for $t = T$ is related to its initial value by means of a nonsingular matrix, which is indeed the monodromy matrix:

$$\mathbf{X}(T) = \mathbf{X}(0)\mathbf{Q} \tag{2.32}$$

And the eigenvalues of $\mathbf{Q}$ are called the characteristic numbers of (2.28), which are constant, independently from the choice of $\mathbf{X}$. The evaluation of such elements permits to characterize the stability of the LCO solution obtained. This can be achieved by the solution of the following linear system [49]:

$$\begin{bmatrix} \mathbf{A}_2^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_2^2 & \mathbf{A}_3^2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{A}_{i-1}^k & \mathbf{A}_i^k & \mathbf{A}_{i+1}^k & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{N-1}^N & \mathbf{A}_N^N & \mathbf{A}_{N+1}^N \end{bmatrix} \begin{pmatrix} \mathbf{X}(t_1) \\ \mathbf{X}(t_2) \\ \vdots \\ \mathbf{X}(t_i) \\ \vdots \\ \mathbf{Q} \end{pmatrix} = - \begin{pmatrix} \mathbf{A}_1^1 \\ \mathbf{A}_1^2 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{2.33}$$

A LCO will be stable if the modules of the eigenvalues of the monodromy matrix $\mathbf{Q}$ have a norm lower than one, however for such solutions there is always an eigenvalue equal to one associated with the perturbations along the LCO in the phase space [23, 40, 43]. If at least one eigenvalue has a module higher than one the solution is unstable. It is worth to be remarked that in case of unstable solutions, if the highest eigenvalue is near one, a finest discretization of the time grid is required to do not attribute the unstable behaviour due to a poor convergence of the collocation method.

Chapter 3

A neural network based reduced order model

For a correct and accurate prediction of the aerodynamic loads in the transonic regime, the flow field should be discretized in space with very small computational cells, thus in order to fill the computational domain the number of these cells can easily reach values around 10^4 for small application and beyond 10^6 for practical industrial applications. Considering that for a three dimensional simulation there are at least 5 unknowns for each cell, if one doesn't take into account possible turbulence models, the number of unknowns that has to be computed at each time step can be quite a burden even with the computational power available nowdays. Moreover, if a time accurate solution of an unsteady flow field is required, the number of computational operations required for one time step is multiplied by the number of time steps employed, resulting in a huge number of computer operations. To worsen the situation, if the boundary displacements are not small, mesh motion should be included in the model, leading to possible problems related to the distorsion of the numerical grid, introducing more mathematical operations that have to be solved at each time step in any case. As a consequence, the computational time required for such practical analyses is so high that simpler, and less accurate, models are often adopted in the prediction of the aerodynamic loads during the conceptual and preliminary design phases of an aircraft.

This chapter treats the problem of the nonlinear model order reduction applied to an aeroelastic system, in order to show a possible approach to reduce the time needed for performing typical aeroelastic analyses, such as stability assessment or system response to an external input, using a model characterized by a small number of DOF, but based on high-fidelity CFD results, thus mantaining the same level of accuracy.

First the problem of the fluid dynamics modeling of a continuous flow is considered, with an explanation in section 3.1 of the models that can be adopted. Then a Finite Volume (FV) approach is discussed in section 3.1.1 in order to discretize in time and space the fluid dynamics model. An extensive description of the aerodynamic solver chosen for the analysis is then given in section 3.1.2. Some possible approaches to the reduced order modeling problem associated to the resulting high order dynamical system are then proposed in sections 3.2 and 3.4, with a deep explanation of the theory associated to recurrent neural networks formulated in a continuous time domain. The training algorithm adopted is explained in section 3.3, with an extension regarding the computation of different elements required during the network training. Finally a discussion about the possible training signals that can be used to obtain the best generalization of the network properties is carried out in section 3.5, discussing the chosen signal tipology on the base of the problem that has to be solved in chapter 5.

3.1 Aerodynamic modeling

Many modern aircrafts, civil and military, are designed for working in a transonic flow, characterized by a flight Mach number included in the interval $M_\infty \in (0.7, 1.4)$. A transonic flow condition is characterized by the presence of subsonic and supersonic regions in the flow field, separated by shocks and expansion waves, as shown in figure 3.1.

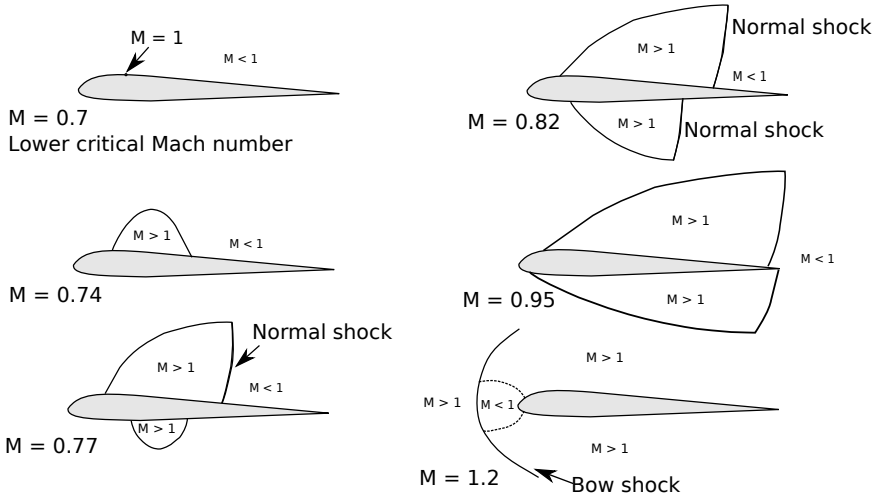


Figure 3.1: Typical transonic flow conditions

Compressibility plays a fundamental role in this phenomenon, thus the mathematical model used in this case is that of the *Full Navier-Stokes* system, written here in *conservative* form:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = \nabla \cdot \mathbf{\Pi} \\ \dfrac{\partial E^t}{\partial t} + \nabla \cdot \left(\left(p + E^t \right) \mathbf{u} \right) = \nabla \cdot (\mathbf{\Pi} \mathbf{u}) - \nabla \cdot \mathbf{q} \end{cases} \tag{3.1}$$

where: ρ is the air density, $\mathbf{u} = (u, v, w)^T$ is the flow velocity vector, E^t is the total energy density, defined as the sum of the internal energy density ρe and the kinetic energy density $\frac{1}{2}\rho|\mathbf{u}|^2$ and p is the thermodynamic pressure. All these variables depend on the position $\mathbf{x}$ and on the time t. The vector $\mathbf{q}$ represents the heat conduction flux, which can be expressed in terms of the temperature gradient by the Fourier's law: $\mathbf{q} = -\mathbf{k}\nabla T$, with $\mathbf{k} = \mathbf{k}(T)$ being the thermal conductivity matrix. The tensor $\mathbf{\Pi}$ is the viscous stress tensor, which, considering a newtonian fluid, can be modeled by a linear relationship to the deformation rate gradient $\mathcal{D}$:

$$\mathbf{\Pi} = \mu \mathcal{D} + \lambda (\nabla \cdot \mathbf{u}) \mathbf{I} \tag{3.2}$$

with $\mu = \mu(T)$ defined as the dynamics viscosity of the fluid, $\lambda = \lambda(T)$ its volume viscosity and $\mathcal{D}$ is defined as:

$$\mathcal{D} = \frac{1}{2} \left(\nabla \otimes \mathbf{u} + (\nabla \otimes \mathbf{u})^T \right) \tag{3.3}$$

Actually, the theromodynamic pressure p is a function of others thermodynamic variables, such as internal energy, density, entropy, etc. Thus $p = p(\rho, e)$ for example, and, since two state equations are needed for defining the thermodynamic state of a gas, another relation, for example a relation for the temperature $T = T(e, \rho)$ must be added for the mathematical closure of the

problem. In the particular case of a Polytropic Ideal Gas (PIG), the two equation reads as:

$$p = (\gamma - 1)\rho e \qquad T = \frac{e}{c_v} \tag{3.4}$$

where c_v is the specific heat at constant volume, which is a constant in the PIG case and γ is the ratio between the specific heat at constant pressure (c_p) and volume c_v.

At high Reynolds number, the fluid viscosity and thermal conductivity are negligible if no flow separation is occuring on the body surface. Then the *Euler* model can be used, written here in *conservative* form:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = \mathbf{0} \\ \dfrac{\partial E^t}{\partial t} + \nabla \cdot \left(\left(p + E^t\right) \mathbf{u} \right) = \mathbf{0} \end{cases} \tag{3.5}$$

which is an PDE *hyperbolic* system that permits the resolution of shocks, contact discontinuities and rarefaction waves, if written in *integral* form [50, 51].

In order to solve the differential problem defined by (3.1) or (3.5), the correct initial and boundary condition should be imposed. The initial condition must specify in the whole domain the full solution in terms of ρ, $\rho\mathbf{u}$, E^t at $t = 0$. The boundary conditions depend on the physical model adopted for representing the fluid flow. If the Navier-Stokes model (3.1) is used, the no-slip condition ($\mathbf{u} = -\mathbf{V}_{\text{body}}$) has to be imposed on all the bodies present in the flow field. Instead, if the Euler model (3.5) is considered, the slip condition is the only boundary condition that can be applied on the present bodies, thus only the normal component of the velocity with respect of the body surface can be imposed $\mathbf{u}^T\hat{\mathbf{n}} = -\mathbf{V}_{\text{body}}^T\hat{\mathbf{n}}$, where $\hat{\mathbf{n}}$ is the normal versor pointing outward the fluid domain. Moreover, since the problem is hyperbolic, external boundary condition can be imposed only along the *inflow* boundary, which extension depends on the solution itself. Different methods are available in the literature for treating this problem [20, 13, 52].

In aeroelastic applications, the analyst is interested in the evaluation of the aerodynamic load acting on the body surface rather than the solution in the entire flow field. This can be obtained integrating the total surface stress, composed, in the case of system (3.1), by the thermodynamic pressure p which acts normally to the surface, therefore with a stress equal to $p\hat{\mathbf{n}}$, and the viscous stress $\tau = \mathbf{\Pi} \cdot \hat{\mathbf{n}}$ which acts tangentially to the surface, along the whole body surface. Instead, if the inviscid flow model of (3.5) is used, than p is the only stress to be integrated.

If one is interested in the linear stability characterization, i.e. flutter, of the aeroelastic system in a transonic flow condition, it has to be considered that the solution of the unsteady flow field around the aircraft and the aerodynamic loads calculation involves considerable difficulty: in such situations the Doublet Lattice Method (DLM), widely employed for subsonic analyses, cannot be considered reliable given that the assumptions on which it is based are no longer met. Conversely it is necessary to adopt mathematical models and numerical methods belonging to the class of the Computational Fluid Dynamics (CFD), discretizing in space and time the aerodynamic problem of interest, modeled for example with a full potential model, or in case of shocks with the Euler model, making a posteriori numerical linearization of the system [3].

A more general approach, that permits to overcome the limitations of a linear system representation, considers the full nonlinearity of the aerodynamic system, thus large shock motion and separated region are allowed, permitting the analysis of particular nonlinear responses, such as LCOs.

3.1.1 Numerical solution of the aerodynamic problem

To numerically solve the aerodynamic problem is necessary to discretize the system (3.1) or (3.5) in space and time. As regards the spatial discretization it is possible to adopt the method of the Finite Volumes (FV), which consists in placing the problem written in *integral* form and discretizing the physical domain with a finite number of computational cells; in particular it is possible to distinguish the *node-centered* strategy, for which the solution is placed in correspondence of each node of the computational grid, and the *cell-centered* strategy, for which the solution is placed in correspondence of the geometrical center of gravity of each element of the grid. If the first strategy is more simple from the conceptual and implementation point of view, the latter permits to use a number of unknowns lower and to impose boundary conditions more simply [13]. For what regards the time discretization is possible to use both explicit and implicit time marching methods [20], with the latter providing a computational time significantly lower despite a more complex implementation.

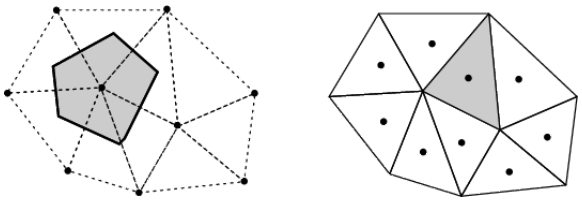


Figure 3.2: Vertex-Centered and Cell-Centered schemes

Calling $\mathbf{w}$ the vector of the conservative variables, $\mathbf{w} = \left(\rho, \rho\mathbf{u}, E^t\right)^T$, system 3.1 can be written in the following compact form:

$$\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathbf{F}\left(\mathbf{w}\right) = \mathbf{0} \tag{3.6}$$

Where $\mathbf{F}\left(\mathbf{w}\right)$ is called the *flux* of the system, and contains all the terms under the divergence operator in 3.1. Such a representation is valid pointly if $\mathbf{w}$ is smooth over all the flow field. This is not the case when shocks are present in the solution, thus a more general formulation is needed for accounting such phenomena. For this reason finite volumes codes are written considering the integral formulation of the equations of motion, which permits to account also discontinuous solutions. The related *integral form*, after having exploited the divergence theorem, reads as

$$\frac{d}{dt}\int_{\Omega} \mathbf{w} d\Omega = -\int_{\partial\Omega} \left[\mathbf{F}_c\left(\mathbf{w}\right) - \mathbf{F}_v\left(\mathbf{w}\right)\right] dS \tag{3.7}$$

Where Ω is the internal volume of the control molecule, meanwhile $\partial\Omega$ being its lateral surface. The vector $\mathbf{F}_c$ is related to the convective transport of quantities in the fluid, and it is usually refered as *convective flux*. Instead vector $\mathbf{F}_v$ contains the viscous stresses as well as the heat diffusion. The expression of such vectors is reported below:

$$\mathbf{F}_c\left(\mathbf{w}\right) = \begin{pmatrix} \left(\mathbf{u}\cdot\hat{\mathbf{n}}\right)\rho \\ \left(\mathbf{u}\cdot\hat{\mathbf{n}}\right)\rho\mathbf{u} + p\hat{\mathbf{n}} \\ \left(\mathbf{u}\cdot\hat{\mathbf{n}}\right)\rho H \end{pmatrix} \qquad \mathbf{F}_c\left(\mathbf{w}\right) = \begin{pmatrix} 0 \\ \boldsymbol{\Pi}\cdot\hat{\mathbf{n}} \\ \left(\boldsymbol{\Pi}\mathbf{u}\right)\cdot\hat{\mathbf{n}} - \mathbf{q}\cdot\hat{\mathbf{n}} \end{pmatrix} \tag{3.8}$$

Where $H = E^t + \frac{p}{\rho}$ is the *total enthalpy* and $\hat{\mathbf{n}}$ is the outward normal vector of $\partial\Omega$. If the flow model assumed is the inviscid one, then the viscous flux must be set to zero ($\mathbf{F}_v = \mathbf{0}$). The integral form permits to obtain solutions with a grade of regularity smaller, thus the treatment of discontinuos solutions, i.e. shocks, is possible. The expression of the fluxes depends on the flow model chosen.

In certain applications, where for example the fluid-structure interaction is investigated, like in aeroelasticity, it is necessary to solve the governing equations on moving and possibly also deforming grids. Therefore it is necessary to give an appropriate kinematical description of the

continuum which occupies the physical domain. Two possible approaches have been proposed in the literature: the Arbitraly Lagrangian Eulerian (ALE) formulation and the dynamic grid method. Both the formulations lead to the same set of equations [20]. The resulting flow model takes into account the relative motion of the grid with respect to the fluid itself. Considering a control volume $\Omega = \Omega(t)$, variable with the time, the system of governing equations (3.7) can be recasted in a similar form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathbf{w} d\Omega = -\int_{\partial\Omega(t)} \left(\bar{\mathbf{F}}_c - \mathbf{F}_v\right) dS \tag{3.9}$$

Where $\bar{\mathbf{F}}_c$ is the corrected convective flux, defined as:

$$\bar{\mathbf{F}}_c = \mathbf{F}_c - (\mathbf{v}_g \cdot \hat{\mathbf{n}})\,\mathbf{w} \tag{3.10}$$

With $\mathbf{v}_g(\mathbf{x}, t)$ being the local velocity of the moving boundaries, which of course has a negative contribution to the convective flux. Instead the viscous fluxes vector remains unaltered. Since the grid is now deforming, besides the mass conservation law and the momentum and energy balance equations, the so called *Geometric Conservation Law* (GCL) must be satisfied in order to aviod errors induced by the deformation of the control volomes. Its integral formulation reads:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathrm{d}\Omega - \int_{\partial\Omega(t)} \mathbf{v}_g \cdot \hat{\mathbf{n}} \mathrm{d}S = 0 \tag{3.11}$$

The GCL results from the requirement that the computation of the control volumes or of the grid velocities must be performed in such a way that the resulting numerical scheme preserves the state of a uniform flow, independently from the deformation of the grid.

Let us begin from the spatial discretization of the problem. After having created a computational mesh, which is composed by cells that fill-up the computational domain, the solution is averaged in each control volume and the integral of the flux across the cell surface is approximated with a summation. Thus defining the *average* vector of the conservative variables as:

$$\tilde{\mathbf{w}}(t) = \frac{1}{\Omega}\int_{\Omega} \mathbf{w}(\mathbf{x}, t) d\Omega \tag{3.12}$$

The following system of Ordinary Differential Equation (ODE) is obtained:

$$\frac{d\left(\Omega\tilde{\mathbf{w}}\right)}{dt} = -\sum_{m=1}^{N_F} \left(\mathbf{F}_c - \mathbf{F}_v\right)_m \Delta S_m = -\mathbf{R} \qquad \text{For each control volume} \tag{3.13}$$

where N_F is the number of faces of the cell considered and $\mathbf{R}$ is called the *residual*. If the cell volume remains constant, it can be brought to the right hand side of 3.13. The value of the fluxes at the m-th face of the control volume is determined after the choice of the flux discretization scheme, which tipically is one of the various *upwind* schemes present in the literature, since their good accuracy near possible discontinuities in the flow field [20].

The resulting ODE system should be discretized in time by a time-marching integrator scheme. Typically an explicit scheme is adopted, where the solution $\tilde{\mathbf{w}}^{n+1}$ at the time step t_{n+1} depends only on the solution at the previous time step $\tilde{\mathbf{w}}^n$. The most popular and widespread explicit methods by far are the multi-stage Runge-Kutta (RK) time-stepping schemes. The RK scheme advances the solution in a number of steps, so called *stages*, which can be viewed as a sequence of updates. Its basic implementation of a $m-$ stages scheme is reported below, for a

non-moving grid:

$$\begin{aligned}
\tilde{\mathbf{w}}^{(0)} &= \tilde{\mathbf{w}}^{n} \\
\tilde{\mathbf{w}}^{(1)} &= \tilde{\mathbf{w}}^{(0)} - \alpha_1 \frac{\Delta t}{\Omega}\mathbf{R}^{(0)} \\
\tilde{\mathbf{w}}^{(2)} &= \tilde{\mathbf{w}}^{(0)} - \alpha_2 \frac{\Delta t}{\Omega}\mathbf{R}^{(1)} \\
\tilde{\mathbf{w}}^{(3)} &= \tilde{\mathbf{w}}^{(0)} - \alpha_3 \frac{\Delta t}{\Omega}\mathbf{R}^{(2)} \\
&\vdots \\
\tilde{\mathbf{w}}^{n+1} &= \tilde{\mathbf{w}}^{(m)} = \tilde{\mathbf{w}}^{(0)} - \alpha_m \frac{\Delta t}{\Omega}\mathbf{R}^{(m-1)}
\end{aligned} \tag{3.14}$$

The coefficients α_j, $j = 1, \dots, m$ are called as *stage coefficients*, and depend by the specific method used. Moreover, $\mathbf{R}^{(j)}$ is the residual evaluated on the solution at stage j. Of course, the above scheme must be executed for all the computational cells involved.

Regardless of the strategy used, it is necessary to ensure that the numerical solution remains limited for each instant of time considered, requiring that the numerical scheme is absolutely stable. While an implicit integration method is absolutely stable regardless of the interval of integration time Δt, for an explicit method the absolute stability is guaranteed only as long as the limit Courant-Friedrichs-Lewy (CFL) condition is satified [20].

When the ALE formulation of the governing equations is considered, it is essential that the GCL (3.11) is temporally discretized using the same scheme as it is applied to the governing equations (3.9) in order to obtain a self-consistent solution method [20].

In practice, when the ALE formulation of the flow model is employed, it is necessary to numerically implement an aeroelastic interface that allows to provide the closed loop connection between the structural system and the aerodynamic system by an appropriate interpolation process, since the two problems are tipically defined over domains with different topology. A robust and efficient interface acting on boundaries belonging to solid walls has been proposed in [13]: it is based on a Moving Least Squares (MLS) approach. Thanks to this interface it is possible to interpolate the movements and the speed of the structural nodes belonging to the contour of the body on the aerodynamic mesh. Once that these information has been translated from the structural to the aerodynamic model, The internal nodes of the computational aerodynamic mesh should be moved accordingly. The simplest and most correct approach is to deform the aerodynamic computational mesh, but this approach is also the most costly from the computational point of view. This approach, for example, can be followed considering a continuum analogy method, which transforms the computational mesh in a elastic elements, characterized by different Young modulus for each cell, based on the position and dimension of the cell itself. The resulting problem is solved forward in time, computing the new position of the mesh's nodes. A less expensive strategy is to change the slip boundary condition in the case of Euler's flow model, $\mathbf{u}^T\hat{\mathbf{n}} = -\mathbf{V}_{\text{body}}^T\hat{\mathbf{n}}$ evaluated on the body surface, by assigning a non-zero value of the normal speed V_n, said transpiration velocity, so as to reproduce the effects of geometrical and kinematic movement of the contour without actually deform the computational mesh. The main limitation of this approach is of course the assumption of small displacements of the contour [13]. An intermediate solution in term of computational cost between the full deformation of the mesh and the transpiration condition is the possibility of interpolating the mesh displacements from the boundary patches to the internal points with an Inverse Distance Weighting (IDW) scheme [53].

It has been chosen to use a free license software for the resolution of the aerodynamic problem. The aerodynamic solver OpenFOAM (which stands for Open Field Operation And Manipulation), developed by *OpenCFD Ltd.* since 1993 with the aim of having a flexible software for numerical simulation using the FV method on multi-disciplinary problems of industrial interest, focusing

in particular on issues related to the fluid dynamics branch. The current version, OpenFOAM *v.1.7.1* [54], is widely used in academical and industrial environment. Since the year 2004, the aerodynamic solver OpenFOAM is freely available online together with the source code at the web site http://www.openfoam.com under the GNU General Public License (GPL). Furthermore, a wide but not complete documentation is available online, which includes an user operations guide, a programming guide and a guide to the source code.

For a high fidelity modeling of the aerodynamic problem it has been chosen the in-house solver AeroFoam, developed in [13] and successively redisigned in [53], which is supported by OpenFOAM libraries only for the management of the computational grid data, for the computation of the numerical solution and for the pre/post-processing phase.

It is a 2D/3D Euler/RANS equations density-based coupled solver for aero-servo-elastic applications in Arbitrary-Lagrangian-Eulerian (*ALE*) formulation for moving grids. It is a FV, *cell-centered* solver, that can treat both structured and unstructured grids.

AeroFoam is the first density-based RANS solver implemented within the framework of OpenFOAM, realized for overcoming the limits of built-in pressure-based solvers in the transonic regime (i.e. *sonicFoam*), since their *non-conservative* formulation does not permit to solve accurately transonic and supersonic regimes of motion. It is well known [20, 52] that in order to accurately reproduce the evolution of any unsteady phenomena, such as shocks, it is always important to write the fluid dynamics problem in conservative form. Within the framework of AeroFoam, the convective fluxes are discretized by the classical Roe's approximated Riemann solver, which is a first order, monotone scheme [52], blended with a centered approximation, such as Lax-Wendroff scheme, giving a second order, high-resolution scheme, completed by the entropy fix of Harten and Hyman and the flux limiter by van Leer [20]. Viscous and conductive fluxes are treated without approximations. The time discretization can also be chosen with a I or II order accuracy, with a wide choice between explicit multi-step Runge-Kutta (RK) schemes, up to 5 stages. Local Time-Stepping (LTS), Residual Smoothing (RS), Dual Time-Stepping (DTS) for unsteady problems and Multi-Grid (MG), such as Full Multi-Grid (FMG) and Full Approximation Storage (FAS) options are available for speed-up the convergence of the simulation [20, 53].

Since they have been widely used in the resolution of the unsteady, transonic, aerodynamic problem considered here, the RS, DTS and MG methods will be shortly explained below. A complete treatment of these arguments can be found in [20].

In time-accurate unsteady simulations, the Courant-Friedrichs-Lewy (CFL) number influences the maximum time step allowed by the numerical integration method used: as the maximum CFL is increased, the simulation is allowed to perform time steps much longer. Explicit time-marching schemes are tipically used for the integration of the ODE system: these numerical methods are simple to implement, but they are characterized by relatively low CFL numbers. Therefore, both the maximum CFL number and the convergence properties of the explicit multi-stage time-stepping scheme can be influenced by optimizing the stage coefficients of the time integrator scheme (3.14). The RS technique is introduced with the aim to lend to the explicit scheme an implicit character and hence increasing the maximum allowable CFL number. This is accomplished by filtering the residual through a Laplacian smoothing operator, before updating its value at the next time step. A further purpose of the residual smoothing is a better damping of the high-frequency error components of the residual, of particular importance for a successful application of the MG method. Residual smoothing can be implemented both in an explicit and implicit manner. It is usually applied at each stage of the explicit time-stepping scheme (3.14).

Since the simulation of unsteady flow phenomena is becoming increasingly important in many engineering disciplines, a fast and efficient methodology for the solution of such problems, such as DTS has been proposed in the literature [20]. Explicit schemes represent the best choice for certain unsteady applications, for example when the time scales are comparable to the spatial

scales over the eigenvalue, i.e., when the physical CFL number is of the order of unity. This is for example the case of aeroacoustics, Direct Numerical Simulation (DNS), and Large Eddy Simulation (LES). Since the global physical phenomena evolve much slower than the local changes of the solution, it is necessary to integrate over a long period of the physical time. In order to obtain a high accuracy, the temporal resolution of the explicit scheme has to be of III or higher order. In other cases, where the physical time scales are large in comparison to the spatial scales divided by the eigenvalue, i.e. the flutter analysis, the CFL number can be chosen in the order of several hundreds or even thousands without impairing the accuracy of the simulation. In practice a *dual* (also referred to as *pseudo*) time stepping τ methodology is employed [20]. Consider for example the time discretization of 3.13, on a non-moving spatial grid, by means of the 2-order Backward Difference Formula (BDF2):

$$\frac{\tilde{\mathbf{w}}^{n+1} - 3\,\tilde{\mathbf{w}}^{n} + 2\,\tilde{\mathbf{w}}^{n-1}}{2\,\Delta t} + \mathbf{R}\left(\tilde{\mathbf{w}}^{n+1}\right) = \mathbf{0} \qquad \text{for each cell} \tag{3.15}$$

The dual time step and the corrispective state $\tilde{\mathbf{w}}^{*}$ are introduced, deriving:

$$\frac{d\tilde{\mathbf{w}}^{*}}{d\tau} + \frac{\tilde{\mathbf{w}}^{*} - 3\,\tilde{\mathbf{w}}^{n} + 2\,\tilde{\mathbf{w}}^{n-1}}{2\,\Delta t} + \mathbf{R}\left(\tilde{\mathbf{w}}^{*}\right) = \frac{d\tilde{\mathbf{w}}^{*}}{d\tau} + \mathbf{R}^{*}\left(\tilde{\mathbf{w}}^{*}\right) = \mathbf{0} \tag{3.16}$$

In each inner iteration the solution is advanced in the pseudo time. If in (3.16) the additional pseudo time derivative vanishes, it becomes (3.13). This approach leads to large physical time steps Δt, which should speed up the time integration. In AeroFoam, DTS is implemented in an implicit form, like the just explained approach.

The MG method [55, 56], permits to solve the set of governing equations (3.13) over a series of coarser grids, accelerating the convergence toward the steady-state value of the solution. The solutions computed on the coarser grids are combined in order to update the solution on the finest one.

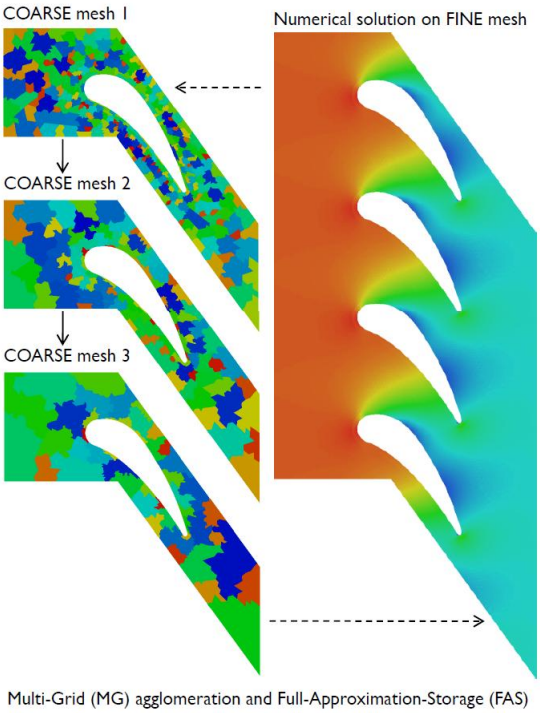


Figure 3.3: Multi-Grid methodology

Thus, this methods employs coarse grids in order to *drive* the solution on the finest grid faster to the steady state. MG permits to use large time steps, since the larger control volume on

the coarser grids. Moreover, it is well known that the numerical integration schemes employed in the solution of problem (3.13) damps very well the high frequencies of the solution error [20, 57], therefore coarse grids damp well the low frequencies of the finer grids, accelerating the convergence. Different approaches to MG hs been implemented as Algebraic Multi-Grid (AMG), where no new grid topology is constructed/stored, the algorithms act only on the operators of the numerical integrator; or the *semicoarsening*, where the grid is coarsened only along a characteristic direction.

A qualitative description of the MG method is given in the following list:

- The solution $\tilde{\mathbf{w}}^n$ and the residual $\mathbf{R}\left(\tilde{\mathbf{w}}^n\right)$ at the current time step are transfered from the finest grid to the coarser grids, through the use of an *interpolation* operator. The residual on the coarser grids is corrected by a forcing function;
- The solution at the next time step is computed on the coarser grids, using the integration scheme adopted on the finer grids;
- This solution is interpolated back up to the finer grids, through a *prolungation* operator, obtaining the physical solution at the next time step $\tilde{\mathbf{w}}^{n+1}$.

In order to obtain a time-accurate load estimation, the physical domain should be discretized with a very high number of computational cells. As a consequence, the computational time required for solving the system (3.13) at each cell and at each time step can be huge, in the order of days, even months, depending on the application and above all on the computational power available. For such reasons, state of the art CFD simulation are nowadays seldom used in the conceptual and preliminary phases of an aircraft design, where the a lot of aircraft configuration should be tested in a relatively small time, in order to evaluate which the best way that has to be followed in the next design phases. However, transonic aerodynamic effects are characteristic of the flow field around a modern jet-liner, and these effects can be accurately predict in the design phase primarily by CFD methods, since transonic wind-tunnel tests are very expensive. Computational parallelization is possible on all the CFD commercial softwares available nowadays, and also in AeroFoam, permitting a considerable time saving regarding the required simulations. However, if the project of a controller system, or some component optimization, or a stochastic characterization of the aeroelastic system is required, a method that permits to save simulation time, mantaining the accuracy of a state of the art CFD method is very attractive.

The proposed ROM satisfy the requests. From a single time-accurate CFD simulation, the nonlinear, dynamics system constructed is able to predict accurately the aerodynamic loads acting on a body, permitting to evaluate, once the aerodynamic systems is coupled to the structural one, the stability characteristics of the resulting aeroelastic system, or also the nonlinear response to external input, such as moving surfaces or wind gust. Of course, in order to obtain a ROM which is able to generalize the response to a generic input, the training signal must be chosen very carefully, exiting all the frequencies and movement amplitudes of interest, as will be explained in the following section.

3.1.2 Case setting in AeroFoam

In this section an example of the files organization in the OpenFOAM environment specialized to the application of the solver AeroFoam is proposed, trying to explain the basic functionalities of the different solution modules.

Consider that a fluid dynamics problem, as a transonic flow past an airfoil, should be solved with the FV method. Thus all the files regarding this problem will be inserted in the folder `TestCase`. In this folder there are present all the instructions that have to be delivered to the solver algorithms in order to solve the problem. A standard structure of the data contained in the folder `TestCase` is given in figure 3.4.

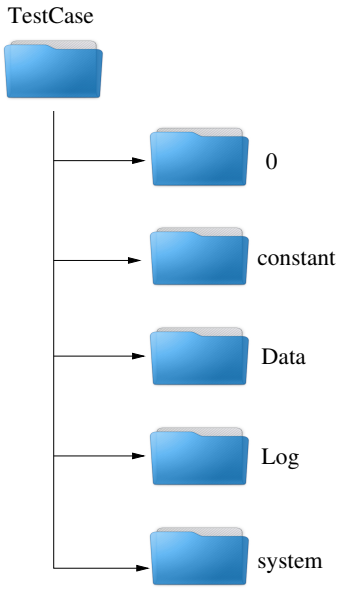


Figure 3.4: Structure of an AeroFoam case

The different subfolders are tipically organized as follows:

- The initial and boundary data of the problem is stored in folder `0`. Thus in the different files `0/p`, `0/U`, `0/T` the initial and boundary conditions for the pressure, the velocity and the temperature rispectively are stored;

- All the files that regard the computational mesh are stored in folder `constant`. Moreover also the thermodynamic properties of the fluid are stored in different files as in a standard OpenFOAM case. The file `constant/dynamicMeshDict` contains all the information regarding the variation of the computational mesh, in terms of deformation or movements in general of the mesh itself. The file `constant/interfaceDict` contains all the instruction that are passed to the solver for what regard the movement of the moving boundaries. Depending on the interface selected in the file `system/controlDict` the movement of the boundary is transfered to the internal mesh through the `Modal` interface if the movement of the structure (that in the aerodynamic problem represents a solid boundary) is expressed in terms of its modal shapes, or through the `Rigid` interface if the structure's movement is expressed in terms of rigid translations and rotations. Finally, the file `constant/pluginDict` contains the instruction regarding the identification of particular inputs and outputs: the ones of interest are selected and their time-accurate values are recorded in the folder `Log`;

- The structure's modal shapes and particular inputs signal that can be called by the file `constant/pluginDict` are stored in folder `Data`;

- All the results of the simulation are stored in folder `Log`, in a text format. This means that at the end of the simulation a time-accurate list of the displacements (rigid or modal) and the loads (rigid or modal) will be available for further analysis;

- The folder `system` contains the *core* of the numerical simulation. The file `system/fvSchemes` as its name recalls, contains all the numerical schemes used to discretize system (3.1) or (3.5). The file `system/fvSolution` contains the numerical methos used to solve the nonlinear algebraic problem that arises in the solution of the discretized version of the continuous models. Finally the file `system/controlDict` calls all the instructions specified in the

different files shortly explained above and permits thus of controlling the test case from one single file.

An example of `system/controlDict` is given below:

```
//.........................Start and End times
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         20;

//.........................AeroFoam parameters
physics         Euler;          // Flow model: Euler vs. RANS
turbulence      off;            // Turbulence model: SpalartAllmaras
                                // vs. KappaOmega
solver          MultiGrid;      // MultiGrid vs. TimeStepping
moving          ALE;            // Moving mesh formulation
interface       Modal;          // Aeroelastic interface

//.........................Explicit-Time-Stepping solver parameters
timeScheme      RK5(1);         // Time integration scheme
timeStepping    localCFL;       // Time stepping strategy
deltaT          1e-6;           // Time step of the simulation
DTS             ( 1 1.0e-3 100 ); // Additional parameters for DTS
CFL             1.0;            // CFL number
MinMax          1e-6;           // Min-Max ratio on timestep,
                                // min(dt)/max(dt)
smoothingWeight 0.25;           // Residual smoothing factor epsilon
smoothingLoops  0;              // Residual smoothing iterations

//.........................Multi-Grid solver parameters
MultiGrid       basic;          // Control strategy: basic, advanced
agglomeration   point;          // Agglomeration strategy: MGridGen,
                                // GAMG, face, point, block
weights         ( 0.5 0.75 1.0 1.0 0.0 ); // CFL, Restrict.,
                                          // Forcing, Prolong., Smooth.

//.........................Basic Multi-Grid parameters
cycle           FAS(V);         // Type of Multi-Grid cycle:
                                // FMG, FAS(V), FAS(W), FAS(F)
levels          2;              // Number of mesh levels
                                // (finest grid is labelled with 0)
iterations      1;              // Number of maximum iterations
residuals       -1;             // Minimum residual: if non-positive
                                // this option is ignored
```

3.2 Continuous Time Recurrent Neural Networks (CTRNN)

Neural networks are a powerful tool used in the context of the nonlinear system modeling, since their *intrinsic* nonlinearity and the presence of the *universal approximation theorem*, which guarantee that a recurrent neural network written in the state-space form can approximate any nonlinear dynamics system, considering an adequate number of hidden neurons [25, 34].

Moreover, neural networks have the attractive property of permitting a *black-box* modeling of the system, without any prior information on it, the only data available is the input-output pairs, in this case computed by a CFD solver.

A lot of applications of *static* neural networks, presented briefly in 1, is present in the literature, regarding character recognition, elaboration of signals, associative memory, identification and

control. Such neural networks work as a nonlinear map from N_{input}-dimensional space to the N_{output}-dimensional space. Adding recurrent connection with tapped delays, such neural framework becomes a *dynamics* map, useful in modeling dynamics systems. Considering a recurrent architecture, it is possible to limit the number of neurons compared to a static network, saving time during the network training.

The possible choices in the recurrent neural network (RNN) modeling are two: discrete-time (DTRNN) [25, 31, 32, 33, 26, 58, 59] and continuous-time (CTRNN) [27, 28, 29, 34]. DTRNN have been extensively used in aeroelasticity in the context of identification and control of deformable aircrafts [59, 58], but no application is present in the literature regarding CTRNN.

CTRNNs introduce further advantages over DTRNNs, even if they are both represented on a computer in a discrete form. Eventually the use of a DTRNN causes a great dependence of the resulting model on the sampling period adopted for training the network, with no information given regarding the model trajectories between the sampling instants. Instead, for a CTRNN the sampling period can be varied without the need of re-training [27]. For these reasons a CTRNN model is used in this work.

No standard parametrization of a generic nonlinear system as:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \end{cases} \tag{3.17}$$

is available nowadays in the literature, thus different formulations are available [27, 28, 25], each one can be adapted to a specific kind of problem. The neural model considered in this thesis has the following form [27]:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x \mathbf{\Phi}\left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u} + \mathbf{b}_1\right) + \mathbf{b}_2 \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{3.18}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$, $\mathbf{W}^x \in \mathbb{R}^{n_x \times n_h}$, $\mathbf{\Phi} : \mathbb{R}^{n_h} \longrightarrow \mathbb{R}^{n_h}$, $\mathbf{b}_1 \in \mathbb{R}^{n_h}$, $\mathbf{W}^a \in \mathbb{R}^{n_h \times n_x}$, $\mathbf{W}^b \in \mathbb{R}^{n_h \times m}$, $\mathbf{b}_2 \in \mathbb{R}^{n_x}$ and $\mathbf{W}^c \in \mathbb{R}^{p \times n_x}$; with n_x, n_h, m and p the dimensions of the state space, hidden space, input space and output space rispectively. The explicit dependence on the time t is dropped here as shorthand notation.

Note that the direct transmission term $\mathbf{W}^d \mathbf{u}$ has never been employed in all the consulted references, maybe for simplicity and for mantaining a limited number of unknowns during the training. However it can be interesting to consider a network output of the type $\mathbf{y} = \mathbf{W}^c \mathbf{x} + \mathbf{W}^d \mathbf{u}$, in analogy to standard system theory. We left this consideration to a future development.

The vector-valued function $\mathbf{\Phi}$ is so defined:

$$\mathbf{\Phi}(\mathbf{z}) = \begin{pmatrix} \phi_1(z_1) \\ \phi_2(z_2) \\ \vdots \\ \phi_{n_h}(z_{n_h}) \end{pmatrix} \tag{3.19}$$

where $\phi_i(z)$ is the activation function of the neuron i. Since the network is formulated in a state-space form, it inherits the properties of the universal approximator of a nonlinear dynamics system, as stated by the *universal approximation theorem* [29, 34]. A classical representation of such network, more frequent in the literature than the matrix formulation, is given below:

$$\dot{x}_j = \sum_{i=1}^{n_h} W^x_{ji} \phi_i \left(\sum_{k=1}^{n_x} W^a_{ik} x_k + \sum_{q=1}^{m} W^b_{iq} u_q + b_{1_i} \right) + b_{2_j} \qquad j = 1, \dots, n_x \tag{3.20}$$

A logical scheme of the network is shown in figure 3.5.

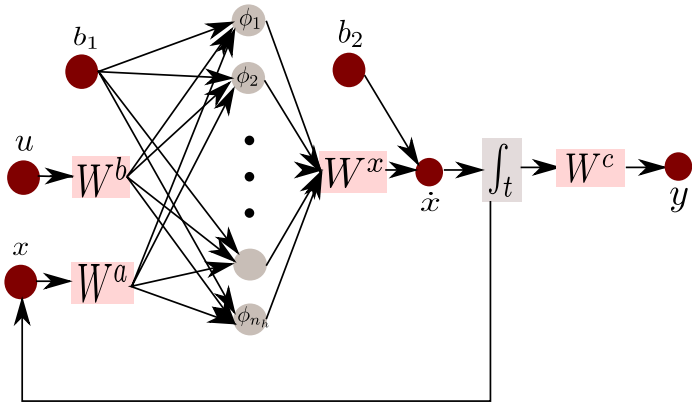


Figure 3.5: An example of CTRNN

The meaning of the synaptic weight matrices should be clear: they combine the input and the internal state to produce the neuron output, a bias being added in order to shift the origin of their activity value. The neuron activation function are sigmoid functions, i.e. hyperbolic tangent or the logistic function. The additional weight matrix $\mathbf{W}^x$ is introduced in order to increase the combination of the states, with the aim of reducing the number of the needed neurons.

A continuos-time state-space representation guaratees a certain form of "symmetry" in the aeroelastic problem formulation: the structural dynamics, represented by a set of ODE is coupled directly to the neural model. In this case the input to the CTRNN is the motion of the structure, meanwhile its output is the aerodynamic load acting on the body surface. Thus the aeroelastic model is represented in a state space form, and it can be discretized once for all in order to perform the numerical integration.

The output-transmission matrix is taken as:

$$\mathbf{W}^c = \begin{bmatrix} \mathbf{I}_{p\times p} & \mathbf{0}_{p\times(n_x-p)} \end{bmatrix} \tag{3.21}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{0}$ the null matrix and the subscript shown the matrices dimensions. The structure of (3.21) means that the output of the first p hidden neurons are the output of the network, which permits to introduce a direct physical interpretation to the results computed [25].

The logistic function has been chosen as activation function:

$$\phi(v) = \frac{1}{1+\exp(-v)} \qquad \phi'(v) = \frac{\exp(-v)}{(1+\exp(-v))^2} \tag{3.22}$$

the first derivative of the activation function will be needed during the training algorithm.

3.3 Training algorithm

The greatest effort in neural network modeling is without doubt 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. Both algorithms have proved good to posses convergence properties [25].

No standard procedure has been defined for the CTRNN case yet. The RTRL has been proposed also in the continuous case [29], but it has been proved that the learning based on a gradient method, such as BTTP and RTRL, is extremely hard [60, 30]. Thus higher order optimization algorithms can be used in this case, such as the Levenberg-Marquardt (LM) optimization method [61] or a Genetic Algorithm (GA) [62].

The training of a neural network can be viewed as a nonlinear optimization problem, thus, in order to solve a problem like this, the computation of the Jacobian matrix matrix is needed. The Jacobian matrix should be intended as the derivative of the system output with respect to the parameters that have to be changed in order to minimize a certain cost function, i.e. the synaptic weights of (3.18). Let us define:

$$\mathbf{\Lambda} = \frac{\partial \mathbf{x}}{\partial \theta} \tag{3.23}$$

as the *state* Jacobian matrix, where $\theta = \left(\mathrm{vec}\left(\mathbf{W}^x\right)^T, \mathrm{vec}\left(\mathbf{W}^a\right)^T, \mathrm{vec}\left(\mathbf{W}^b\right)^T, \mathbf{b}_1^T, \mathbf{b}_2^T\right)^T$, with $\mathrm{vec}\left(\cdot\right)$ the operator that orders a matrix stacking its columns. An example makes the things easier to understand:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \mathrm{vec}\left(\mathbf{A}\right) = \begin{pmatrix} a \\ c \\ b \\ d \end{pmatrix} \tag{3.24}$$

Moreover, since the system is dynamic, also the Jacobian matrix will change in function of the time. Thus a system of n_θ ODEs will be added and coupled to the original n_x ODEs of (3.18) during the network training. The analytical computation of such matrix is detailed in the next section.

3.3.1 Analytical computation of the state Jacobian matrix

Having defined the state Jacobian matrix in (3.23), it is easy to compute its dimensions: it is a $\mathbb{R}^{n_x \times n_\theta}$ matrix, with $n_\theta = 2n_x \cdot n_h + n_h \cdot m + n_h + n_x$. For a straightforward computation of its elements, it becomes easier to split the matrix in blocks:

$$\mathbf{\Lambda} = \begin{bmatrix} \dfrac{\partial \mathbf{x}}{\partial W_{ij}^x} & \dfrac{\partial \mathbf{x}}{\partial W_{ij}^a} & \dfrac{\partial \mathbf{x}}{\partial W_{ij}^b} & \dfrac{\partial \mathbf{x}}{\partial b_{1_j}} & \dfrac{\partial \mathbf{x}}{\partial b_{2_j}} \end{bmatrix} \tag{3.25}$$

Where each sub-matrix have the following structure:

$$\frac{\partial \mathbf{x}}{\partial W_{ij}^x} = \left[\frac{\partial \mathbf{x}}{\partial W_{11}^x}, \frac{\partial \mathbf{x}}{\partial W_{21}^x}, \ldots, \frac{\partial \mathbf{x}}{\partial W_{n_x 1}^x}, \frac{\partial \mathbf{x}}{\partial W_{12}^x}, \frac{\partial \mathbf{x}}{\partial W_{22}^x}, \ldots, \frac{\partial \mathbf{x}}{\partial W_{n_x 2}^x}, \ldots, \frac{\partial \mathbf{x}}{\partial W_{n_x n_h}^x} \right] \tag{3.26}$$

Taking as example the derivative of the state $\mathbf{x}$ with respect of the elements of the synaptic weights matrix $\mathbf{W}^x$. Remembering the size of each vector/matrix involved and the order with they have been sorted, the assembling is straightforward. Defining $\mathbf{I}_{ij}$ the single-entry matrix [63], as the matrix with 1 at the position (i, j) and zero elsewhere, and as shorthand notation $\mathbf{z} = \mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u} + \mathbf{b}_1$, then the direct computation of the blocks of (3.25) from (3.18) reads as:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W_{ij}^x} = \mathbf{I}_{ij}\mathbf{\Phi}(\mathbf{z}) + \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\mathbf{W}^a \frac{\partial \mathbf{x}}{\partial W_{ij}^x} \qquad \begin{matrix} i = 1, \ldots, n_x \\ j = 1, \ldots, n_h \end{matrix} \tag{3.27}$$

with $\mathbf{\Phi}'(\mathbf{z}) = \text{diag}_{n_h}\left[\phi_1'(z_1), \phi_2'(z_2), \cdots, \phi_{n_h}'(z_{n_h})\right]$, then

$$\frac{\partial \dot{\mathbf{x}}}{\partial W_{ij}^a} = \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \left(\mathbf{I}_{ij}\mathbf{x} + \mathbf{W}^a \frac{\partial \mathbf{x}}{\partial W_{ij}^a} \right) \qquad \begin{matrix} i = 1, \dots, n_h \\ j = 1, \dots, n_x \end{matrix} \tag{3.28}$$

$$\frac{\partial \dot{\mathbf{x}}}{\partial W_{ij}^b} = \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \left(\mathbf{I}_{ij}\mathbf{u} + \mathbf{W}^a \frac{\partial \mathbf{x}}{\partial W_{ij}^b} \right) \qquad \begin{matrix} i = 1, \dots, n_h \\ j = 1, \dots, m \end{matrix} \tag{3.29}$$

$$\frac{\partial \dot{\mathbf{x}}}{\partial b_{1_j}} = \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \left(\mathbf{I}_j + \mathbf{W}^a \frac{\partial \mathbf{x}}{\partial b_{1_j}} \right) \qquad j = 1, \dots, n_h \tag{3.30}$$

with $\mathbf{I}_j$ the single-entry vector, 1 at the position j and zero elsewhere, finally:

$$\frac{\partial \dot{\mathbf{x}}}{\partial b_{2_j}} = \mathbf{I}_j + \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \mathbf{W}^a \frac{\partial \mathbf{x}}{\partial b_{2_j}} \qquad j = 1, \dots, n_x \tag{3.31}$$

assembling the different blocks, let us define:

$$\mathbf{U} = \begin{bmatrix} \mathbf{I}_{ij}\mathbf{\Phi}(\mathbf{z}) & \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \mathbf{I}_{ij}\mathbf{x} & \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \mathbf{I}_{ij}\mathbf{u} & \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \mathbf{I}_j & \mathbf{I}_j \end{bmatrix} \tag{3.32}$$

and

$$\hat{\mathbf{A}} = \mathbf{W}^x \mathbf{\Phi}'(\mathbf{z}) \mathbf{W}^a \tag{3.33}$$

thus the Jacobian matrix dynamics is governed by the following matrix differential equation:

$$\dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \tag{3.34}$$

At this point, the coupling of (3.18) and (3.34) defines the *nonlinear state dynamics* of the network:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x \mathbf{\Phi}\left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u} + \mathbf{b}_1 \right) + \mathbf{b}_2 \\ \dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{3.35}$$

Depending on the number of Jacobian matrix elements to be computed, this sensitivity computation can require more than the 90% of the total computation time required for solving a training problem, constituting a real bottleneck for the training algorithm [27].

The training of the network should find out the optimal synaptic weights minimizing a certain cost function. The cost function used here is the most common found in the literature, i.e. a quadratic function of the error $\mathbf{e}(t)$, defined as:

$$\mathbf{e}(t) = \hat{\mathbf{y}}(t) - \mathbf{W}^c \mathbf{x}(t) \tag{3.36}$$

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

$$F = \frac{1}{2} \int_{t_0}^{t_1} \mathbf{e}^T(t) \mathbf{e}(t)\, dt \tag{3.37}$$

Of course, since the output of the real system is available only at certain sampling points, then the related discretized cost function is:

$$F = \frac{1}{2} \sum_{k=1}^{N_t} \mathbf{e}^T(t_k) \mathbf{e}(t_k) \tag{3.38}$$

where N_t is the number of sampling points. The optimization algorithm will find the minimum of the cost function with respect to the variables, the synaptic weights in this case:

$$\min_{\theta} F = \min_{\theta} \frac{1}{2} \sum_{k=1}^{N_t} \mathbf{e}^T(t_k, \theta) \mathbf{e}(t_k, \theta) \tag{3.39}$$

where the dependece on the optimization variables θ has been made explicit. In case some part of the time history is of particular importance, for example the steady state value, the previous cost function 3.38 may be modified introducing a related set of weights $\hat{\mathbf{W}}(t_k)$, tipically a diagonal matrix, which permit to ponder the prominence of the istantaneous quadratic error:

$$\hat{F} = \frac{1}{2}\sum_{k=1}^{N_t} \hat{\mathbf{W}}(t_k)\mathbf{e}^T(t_k)\mathbf{e}(t_k) \tag{3.40}$$

Since the LM algorithm is very robust close to the optimum solution but somewhat weak with a starting point θ_0 far away from it, a hybrid technique has been implemented, running a few iterations of the GA, which is a global optimization algorithm, and then, when the solution is sufficiently close to the optimum, the LM algorithm is resumed. It is well known that global methods are good at finding regions while local methods are good at finding points [26]. The advantage of this hybrid method is that the GA does not need the Jacobian matrix computation in order to converge, meanwhile the LM algorithm is known to be a fast optimization method since its almost 2^{nd} order converging, without the need of the computation of the Hessian matrix.

The GA is a method for solving optimization problems based on natural selection. It takes an initial population for the vector variable θ and evaluate the cost function (3.38) for each element (or *individual*) of the current generation. Then it assignes a score to each individual and those with the highest score are immediately upgraded to the next generation (*elite children*), meanwhile the others are mixed in both deterministic and random manner. The deterministic mixing employs the *recombination* option, which cuts the current individual in two or more parts and through a crossover function mixes their characteristics. Such a recombination is governed by a crossover rate, determining the probability with which the crossover is performed. The points where the individuals are cut is chosen randomly, the only constraint present is that the cut must not happen at the extremes of such individuals. Instead, the random mixing employes the *mutation* option, which simply inverts randomly some elements of an individual, obtaining thus a new individual for the new generation. Also the mutation option is moduled by the mutation rate, which has the same function of the crossover rate. The main purpose of mutation is to prevent the GA from getting stuck in certain regions of the parameter space [26]. In this way a new generation is built up, composed by *children*, obtained through a mix of recombination and mutation of the individuals of the previous generation, which can be called *parents*. Along with this process, the *population* evolves toward an optimal solution, where the cost function (3.38) is stationary from one iteration to the other or its value is below a certain threshold [62]. At this point the algorithm can be stopped and the converged result can be adopted as initial guess for the LM algorithm.

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

$$\mathbf{E}(\theta) = \begin{pmatrix}\mathbf{e}(t_1,\theta)^T & \mathbf{e}(t_2,\theta)^T \cdots \mathbf{e}(t_{N_t},\theta)^T\end{pmatrix}^T \tag{3.41}$$

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

$$\mathbf{J}(\theta) = \frac{\partial \mathbf{E}(\theta)}{\partial \theta} \tag{3.42}$$

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

$$\mathbf{J}(\theta) = \begin{bmatrix}\mathbf{J}(t_1,\theta)^T & \mathbf{J}(t_2,\theta)^T \cdots \mathbf{J}(t_{N_t},\theta)^T\end{bmatrix}^T \tag{3.43}$$

each of dimensions $\mathbb{R}^{p\times n_\theta}$, defined as:

$$\mathbf{J}(t_k,\theta) = \frac{\partial \mathbf{e}(t_k,\theta)}{\partial \theta} = -\mathbf{W}^c\mathbf{\Lambda}(t_k,\theta) \qquad k = 1,\dots,N_t \tag{3.44}$$

Then the LM algorithm can be applied, its n-th iteration reads:

$$\Delta\theta_n = -\left[\mathbf{J}(\theta_n)^T\mathbf{J}(\theta_n) + \mu\mathbf{I}\right]^{-1}\mathbf{J}(\theta_n)^T\mathbf{E}(\theta_n) \tag{3.45}$$

with the updating step:

$$\theta_{n+1} = \theta_n + \Delta\theta_n \tag{3.46}$$

where μ is a fundamental scalar, which, when it is very small, make the algorithm act like a *modified Newton-Raphson* method, approximating the Hessian with the term $\mathbf{J}(\theta_n)^T\mathbf{J}(\theta_n)$, meanwhile, when μ is very large, the LM algorithm works like to a gradient descent method, with small steps in (3.46) [25, 61].

For both the optimization algorithms used, a ODE system must be solved forward in time, with the current value of the network synaptic weights. Initial conditions for (3.35) are thus required. Because of the structure chosen for $\mathbf{W}^c$ in (3.21) it is possible to assign the initial condition of the network state as the initial value of the output of the high-fidelity simulation, defining thus a physical meaning for the network:

$$\mathbf{x}_0 = \begin{pmatrix}\hat{\mathbf{y}}_0^T & \mathbf{0}^T\end{pmatrix}^T \tag{3.47}$$

where the size of the null vector is equal to $n_x - p$. Instead, the initial condition of the state Jacobian matrix is taken as the null matrix, which means that the initial network synaptic weights reside at a stationary point of $\mathbb{R}^\theta$:

$$\mathbf{\Lambda}_0 = \mathbf{0} \tag{3.48}$$

Therefore, adding (3.47) and (3.48) to (3.35), the nonlinear, dynamics problem which has to be solved at each iteration of the optimization is defined by:

$$\begin{cases}\dot{\mathbf{x}}(t) = \mathbf{W}^x\mathbf{\Phi}\left(\mathbf{W}^a\mathbf{x}(t) + \mathbf{W}^b\mathbf{u}(t) + \mathbf{b}_1\right) + \mathbf{b}_2 \\ \dot{\mathbf{\Lambda}}(t) = \hat{\mathbf{A}}\mathbf{\Lambda}(t) + \mathbf{U}(t) \\ \mathbf{x}(t=0) = \mathbf{x}_0 \\ \mathbf{\Lambda}(t=0) = \mathbf{\Lambda}_0\end{cases} \tag{3.49}$$

and the output equation, required for the computation of the error is:

$$\mathbf{y}(t) = \mathbf{W}^c\mathbf{x}(t) \tag{3.50}$$

Since only aerodynamic nonlinearities are considered here, i.e. strong shocks with large motions, the CTRNN will identify the aerodynamic response computed by a high-fidelity CFD code, such as AeroFoam. Then the CTRNN model computed will be coupled to the linear structural dynamics in order to form the ROM and compute a much faster nonlinear aeroelastic response than that required by a full order high-fidelity simulation.

The logical scheme of the network training is given in figure 3.6.

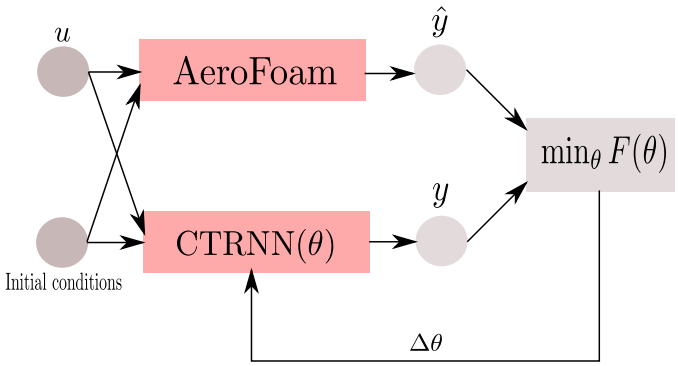


Figure 3.6: Data's flux during the training procedure

3.4 Other possible parametrizations of a CTRNN

The recurrent neural network presented in section 3.2 can be efficiently used in the reduced order modeling of a dynamics system, as shown in [27]. However, the number of variables that have to be computed, previously defined as n_θ can be quite large and this can mean that the convergence properties of the proposed method are in a some manner limited, since the subspace $\mathbb{R}^{n_\theta}$ has to be swept completely, in order to find an optimally trained solution. Consequently, the neural network described previously will be called in the following as Extended Continuous Time Recurrent Neural Network (ECTRNN).

In this section, two further neural network parametrizations are presented, and will be considered in the test case chapter. The first neural network explained is very similar to the previous ECTRNN, with only a slight modification, meanwhile the second one presents a few significant differences.

3.4.1 Combined Continuous Time Recurrent Neural Network (C^{2}TRNN)

The ECTRNN described in section 3.2 present a lot of unknowns: all of its synaptic weights and the bias vectors. In order to reduce the number of unknowns and thus the dimension of the subspace $\mathbb{R}^{n_\theta}$ where the optimization method will seek the solution, the bias vectors are eliminated from the previous formulation, and thus the following CTRNN model is obtained:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x \mathbf{\Phi} \left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u} \right) \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{3.51}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$, $\mathbf{W}^x \in \mathbb{R}^{n_x \times n_h}$, $\mathbf{\Phi} : \mathbb{R}^{n_h} \longrightarrow \mathbb{R}^{n_h}$, $\mathbf{W}^a \in \mathbb{R}^{n_h \times n_x}$, $\mathbf{W}^b \in \mathbb{R}^{n_h \times m}$ and $\mathbf{W}^c \in \mathbb{R}^{p \times n_x}$; with n_x, n_h, m and p the dimensions of the state space, hidden space, input space and output space rispectively. A classical representation of such network, encountered more frequently in the literature than the matrix formulation is given below:

$$\dot{x}_j = \sum_{i=1}^{n_h} W^x_{ji} \phi_i \left(\sum_{k=1}^{n_x} W^a_{ik} \, x_k + \sum_{q=1}^{m} W^b_{iq} \, u_q \right) \qquad j = 1, \dots, n_x \tag{3.52}$$

Its unknowns are collected in the vector θ, defined as:

$$\theta = \left(\mathrm{vec}\left(\mathbf{W}^x\right)^T, \mathrm{vec}\left(\mathbf{W}^a\right)^T, \mathrm{vec}\left(\mathbf{W}^b\right)^T \right)^T \tag{3.53}$$

In this case $n_\theta = 2\, n_h \, n_x + n_h \, m$, thus there are $n_x + n_h$ less than in the previous case. Since the output the network activation functions are combined through the matrix $\mathbf{W}^x$, this network will be called Combined Continuous Time Recurrent Neural Network (C^{2}TRNN).

A graphical description of this network is given in figure 3.7.

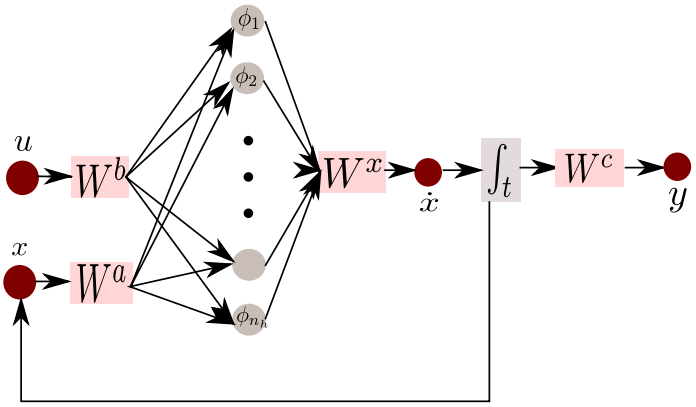


Figure 3.7: Example of C^{2}TRNN

Analytical computation of the state Jacobian matrix

Since the Jacobian matrix of a C^{2}TRNN is required by the LM method during the training, the analytical computation of the state Jacobian matrix, defined in (3.23) is presented.

As previously done, it is easy to split the matrix $\mathbf{\Lambda}$ in blocks for a faster computation of its elements:

$$\mathbf{\Lambda} = \begin{bmatrix} \dfrac{\partial \mathbf{x}}{\partial W^x_{ij}} & \dfrac{\partial \mathbf{x}}{\partial W^a_{ij}} & \dfrac{\partial \mathbf{x}}{\partial W^b_{ij}} \end{bmatrix} \tag{3.54}$$

remembering the size of each vector/matrix involved and the order with they have been sorted, then, the assembling is straightforward. Considering the shorthand notation $\mathbf{z} = \mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}$, the direct computation of the blocks of (3.54) from (3.51) reads as:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^x_{ij}} = \mathbf{I}_{ij}\mathbf{\Phi}(\mathbf{z}) + \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^x_{ij}} \qquad \begin{matrix} i = 1, \dots, n_x \\ j = 1, \dots, n_h \end{matrix} \tag{3.55}$$

with $\mathbf{\Phi}'(\mathbf{z}) = \mathrm{diag}_{n_h}\left[\phi'_1(z_1), \phi'_2(z_2), \cdots, \phi'_{n_h}(z_{n_h})\right]$, then

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^a_{ij}} = \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\left(\mathbf{I}_{ij}\mathbf{x} + \mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^a_{ij}}\right) \qquad \begin{matrix} i = 1, \dots, n_h \\ j = 1, \dots, n_x \end{matrix} \tag{3.56}$$

$$\frac{\partial \dot{\mathbf{x}}}{\partial W^b_{ij}} = \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\left(\mathbf{I}_{ij}\mathbf{u} + \mathbf{W}^a\frac{\partial \mathbf{x}}{\partial W^b_{ij}}\right) \qquad \begin{matrix} i = 1, \dots, n_h \\ j = 1, \dots, m \end{matrix} \tag{3.57}$$

assembling the different blocks (3.55), (3.56) and (3.57), let's define:

$$\mathbf{U} = \begin{bmatrix} \mathbf{I}_{ij}\mathbf{\Phi}(\mathbf{z}) & \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\mathbf{I}_{ij}\mathbf{x} & \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\mathbf{I}_{ij}\mathbf{u} \end{bmatrix} \tag{3.58}$$

and

$$\hat{\mathbf{A}} = \mathbf{W}^x\mathbf{\Phi}'(\mathbf{z})\mathbf{W}^a \tag{3.59}$$

thus the Jacobian matrix dynamics is governed by the following matrix differential equation:

$$\dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \tag{3.60}$$

At this point, the coupling of (3.51) and (3.60) defines the *nonlinear state dynamics* of a C^{2}TRNN:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x \mathbf{\Phi}\left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u}\right) \\ \dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \\ \mathbf{J} = -\mathbf{W}^c \mathbf{\Lambda} \end{cases} \tag{3.61}$$

the set of ODE is very similar to that proposed in 3.35, but in this case the number of variables involved is smaller. This fact can results in a faster convergence of the optimization method. The absence of the bias terms can be interpreted as a sort of *simmetry* in the behaviour of the network, a feature that can be wanted or unwanted in relation of the application of the CTRNN.

3.4.2 Continuous Time Recurrent Neural Network with Linear Input (CTRNN.LI)

The CTRNN presented here is not a neural network in a strict sense. It is recurrent since the network state is also a nonlinear input for the network itself, but the external input, previously called as $\mathbf{u}$ enter into the network dynamics linearly, through a simple combination of its values. This formulation recalls in some manner the one presented in [28], with slight modifications for what regard the structure of the synaptic weight matrices.

The dynamic model used for representing the CTRNN.LI is:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^a \mathbf{\Phi}\left(\mathbf{x}\right) + \mathbf{W}^b \mathbf{u} \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{3.62}$$

Thus this formulation can be interpreted as a nonlinear generalization of a linear system canonical representation: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, where in place of $\mathbf{x}$ there is the nonlinear output of an activation function $\mathbf{\Phi}\left(\mathbf{x}\right)$. However the external input always enter linearly in the system dynamics. A graphical representation of the CTRNN.LI is given in figure 3.8.

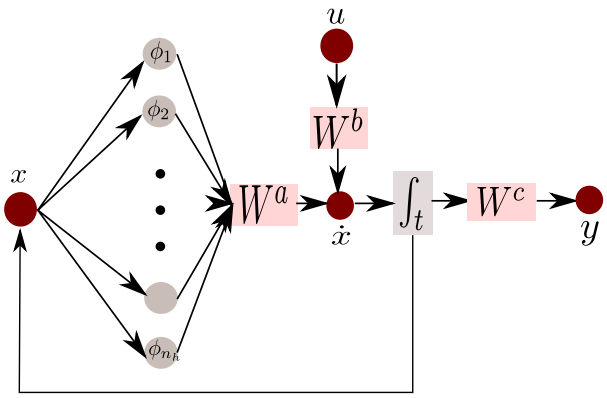


Figure 3.8: Example of CTRNN.LI

A classical representation of such network is given below:

$$\dot{x}_j = \sum_{i=1}^{n_x} W_{ji}^x \phi_i\left(x_i\right) + \sum_{q=1}^{m} W_{jq}^b u_q \qquad j = 1, \ldots, n_x \tag{3.63}$$

This formulation permits to further reduce the number of unknowns n_θ, that in this case is equal to $n_\theta = n_x\, n_x + n_x\, m$. It is a great reduction of the number of unknown from the ECTRNN model and even less unknown than the C^{2}TRNN model, since the weight matrix $\mathbf{W}^x$ is not considered here. The unknown vector θ is now defined as:

$$\theta = \left(\mathrm{vec}\left(\mathbf{W}^a\right)^T, \mathrm{vec}\left(\mathbf{W}^b\right)^T\right)^T \tag{3.64}$$

Analytical computation of the state Jacobian matrix

Since the Jacobian matrix of the CTRNN.LI is also required by the LM method during the training phase, the analytical computation of the state Jacobian matrix $\mathbf{\Lambda}$, defined in (3.23) is presented.

As for the previous formulations, it is easy to split the matrix $\mathbf{\Lambda}$ in blocks for a faster computation of its elements:

$$\mathbf{\Lambda} = \begin{bmatrix} \dfrac{\partial \mathbf{x}}{\partial W_{ij}^a} & \dfrac{\partial \mathbf{x}}{\partial W_{ij}^b} \end{bmatrix} \tag{3.65}$$

remembering the size of each vector/matrix involved and the order with which they have been sorted, the assembling is straightforward. The direct computation of the blocks of (3.65) from (3.62) reads:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W_{ij}^a} = \mathbf{I}_{ij}\mathbf{\Phi}'(\mathbf{x}) + \mathbf{W}^a\mathbf{\Phi}'(\mathbf{x})\frac{\partial \mathbf{x}}{\partial W_{ij}^a} \qquad \begin{matrix} i = 1, \dots, n_x \\ j = 1, \dots, n_x \end{matrix} \tag{3.66}$$

with $\mathbf{\Phi}'(\mathbf{x}) = \mathrm{diag}_{n_x}\left[\phi_1'(x_1), \phi_2'(x_2), \cdots, \phi_{n_x}'(x_{n_x})\right]$, then:

$$\frac{\partial \dot{\mathbf{x}}}{\partial W_{ij}^b} = \mathbf{I}_{ij}\mathbf{u} + \mathbf{W}^a\mathbf{\Phi}'(\mathbf{x})\frac{\partial \mathbf{x}}{\partial W_{ij}^b} \qquad \begin{matrix} i = 1, \dots, n_x \\ j = 1, \dots, m \end{matrix} \tag{3.67}$$

assembling the different blocks (3.66) and (3.67), let's define:

$$\mathbf{U} = \begin{bmatrix} \mathbf{I}_{ij}\mathbf{\Phi}'(\mathbf{x}) & \mathbf{I}_{ij}\mathbf{u} \end{bmatrix} \tag{3.68}$$

and

$$\hat{\mathbf{A}} = \mathbf{W}^a\mathbf{\Phi}'(\mathbf{x}) \tag{3.69}$$

thus the Jacobian matrix dynamics is governed by the following matrix differential equation:

$$\dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \tag{3.70}$$

At this point, the coupling of (3.62) and (3.70) defines the *nonlinear state dynamics* of the CTRNN.LI:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^a\mathbf{\Phi}(\mathbf{x}) + \mathbf{W}^b\mathbf{u} \\ \dot{\mathbf{\Lambda}} = \hat{\mathbf{A}}\mathbf{\Lambda} + \mathbf{U} \\ \mathbf{y} = \mathbf{W}^c\mathbf{x} \\ \mathbf{J} = -\mathbf{W}^c\mathbf{\Lambda} \end{cases} \tag{3.71}$$

Formally, this dynamic problem is very similar to the previous ones presented: (3.35) and (3.61), but in practice the expressions in the ODE are now very different. Since a smaller number of unknowns, the convergence properties of the LM method should be improved, resulting in a faster training algorithm. It will be interesting to analyze the approximation properties of the CTRNN.LI, because of the presence of the linear term involving the external input.

3.5 Training signals

The design of the training signal for the identification of a nonlinear system is a non-trivial task in view of the invalidation of the superposition principle widely used in linear system analysis. Such a design requires not only the definition of the signal frequency content, but also the amplitude and the phase shift between the input must be carefully chosen for the case of Multi Input Multi Output (MIMO) systems, as the one under investigation.

For black box modeling the training signal is the most important source of information. If the signal does not excite all the frequencies and amplitudes of interest, the resulting ROM will not be characterized by the same behaviour that determines the response of the full order model. Thus the design of the training signal is a key step in the construction of the neural network based ROM. Unfortunately, very few tools exist and little research is devoted in this subject. This can be due to the high dependence of their design from the particular physical behaviour of the system under consideration [26]. Independently of the chosen model architecture and structure, the quality of the identification signal determines an upper bound on the best accuracy that can be achieved. If for linear systems the training signal can be designed quite automatically, each nonlinear system requires an individual design. Recalling partially the hints given in [26], the main issues that influence the design of the excitation signal are:

Purpose of modeling The model operational states of interest should be excited by the input signal. Thus if the steady state behaviour of the system is required by the model, the training signal must contain some parts with constant values. Instead, if only the unsteady behaviour is of interest, e.g. in stability analysis, the input signal can be completely unsteady, without containing constant parts;

Lenght of the training data set Since this work is related to computer experiments, the only limit in information recording is given only by the computer memory. The lenght of the training data should be selected as a compromise between a long list of training samples, which guarantees a more precise model if there is a reasonable data distribution, and the computational time, which is clearly shorter if the training signal is composed by a few samples;

Range of the input signal The full order system should be driven through all the operational regimes that might occour in real operations. Unrealistic operation conditions need not to be considered. Moreover, it is important that the training data covers the limits of the input range because model extrapolation is much more inaccurate than interpolation.

Various tipologies of input signals have been considered:

- Series of steps;
- Series of ramps;
- Sinusoidal input with randomly varying amplitude and frequency, here called *hybrid* signal;
- Random signal.

Since the range of frequencies and amplitudes of the phenomena under interest can be known a priori, i.e. LCO amplitude and frequencies, the input signals can be designed ad hoc.

In aeroelasticity, a widely used variable is the *reduced frequency* k, which is an adimensional frequency, defined as:

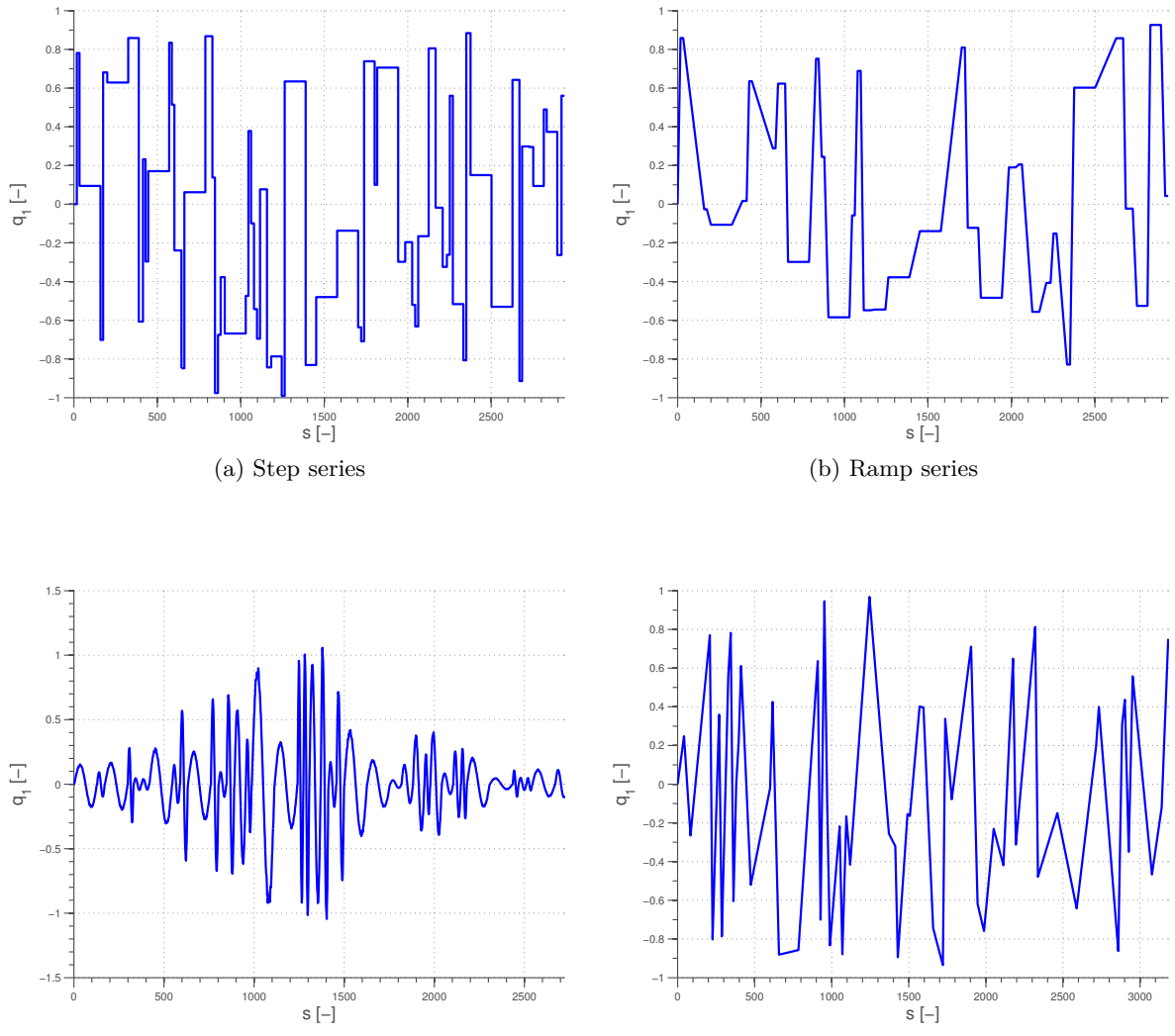
$$k = \frac{\omega l_a}{V_\infty} \tag{3.72}$$

where ω is the physical circular frequency, measured in $[rad\, s^{-1}]$, l_a is the reference aerodynamic lenght, which can be equal to the wing chord c for symmetric problems, or to the wing span b for asymmetric problems, and finally V_∞ is the flow velocity.

For sinusoidal inputs, the signal frequency can be directly imposed, meanwhile for the other signal tipologies a different approach has been considered. Since the reduced frequency is defined as in (3.72), this definition can be exploited to compute the time period T_P of the signal required to excite a particular reduced frequency k:

$$T_P = \frac{\frac{2\pi l_a}{V_\infty}}{k} = \frac{C_\tau}{k} \tag{3.73}$$

thus, specifying the range of reduced frequency that has to be excited, the training signal can be efficiently created with the correct time steps. An example of the signals proposed is given in figures 3.9a, 3.9b, 3.8c and 3.8d.



(a) Step series

(b) Ramp series

(c) Hybrid

(d) Random

Figure 3.8: Samples of the training signals proposed

Where s is the adimensional time, defined as $s = \frac{2\,V_\infty\,t}{l_a}$. The signals with steps and ramps can be useful when the steady state value of the response are of interest, for example in a simulation where the system is excited. Since this work considers only stability issues, i.e. LCO, these signal are not considered in the next analyses. The time lenght of the flat part of the step signal and both the flat and linearly variable parts of the ramp signal are determined by applying (3.73).

In the case in which the steady state response is of interest, no error should be present when the transient part is over. Such a result can be obtained weighting the error defined by 3.36 where the training target becomes constant, therefore obtaining a weighted nonlinear least square problem 3.40, solved by the LM algorithm. The weighted part of the error should be minimized with greater efficiency, as can be seen in [26].

The hybrid signal is generated by chosing its amplitude and frequency randomly for each local period. Which means that once the frequency has been chosen from a set of specified frequencies, it is mantained constant for one period. Then the frequency is chosen again randomly and the process is continued. Also the amplitude of the signal is chosen randomly, but in this case it is also modulated by a gaussian function centered at the mean time of the simulation horizon T_S, $\mu = \dfrac{T_S}{2}$ and with a standard deviation equal to $\sigma = \dfrac{T_S}{6}$. Since the signal is deterministic, but from period to period both frequency and amplitude are varied randomly, this signal has been called hybrid. In this manner all the frequency of interest should be excited more than once, also with a varying amplitude that also vary. The occurrencies of a particular frequency during the generation of the signal are stored for a final check, with an example shown in figure 3.9.

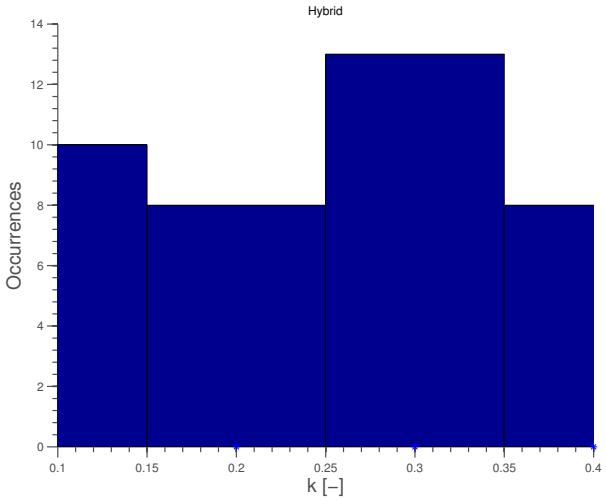


Figure 3.9: Check of the hybrid signal frequency content

Finally, the random input is generated taking random signal amplitudes and frequencies from a priori defined ranges of interest. The time steps lenght is chosen considering (3.73) after the reduced frequency has been chosen randomly. The amplitude are also chosen randomly, but in this case it is not moduled by a gaussian function. The random input is thus very similar to the hybrid one, since both amplitude and characteristic frequencies are varied in a random manner; the only difference is related to the *regularity* of the signal, meanwhile the random signal presents different thin peaks with a corresponding discontinuous derivative, the hybrid signal is smoother, with possible derivatives discontinuity only at when passing from a period to an another. Hovewer both signal can be considered in the study of a completely unsteady flow, which is of course of interest in a stability analysis.

The characteristic frequencies of the signal can be checked along with the generated signal, as shown in figure 3.10.

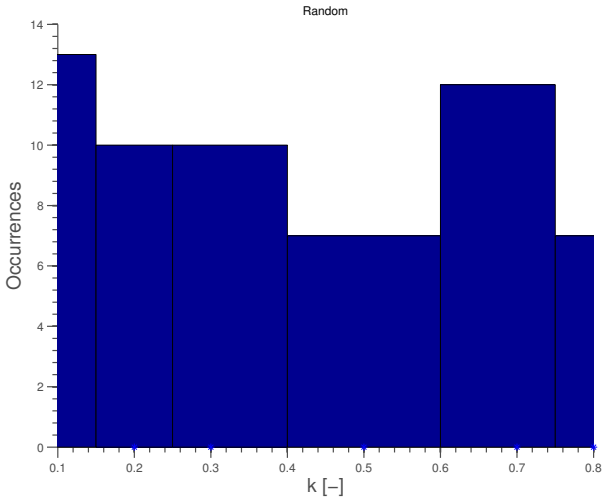


Figure 3.10: Check of the random signal frequency content

In this example the reduced frequency range of interest is a series of discrete elements in the range $k \in [0.1, 0.8]$, the number of occurrencies being shown on the ordinate axis.

Thus, signals like 3.8c or 3.8d can be used as training signal of a network that will then be applied to reproduce highly unsteady responses, such as the undamped oscillations which characterizes an LCO. The generalization properties of the network after such training will be analyzed in chapters 5 and 6.

Chapter 4

Automatic differentiation

High-speed computers and sophisticated software tools has made the derivative evaluation of functions defined in computer programs easier to understand and implement, permitting a significant reduction of the computational time, along with the increasing automaticity of the process. On one hand, the dependence of certain program output on certain input parameters can now be determined and quantified more or less automatically, if the function is known analitically. On the other hand, such qualitative and quantitative dependence analysis is invaluable for the optimization of key output objectives with respect to suitable decision variables or the identification of model parameters with respect to given data. During the optimization procedure, a lot of function and related sensitivities (or gradients) must be computed, thus the derivative computation should be both fast and accurate. Derivatives that can be gradients, Jacobian or Hessian matrices, depending on the optimization algorithm used [64].

Derivatives of mathematical function can be obtained by the following four approaches:

- Analytic differentiation
- Finite differentiation;
- Symbolic differentiation;
- Automatic differentiation.

Of course analytic differentiation is the first approach to relatively simple differentiation problems. If the function to be differentiated depends on a small number of variables, the computation can be performed at once manually. However, if the function is composed by a series of complex functions, i.e. trascendental, highly nonlinear, composed, or if the set of dependent/independent variables is large, then analytic differentiation may be almost impossible and subjected to errors difficult to debug. The other three approaches fall into the field of the computational differentiation, and will be analyzed in the following sections.

The present chapter discusses all the possible approaches to the evaluation of a function derivatives with particular emphasis on an advanced techniques available today for such operations: i.e. Automatic Differentiation (AD). The finite differentiation approach is presented in section 4.1, discussing its applications, strengths and weaknesses in the approximation of a function derivative. In section 4.2 the symbolic differentiation technique is briefly introduced. Section 4.3 presents the different algorithms used by the AD techniques, as the forward and the reverse mode, introduced by simple scalar examples. Then in section 4.3.1 the AD techniques presented are applied to compute the Taylor series expansion elements of a generic continuous function using both the forward mode (for vectors) and the reverse mode (for matrices). Section 4.3.2 presents an explicit method for integrating a system of ODEs based on the Taylor series expansion of the solution. Some examples of the integration technique proposed are given in section 4.3.3,

accompanied by a description of the converging properties of such method and a tuning of the different parameters involved in the algorithms. Finally an application of the AD techniques proposed is implemented in section 4.4 in order to reduce the time required for the training of the CTRNN.

4.1 Finite differentiation

The most common method used to evaluate the derivatives of a function defined inside a computer code is the finite difference method. It applies *small numerical perturbation* to the defined function, computing an approximate value of the derivative, which accuracy depends on the size of the perturbation. The concept of finite differentiation can be applied in both mono-dimensional and multi-dimensional space, for example, assuming a continuous univariate function $f(x) : \mathbb{R} \longrightarrow \mathbb{R}$, its derivative, evaluated at the point x_0, can be approximated as:

$$f'(x)\big|_{x_0} \approx \frac{f(x_0 + h) - f(x_0)}{h} \tag{4.1}$$

with h a small numerical perturbation to the value x_0. This approximation is called forward finite difference. Greater accuracy can be obtained by a centered finite difference:

$$f'(x)\big|_{x_0} \approx \frac{f(x_0 + h) - f(x_0 - h)}{2\,h} \tag{4.2}$$

Forward (or backward) finite differences guarantee first order accuracy, meanwhile central finite differences produce a second order approximation. Their accuracy can be evaluated truncating the Taylor series of $f(x)$ at the first and second order respectively [65].

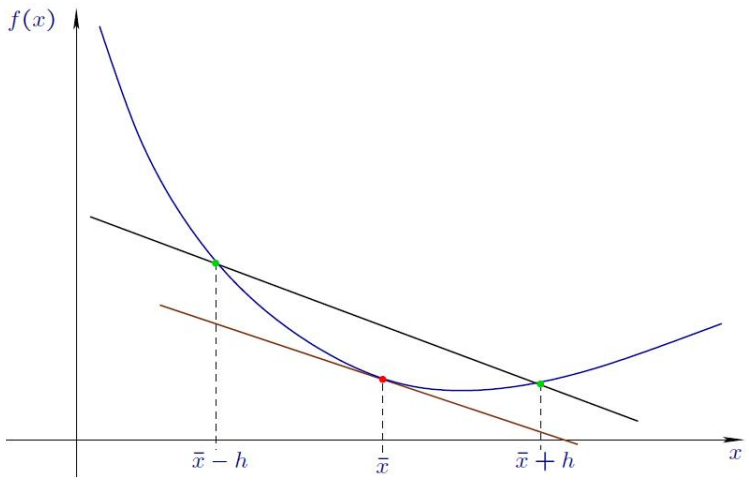


Figure 4.1: Finite difference approximation

For a vector-valued function $\mathbf{F}(\mathbf{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, its derivative with respect to the set of independent variables $\mathbf{x}$ is called the *Jacobian matrix* $\mathbf{J}(\mathbf{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m \times \mathbb{R}^n$, and it can be approximated by finite differences and evaluated at the point $\mathbf{x}_0$ as follow:

$$J_{i,j}(\mathbf{x}_0) = \left.\frac{\partial F_i(\mathbf{x})}{\partial x_j}\right|_{\mathbf{x}_o} \approx \frac{F_i(\mathbf{x}_0 + h\,\mathbf{e}_j) - F_i(\mathbf{x}_0)}{h} \tag{4.3}$$

with $\mathbf{e}_j$ the single entry vector. Of course the central formula can be applied in this case also.

The main advantage of the finite differentiation is that only function evaluations are needed to compute the derivatives, thus its implementation is straightforward.

Disadvantages are also present, and resides on the tradeoff between truncation and round-off error. Since the Taylor series approximation, from which the above formulas are derived, is valid only near the evaluation point, a small perturbation value tends to reduce the truncation error. Theoretically, with a perturbation that tends to zero, the approximated value tends to the exact value, but since the reality is not just the theory, the round-off error becomes predominant. The best value of the perturbation can be found testing the finite difference formula with different values of perturbation, but this can be very costly from the computational point of view and even when the optimal value of the perturbation is found, the resulting derivative value remains an approximation. A procedure for the automatic estimation of the optimal differentiation interval can be found in [66], which is based also in the computation of the second derivative of the function, since its absolute value modules the error of the first order approximation. Moreover, it should be noted that meanwhile a forward or backward approximation needs only one additional function evaluation, a central difference approximation requires two additional function evaluations. Therefore, if the accuracy of the forward/backward difference is not jeopardized, i.e. near the minimum point of an optimization algorithm, this choice should be preferred to the adoption of a central difference scheme. Finite differentiation remains however a technique followed in practical applications, since its good approximation of low-order derivative terms. Nonetheless, the associated computation cost is often too high, since the number of operations required for evaluating the derivative of a function which depends on N variables is approximately equal to $N + 1$ times the number of operation required to evaluate the function itself.

4.2 Symbolic differentiation

Symbolic differentiation is a computer aided analog to analytical or hand differentiation employing a graph theoretical approach. The formula of a function is transformed automatically into a formula of its derivative, which can be evaluated or further transformed by a computer program. Symbolic differentiation is usually performed by interpreted languages like *Maple* or *Mathematica*. However this approach has shown to be unable to deal with loops and subroutines present in computer codes [27].

```
>
> y1 := nu * tan(omega * t)/(gamma - tan(omega * t));
```

$$y1 := \frac{\nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)}$$

```
> y2 := gamma * nu * tan(omega * t)/(gamma - tan(omega*t));
```

$$y2 := \frac{\gamma\, \nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)}$$

```
> diff(y2,nu);
```

$$\frac{\gamma \tan(\omega\, t)}{\gamma - \tan(\omega\, t)}$$

Figure 4.2: Example of symbolic differentiation

It should be noted that this approach can be followed for low-order derivatives computation, since for each transformation the expression of the formula is likely to double the size, leading the the impossibility of evaluating the computational cost of the operations required, growing with the increasing complexity of the function considered. Due to these limitations, the symbolic differentiation approach is seldom used in practical applications [27].

4.3 Automatic differentiation (AD)

In practical engineering applications, the analyst has at hand computer programs that compute numerical values of some functions. Typically, it is desired to obtain occurate values for the function derivative as well, since they play a central role especially in model *validation* (sensitivity analysis) and model *optimization*. Automatic differentiation (AD) has been developed in order to compute automatically the derivatives of such computer functions. The algorithms proposed in [64] act on user-defined functions, generating a transformed program that can compute various derivatives with great accuracy and efficiency. Like finite differences, AD requires only the evaluation of the function, but in this case the derivative is not approximated, but the result is the exact, analytical derivative, free of truncation errors, but nevertheless affected by round-off error.

The efficiency of AD derives on its *algorthmic* nature: instead of computing the function derivatives considering different sets of inputs, a new program is created, which computes the exact derivatives along with the original computer program. Thus, each time the function has the value f, the *dual* or *differentiated* program holds the differentiated value $\mathrm{d}f$. Moreover, when the original program executes some mathematical operations, the differentiated program performs dual operations on the differentiated values.

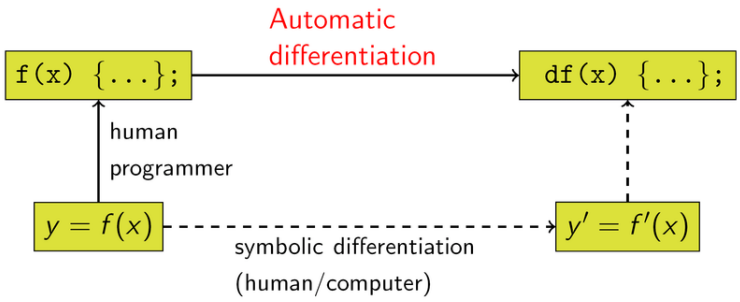


Figure 4.3: Automatic differentiation data flow

Like symbolic differentiation, AD operates by the systematic application of the *chain rule*, which is not applied to the symbolic expression of the function, but to its actual numerical values. The efficiency of these algorithms resides on the an *evaluation trace* set up during the evaluation of the function. This evaluation trace is basically a record of a particular run of a given computer program, with a specified value of the input variables, showing the sequence of the operations performed. An example can made the concept clear: consider the function

$$f(x_1, x_2) = x_1 + x_2 + \sin(x_1\, x_2) \tag{4.4}$$

f will be called the *dependent* variable, meanwhile the variables x_i will be referred as *independent* variables. The partial derivatives of f are easily computed as:

$$\frac{\partial f}{\partial x_1} = 1 + \cos(x_1\, x_2)\, x_2 \qquad \frac{\partial f}{\partial x_2} = 1 + \cos(x_1\, x_2)\, x_1 \tag{4.5}$$

AD introduce an algorithmic approach to the computation of the derivatives of the function $f(x_1, x_2)$:

$f(x_1, x_2)$	$\mathrm{d}f(x_1, x_2)$
$v_{-1} = x_1$	$\mathrm{d}v_{-1} = \mathrm{d}x_1$
$v_0 = x_2$	$\mathrm{d}v_0 = \mathrm{d}x_2$
$v_1 = v_{-1}$	$\mathrm{d}v_1 = \mathrm{d}v_{-1}$
$v_2 = v_0$	$\mathrm{d}v_2 = \mathrm{d}v_0$
$v_3 = v_1\, v_2$	$\mathrm{d}v_3 = \mathrm{d}v_1\, v_2 + v_1\, \mathrm{d}v_2$
$v_4 = \sin(v_3)$	$\mathrm{d}v_4 = \cos(v_3)\mathrm{d}v_3$
$v_5 = f(x_1, x_2) = v_1 + v_2 + v_4$	$\mathrm{d}v_5 = \mathrm{d}f(x_1, x_2) = \mathrm{d}v_1 + \mathrm{d}v_2 + \mathrm{d}v_4$

Table 4.1: Evaluation of f and its diffentiated value $\mathrm{d}f$

at this point, for the evaluation of the partial derivatives it is sufficient to set $\mathrm{d}v_{-1} = 1$ and $\mathrm{d}v_0 = 0$ if the value of $\dfrac{\partial f}{\partial x_1}$ is desired, or set $\mathrm{d}v_{-1} = 0$ and $\mathrm{d}v_0 = 1$ if the other derivative is requested:

$\dfrac{\partial f}{\partial x_1}$	$\dfrac{\partial f}{\partial x_2}$
$\mathrm{d}v_{-1} = 1$	$\mathrm{d}v_{-1} = 0$
$\mathrm{d}v_0 = 0$	$\mathrm{d}v_0 = 1$
$\mathrm{d}v_1 = 1$	$\mathrm{d}v_1 = 0$
$\mathrm{d}v_2 = 0$	$\mathrm{d}v_2 = 1$
$\mathrm{d}v_3 = v_2$	$\mathrm{d}v_3 = v_1$
$\mathrm{d}v_4 = \cos(v_3)\, v_2$	$\mathrm{d}v_4 = \cos(v_3)\, v_1$
$\mathrm{d}v_5 = \dfrac{\partial f}{\partial x_1} = 1 + \cos(v_3)\, v_2 = 1 + \cos(x_1\, x_2)\, x_2$	$\mathrm{d}v_5 = \dfrac{\partial f}{\partial x_2} = 1 + \cos(v_3)\, v_1 = 1 + \cos(x_1\, x_2)\, x_1$

Table 4.2: Evaluation of the partial derivatives of f with AD

In a computer algorithm, this list defines a sequence of mathematical functions as well as a sequence of numerical values. It is now clear that the evaluation trace does not correspond directly to the expression of f as it was originally written, but to a more efficient evaluation with repeated subexpressions evaluated once and then re-used.

The example above shows one of the two main basic algorithms used by AD in order to compute derivatives: the *forward* mode. It is called in this way since the derivative values $\mathrm{d}v_i$ are carried along simultaneously with the values v_i themselves. Thus the derivatives are propagated through the computation using the chain rule, and the function and its derivatives can be evaluated in parallel. It is the easiest mode to implement, and its computational effort is proportional to the product of the number of independent variables and the number of dependent variables involved in the problem. An alternative to the forward mode is represented by the *reverse* or *adjoint* mode. This mode follows basically the inverse sense of the derivatives propagation: rather than choosing an input variable and computing the sensitivity of every intermediate variable with respect of that input, an output variable is chosen and its sensitivities are computed with respect to each of the intermediate variables. Hence adjoint sensitivities are naturally computed backward, starting from the output variables. The reverse mode can be interpreted as the generalization of the back propagation algorithm used in the training of neural networks. Considering the example presented above, to each variable v_i is associated the

adjoint variable $\hat{v}_i = \dfrac{\partial f}{\partial v_i}$, and also in this case, the evaluation procedure can be mechanically transformed to provide a procedure for the computation of the adjoint terms.

$$\hat{f} = 1$$

$$\hat{v}_5 = 1$$
$$\hat{v}_4 = 1$$
$$\hat{v}_3 = \hat{v}_4 \cos(v_3) = \cos(v_3)$$
$$\hat{v}_2 = 1 + \hat{v}_4 \hat{v}_3 v_1$$
$$\hat{v}_1 = 1 + \hat{v}_4 \hat{v}_3 v_2$$
$$\hat{v}_0 = \hat{v}_2 = \frac{\partial f}{\partial x_2} = 1 + \cos(x_1 x_2) x_1$$

$$\hat{v}_{-1} = \hat{v}_1 = \frac{\partial f}{\partial x_1} = 1 + \cos(x_1 x_2) x_2$$

Table 4.3: Computation of the partial derivatives of f through reverse mode of AD

It is clear that while with the forward mode the same path has been followed for a number equal to the number of partial derivative desired, the reverse mode permits to compute all the partial derivative of a function at once, following the inverse path one time only. This fact may be exploited in different applications, as will be explained later.

Therefore, with the forward and reverse procedures explained above, truncation-error-free numerical values of derivatives can be obtained, rather than the corresponding symbolic expressions. Of course, since the chain rule will be applied to floating point numbers causes some round-off error but this makes the memory and the run-time bounded a priori, as can be seen from tables (4.1), (4.2) and (4.3). Moreover, as may be seen by the simple examples presented, the parallel evaluation of the function and its differentiated value comports an effort which is always a small multiple of the one required for the evaluation of the function only.

The formulation of the forward and reverse mode can be reformulated easily for a multidimensional problem. The values of the derivative vectors will depend on new defined quantities, as the *seed directions* x_t and the *weight functionals* $\hat{z}$, rather than representing the derivatives of a particular dependent variable z_i with respect to a particular dependent variable x_j. Assuming a differentiable function $\mathbf{z} = \mathbf{f}(\mathbf{x})$, $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ declared in a computer program, and assuming also that the trajectory of $\mathbf{x}$ is parametrized on a set of variables $\mathbf{t}$, such that $\mathbf{x} = \mathbf{x}(\mathbf{t})$; the derivative of $\mathbf{z}$ with respect to such variable can be computed with the forward mode:

$$\mathbf{z_t} = \mathbf{f}'(\mathbf{x})\mathbf{x_t} \tag{4.6}$$

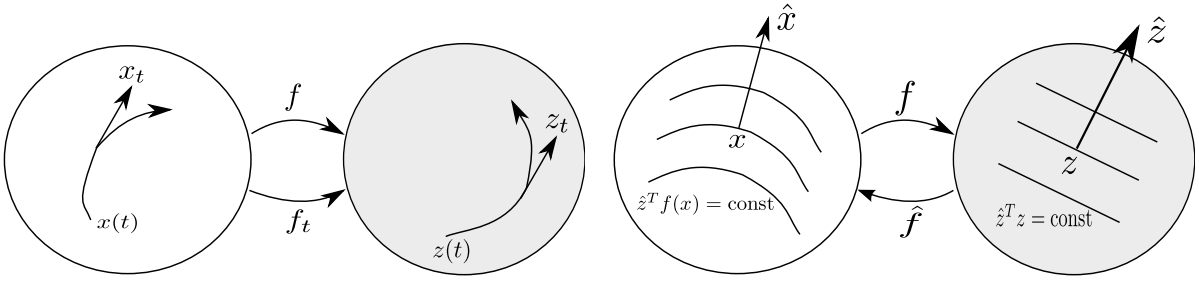
given the direction $\mathbf{x_t}$, the directional derivative of $\mathbf{f}$ along $\mathbf{x_t}$ is computed. The corresponding *tangent value* $\mathbf{z_t}$ is obtained calculating the tangent values of all the intermediate values occurring during the function evaluation. Looking at (4.6), the most natural computation of $\mathbf{z_t}$ seems to be the direct computation of the product between the Jacobian matrix and the seed direction, which means that the two terms are computed separately. However, this approach is quite expensive, except when computing $\mathbf{z_t}$ for very many directions $\mathbf{x_t}$ at some fixed point $\mathbf{x}$ [64]. Instead, a parallel evaluation of the two terms, with exchange of the partial results obtained along the evaluation trace make the forward mode much more efficient [67]. Hovewer, it can be proved that the cost for evaluating such a matrix product grows linearly in the number of columns of $\mathbf{x_t}$ [64]. It is now clear what *forward mode* means: the program flow of function evaluation and the derivative computation have the same direction.

It has been seen that the cost of the forward mode grows linearly with the number of domain directions $\mathbf{x_t}$, represented by the size of the $\mathbf{t}$ element. This may seem a natural or even inevitable cost in the derivative evaluation. However, AD can benefit from a small number of dependent

variables $\mathbf{z}$ just as much as from a small number of independent variables $\mathbf{x}$. In the reverse mode, the function is differentiated in reverse order, from the end to the beginning of the computer program. This can be done introducing the definition of *adjoint variables*, $\hat{\mathbf{x}}$ and the *weight functional* $\hat{\mathbf{z}}$, which are related by:

$$\hat{\mathbf{x}}^T = \hat{\mathbf{z}}^T \mathbf{f}'(\mathbf{x}) \tag{4.7}$$

thus it can be seen as a weighted derivative, where the constant vector $\hat{\mathbf{z}}$ plays a dual role to the domain direction $\mathbf{x_t}$ considered in the forward mode. Geometrically the adjoint quantities are the normal vectors in the space in which the function belongs, as shown in figure 4.4b. If in the forward mode $\mathbf{z_t} = \mathbf{f}'(\mathbf{x})\ \mathbf{x_t} = \mathbf{f_t}(\mathbf{x}, \mathbf{x_t})$ has been evaluated, in this case $\hat{\mathbf{x}}^T = \hat{\mathbf{z}}^T \mathbf{f}'(\mathbf{x}) = \hat{\mathbf{f}}(\mathbf{x}, \hat{\mathbf{z}})$ is the relation obtained. The latter relation can be obtained considering a nonlinear transformation (a composition of functions) which maps $\mathbf{x}$ to the subspace at which behold $\mathbf{z}$, permitting also to satisfy the identity $\hat{\mathbf{z}}^T \mathbf{z_t} = \hat{\mathbf{x}}^T \mathbf{x_t}$ [64]. A graphical summary of the geometrical interpretation of forward and the reverse mode is given in figure 4.4, where is enhanced the different work directions of the forward and reverse operators, $\mathbf{f_t}$ and $\hat{\mathbf{f}}$ respectively.



(a) Mapping of the tangent x_t into the tangent z_t by f_t (b) Mapping of the normal $\hat{x}$ into the normal $\hat{z}$ by $\hat{f}$

Figure 4.4: Forward and reverse mapping

Although the derivative computation by means of the reverse mode can appear a little bit clumsy, it has be proved that in many application, where the number of dependent variables is several orders of magnitude smaller than the number of independent variables, e.g. system optimization based on a scalar cost function, that the reverse method does indeed provide a complete and accurate evaluation of the sensitivities with a computational cost proportional on the number of the dependent variables, thus well bounded a priori. Of course the reverse mode require the record of the entire evaluation trace, since the derivative propagation is done backward, resulting in a greater memory occupation. Moreover, the reverse mode implementation is quite more complex than the forward one as can be seen in [64].

4.3.1 Taylor series expansion using AD

Consider a $d-$time continuously differentiable function $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, with the independent variable path $\mathbf{x}(t)$ governed by the scalar real variable t. If the function $\mathbf{x}(t)$ is assumed to be a composition of a sequence of continuously differentiable functions, this path can be expressed in terms of the Taylor series expansion around the origin:

$$\mathbf{x}(t) = \mathbf{x}(0) + \left.\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}\right|_{t=0} t + \frac{1}{2} \left.\frac{\mathrm{d}^2\mathbf{x}(t)}{\mathrm{d}t^2}\right|_{t=0} t^2 + \cdots + \frac{1}{d!}\frac{\mathrm{d}^d\mathbf{x}(t)}{\mathrm{d}t^d} + \mathrm{o}\left(t^{d+1}\right) \tag{4.8}$$

or in a more compact form:

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{x}_1\, t + \mathbf{x}_2\, t^2 + \cdots + \mathbf{x}_d\, t^d + \mathrm{o}\left(t^{d+1}\right) \tag{4.9}$$

with:

$$\mathbf{x}_i = \frac{1}{i!} \left. \frac{\mathrm{d}^i \mathbf{x}(t)}{\mathrm{d}t^i} \right|_{t=0} \tag{4.10}$$

the element $\mathbf{x}_0$ being simply the value of $\mathbf{x}(t)$ at the origin, meanwhile the elements $\mathbf{x}_1$ and $\mathbf{x}_2$ can be interpreted as the tangent and the curvature at the base point $\mathbf{x}_0$.

Consequently, also the function $\mathbf{z}(t) = \mathbf{f}(\mathbf{x}(t))$ can be expressed in Taylor series:

$$\mathbf{z}(t) = \sum_{i=1}^{d} \mathbf{z}_i \, t^d + \mathrm{o}\left(t^{d+1}\right) \tag{4.11}$$

where the coefficients $\mathbf{z}_i$ have the same meaning as before:

$$\mathbf{z}_i = \frac{1}{i!} \left. \frac{\mathrm{d}^i \mathbf{z}(t)}{\mathrm{d}t^i} \right|_{t=0} \tag{4.12}$$

these coefficients can be computed analytically, once the function $\mathbf{f}$ is known, simply applying the chain rule:

$$\begin{aligned} &\mathbf{z}_0 = \mathbf{f}(\mathbf{x}_0) \qquad \mathbf{z}_1 = \mathbf{f}'(\mathbf{x}_0)\mathbf{x}_1 \\ &\mathbf{z}_2 = \mathbf{f}'(\mathbf{x}_0)\mathbf{x}_2 + \frac{1}{2}\mathbf{f}''(\mathbf{x}_0)\mathbf{x}_1\mathbf{x}_1 \\ &\mathbf{z}_3 = \mathbf{f}'(\mathbf{x}_0)\mathbf{x}_3 + \mathbf{f}''(\mathbf{x}_0)\mathbf{x}_1\mathbf{x}_2 + \frac{1}{6}\mathbf{f}'''(\mathbf{x}_0)\mathbf{x}_1\mathbf{x}_1\mathbf{x}_1 \end{aligned} \tag{4.13}$$

The number of terms that occur in these symbolic expressions for the $\mathbf{z}_j$ elements in terms of the first i derivatives of $\mathbf{f}$ and the input coefficients $\mathbf{x}_i$ with $i \leq j$ grows very rapidly with j. Fortunately, this exponential growth does not occur in automatic differentiation, where the many terms are somehow implicitly combined so that storage and operations count grow only quadratically with bound d on the order j.

Assumed $\mathbf{f}$ as analytic, this property is inherited by the functions:

$$\mathbf{z}_j = \mathbf{z}_j \left(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_j\right) \tag{4.14}$$

which are also $d-$continuously differentiable and their derivatives satisfy the identity [64]:

$$\frac{\partial \mathbf{z}_j}{\partial \mathbf{x}_i} = \frac{\partial \mathbf{z}_{j-i}}{\partial \mathbf{x}_0} = \mathbf{A}_{j-i} = \mathbf{A}_{j-i} \left(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{j-i}\right) \tag{4.15}$$

where $\mathbf{A}_j \in \mathbb{R}^{n \times n}$ are the Taylor coefficients of the Jacobian path:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \mathbf{A}_0 + \mathbf{A}_1 \, t + \mathbf{A}_2 \, t^2 + \cdots + \mathbf{A}_d \, t^d + \mathrm{o}\left(t^{d+1}\right) \tag{4.16}$$

Through AD techniques, the elements $\mathbf{z}_i$ and $\mathbf{A}_i$, with $i = 0, ..., d$ can be efficiently computed, the former with the use of the forward mode, meanwhile the latter can be obtained with the reverse mode. The run time and memory requirement associated with these calculations grow only as d^2 [27, 64].

4.3.2 Solution of ODE systems with AD

AD techniques can be exploited also in solving a system of ODEs [64], consider for example the following initial value problem:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{4.17}$$

defining again $\mathbf{z}(t) = \mathbf{f}(\mathbf{x}(t))$, the Taylor coefficients of the solution can be computed by the following iterative formula:

$$\mathbf{x}_{i+1} = \frac{\mathbf{z}_i}{i+1} \tag{4.18}$$

the above formula derives from the application of the forward mode to (4.17). The usual procedure is to calculate $\mathbf{z}_0 = \mathbf{x}_1$ from $\mathbf{x}_0$, then to plug in $\mathbf{x}_0 + \mathbf{x}_1 t$ in $\mathbf{z}(t)$ in order 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 each time [64].

However, like all explicit numerical integrators, for stability reasons, Taylor expansion methods are forced to make the stepsize excessively small, wherever the system is stiff. However, there are *A-stable* implicit variants that overcome this limitation [67]. It will be seen that this fact will not be a limitation in this work.

Along with the solution of the ODE, 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:

$$\begin{cases} \dot{\mathbf{B}} = \mathbf{f}'(\mathbf{x})\mathbf{B} \\ \mathbf{B}(0) = \mathbf{I} \end{cases} \tag{4.19}$$

with $\mathbf{I}$ the identity matrix. Of course the Jacobian matrix dynamics must be integrated along with the state dynamics itself, because of its implicit dependence on $\mathbf{x}$. The expression of the Taylor elements of $\mathbf{B}$ can be obtained applying the chain rule to (4.18):

$$\mathbf{B}_{i+1} = \frac{\mathrm{d}\mathbf{x}_{i+1}}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\frac{\mathrm{d}\mathbf{z}_i}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\sum_{j=0}^{i}\frac{\partial \mathbf{z}_i}{\partial \mathbf{x}_j}\frac{\mathrm{d}\mathbf{x}_j}{\mathrm{d}\mathbf{x}_0} = \frac{1}{i+1}\sum_{j=0}^{i}\mathbf{A}_{i-j}\mathbf{B}_j \tag{4.20}$$

Practically, a more straightforward approach can be used: defining a normalized time $\tau = t/\Delta t$, where Δt is the integration step used, the ODE system (4.17) results transformed in:

$$\frac{\mathrm{d}}{\mathrm{d}\tau} = \frac{1}{\Delta t}\frac{\mathrm{d}}{\mathrm{d}t} \rightarrow \frac{\mathrm{d}\hat{\mathbf{x}}(\tau)}{\mathrm{d}\tau} = \hat{\mathbf{x}}' = \Delta t\, \mathbf{f}(\hat{\mathbf{x}}(\tau)) \tag{4.21}$$

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

$$\hat{\mathbf{x}}(\tau) = \hat{\mathbf{x}}_0 + \hat{\mathbf{x}}_1\tau + \hat{\mathbf{x}}_2\tau^2 + \cdots + \hat{\mathbf{x}}_d\tau^d + \mathrm{o}\left(\tau^{d+1}\right) \tag{4.22}$$

$$\mathbf{A}(\tau) = \mathbf{A}_0 + \mathbf{A}_1\tau + \mathbf{A}_2\tau^2 + \cdots + \mathbf{A}_d\tau^d + \mathrm{o}\left(\tau^{d+1}\right) \tag{4.23}$$

meanwhile the Taylor series elements of the Jacobian matrix are assembled by (4.20). Starting from the initial value $\mathbf{x}_0$ for the state and $\mathbf{B}_0$ for the Jacobian matrix, the solution at each time step $t_{i+1} = t_i + \Delta t$ is computed explicitily as follows:

$$\mathbf{x}(t_i + \Delta t) = \sum_{j=0}^{d}\hat{\mathbf{x}}_j \tag{4.24}$$

and

$$\mathbf{B}(t_i + \Delta t) = \sum_{j=0}^{d}\mathbf{B}_j \tag{4.25}$$

with the maximum Taylor series degree d that can be adjusted in order to obtain an accurate solution of (4.17).

4.3.3 Some simple examples of the AD integrator

In this section some simple examples will be considered, exploting the concept presented previosly, and describing the AD package adopted for performing these simulations.

Unforced linear, dynamic system

A second order dynamic system, composed by a mass, a spring and a damper, represented in figure 4.5, respond to a given initial condition.

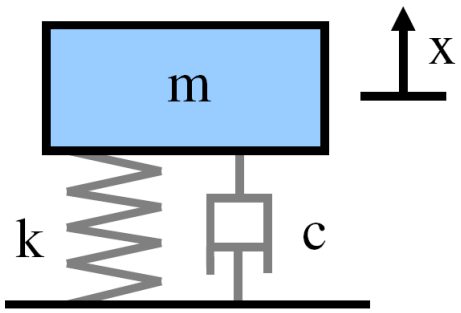


Figure 4.5: Second order dynamic system under consideration

The system dynamic is governed by the well known dynamic equilibrium:

$$m\,\ddot{x} + c\,\dot{x} + k\,x = 0 \tag{4.26}$$

with the following initial data, (4.26) constitutes an initial value problem:

$$\begin{cases} x(t=0) = 0.1 \\ \dot{x}(t=0) = 0 \end{cases} \tag{4.27}$$

The system is characterized by the following parameters:

$m = 1\,[kg]$
$c = 2\,[N\,s\,m^{-1}]$
$k = 100\,[N\,m^{-1}]$

Table 4.4: Parameters that characterize the dynamic system

The time horizon considered is $T_h = 2\,[s]$, with an integration step $\Delta t = 0.02\,[s]$, chosen as about 30 times smaller than the period of the dynamic system considered.

Starting from the given initial condition, the system (4.26) is integrated forward in time, using the approach given in sections 4.3.1 and 4.3.2.

The Taylor series elements needed in (4.18) are computed through the package ADOL-C (**A**utomatic **D**ifferentiation by **O**ver**L**oading in **C**++) [67], that facilitates the evaluation of first and higher order derivatives of vector-valued functions defined by computer programs. The use of operator overloading permits to generate not an intermediate source code, as required by a source transformation approach, thus saving run time memory. ADOL-C permits also the computation of directional derivatives, gradients of all the Taylor coefficients with respect to all the independent variables. Relative to the cost of evaluating the user-defined function, the cost for evaluating any such scalar-vector pair grows as the square of the degree of the derivative, but it is still completely independent of the number of dependent and independent variables. The key ingredient of automatic differentiation by operator overloading is the concept of *active variable*: all the variables that may be considered as differentiable quantities at some point during the evaluation of the program must be considered of an active type. In ADOL-C, such active variables are declared with the library-defined type `adouble`. In order to apply the AD

algorithms, the basic differentiable functions sin, cos, tan, exp, ..., and also the usual binary operators $+$, $-$, $\times$, $\div$ have been *overloaded*.

The first order representation of the system is given below:

$$\mathbf{v} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad \dot{\mathbf{v}} = \begin{bmatrix} 0 & 1 \\ -k & -c \end{bmatrix} \mathbf{v} \rightarrow \dot{\mathbf{v}} = \mathbf{A}\mathbf{v} \tag{4.28}$$

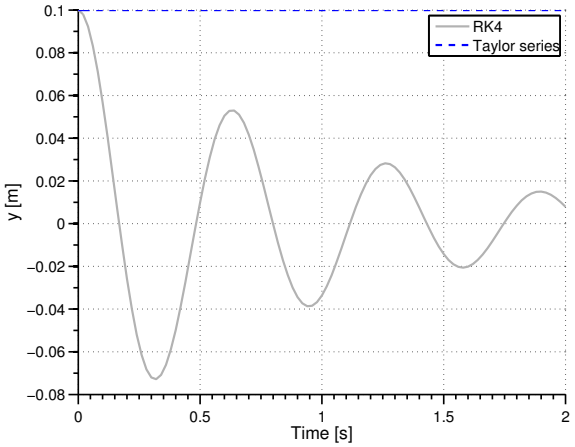
using the normalized time τ:

$$\hat{\mathbf{v}}' = \Delta t \mathbf{A} \hat{\mathbf{v}} \tag{4.29}$$

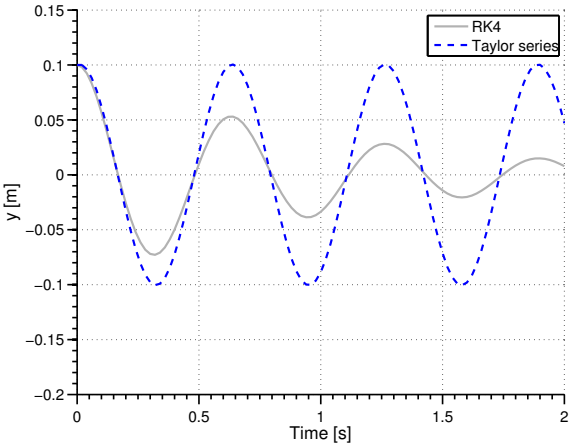
In a C-like program, the fraction of the code representing the right hand side (*rhs*) of the ODE system must be marked as an active section, by the commands `trace_on` and `trace_off`:

```
double pz[2], pzp[2];          // Real input and output
                               // of the rhs evaluation
adouble z[2], zp[2];           // Active variables relative to
                               // pz and pzp respectively

trace_on( tag );               // Begin of the active section
for(i=0; i<2; i++)
{
   z[i] <<= pz[i];             // Assignment of the independent
                               // variables
}

zp[0] = z[1];                  // rhs evaluation
zp[1] = -k*z[0] - c*z[1];

for(i=0; i<2; i++)
{
   zp[i] >>= pzp[i];           // Assignment of the dependent variables
}

trace_off( );                  // End of the active section
```

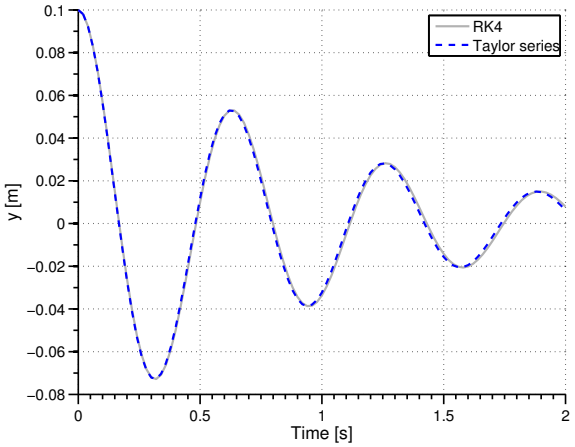
Using the driver `forode`, which implements the AD forward mode, computing recursively the Taylor coefficients of the solution, as explained in 4.3.2, the following result regarding the initial value problem associated to (4.26) has been obtained, considering different Taylor series orders. The output considered is the mass displacement.



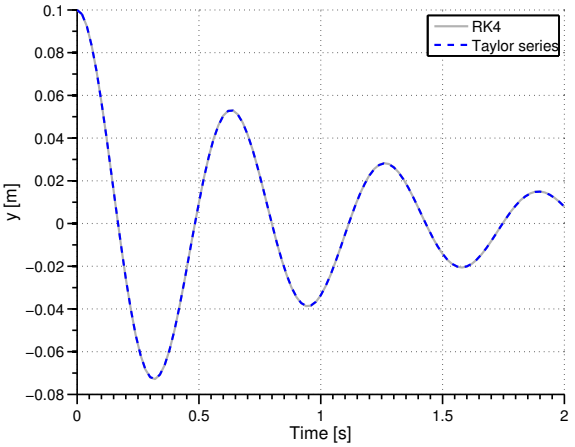
(a) Taylor series' degree = 0



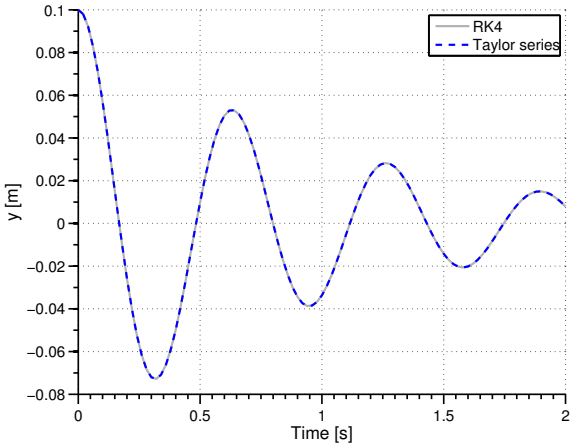
(b) Taylor series' degree = 1



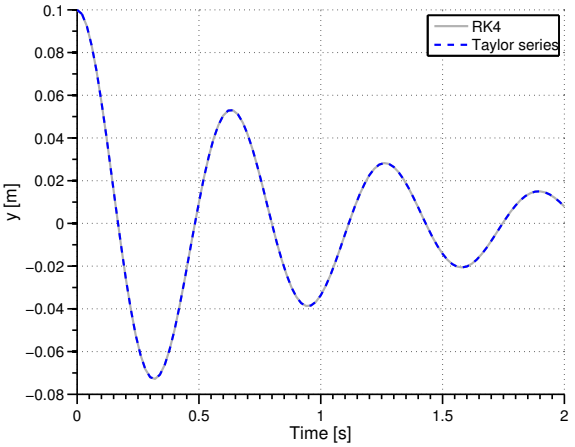
(c) Taylor series' degree = 2



(d) Taylor series' degree = 3



(e) Taylor series' degree = 4



(f) Taylor series' degree = 5

Figure 4.6: AD integrator example - unforced dynamic system

The results are compared to the high-order ODE integrator Runge-Kutta 4 (RK4). It can be seen that above the 2nd order for the Taylor series expansion the results are identical. This result could be predicted, because of RK schemes are derived from imposing the equality of the first Taylor expansion terms of the approximated solution with the corresponding terms of

the Taylor expansion of the exact one [65]. It is curious to note that a first order Taylor series approximation leads to a simply stable system, with no damping. This will be clear with the next analysis of the Taylor series terms.

Since this example is almost trivial, it is possible to compute the Taylor series' expansion elements analytically:

$$\hat{\mathbf{v}}_0 = \hat{\mathbf{v}}|_{\tau=0} \qquad \hat{\mathbf{v}}_1 = \mathbf{A}\hat{\mathbf{v}}_0\Delta t \qquad \hat{\mathbf{v}}_2 = \mathbf{A}^2\hat{\mathbf{v}}_0\Delta t^2 \tag{4.30}$$

and so on. Since the structure of the state dynamic matrix $\mathbf{A}$, a first order approximation with a Taylor series results in a damping term always equal to zero, leading to the simple stable motion shown in figure 4.6b.

The computation of the Jacobian matrix is trivial, since in this case it is simply equal to the matrix $\mathbf{A}$, but ADOL-C is applied in order to compute such matrix. The command `reverse` is used, which clearly implements the AD reverse mode, computing the matrices $\mathbf{A}_i$ presented in (4.16). As expected, only the first term $\mathbf{A}_0$ is not null, equal to the state dynamic matrix $\mathbf{A}$. As demonstration the result of a second order approximation is reported below:

$$\mathbf{A}_0 = \begin{bmatrix} 0 & 1 \\ -100 & -2 \end{bmatrix} \qquad \mathbf{A}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.31}$$

Forced linear, dynamic system

The previous example is repeated here with a slight modification: i.e. the dynamic system is now forced by a sinusoidal input, whose characteristics are reported below: The aim of this example

$f(t) = \begin{cases} 0 & t \leq 0.5\,[s] \\ \sin(\omega_f\, t) & t > 0.5\,[s] \end{cases}$
$\omega_f = 5\,[rad\,s^{-1}]$

Table 4.5: Input's characteristics

is to shown that also a *nonautonomous* system can be integrated with the algorithms explained in 4.3.2, without loss of accuracy. The same time horizon and time step of the previous example are used here.

The dynamic equilibrium now reads:

$$m\,\ddot{x} + c\,\dot{x} + k\,x = f(t) \tag{4.32}$$

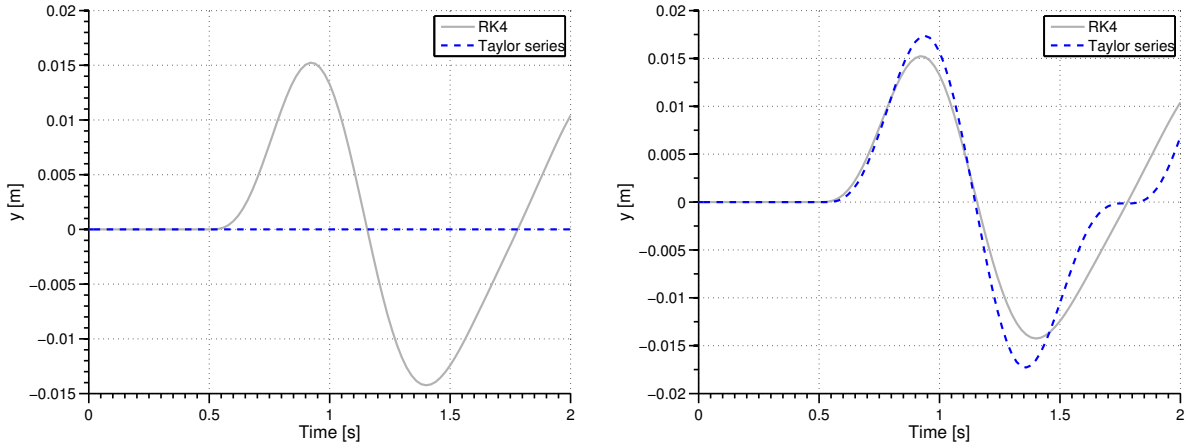
And the first order representation of the system is given below:

$$\mathbf{v} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad \dot{\mathbf{v}} = \begin{bmatrix} 0 & 1 \\ -k & -c \end{bmatrix} \mathbf{v} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f(t) \rightarrow \dot{\mathbf{v}} = \mathbf{A}\mathbf{v} + \mathbf{B}f(t) \tag{4.33}$$

using the normalized time τ:

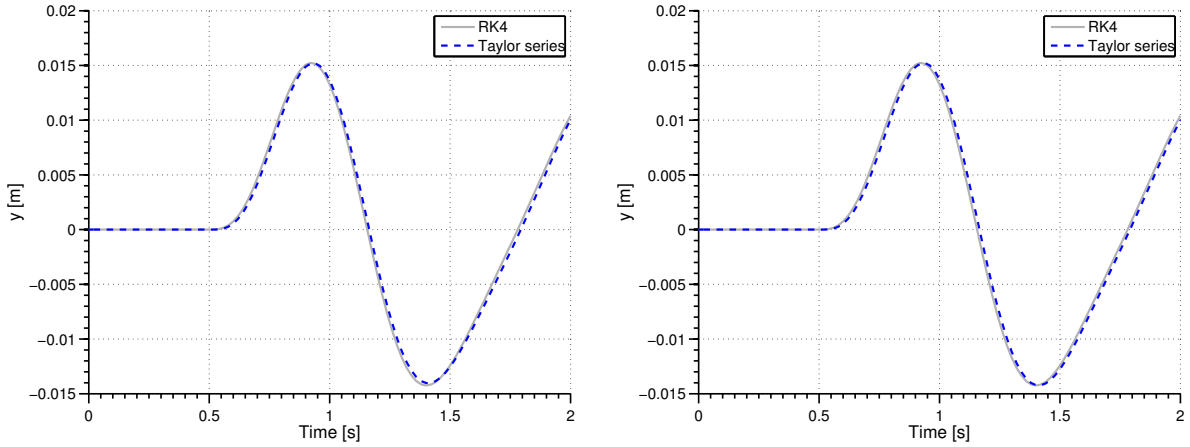
$$\hat{\mathbf{v}}' = \Delta t\mathbf{A}\hat{\mathbf{v}} + \Delta t\mathbf{B}\hat{f}\,(\tau) \tag{4.34}$$

Using the driver `forode`, the following result regarding the initial value problem associated to (4.32) with homogenous initial conditions has been obtained, considering different Taylor series orders. The forcing term is treated explicitly, which means that at each time step considered the forcing is constant and equal to $f = f(\tau = 0)$, meanwhile the Taylor series elements of $\hat{\mathbf{v}}$ are computed at once. The output considered is the mass displacement.



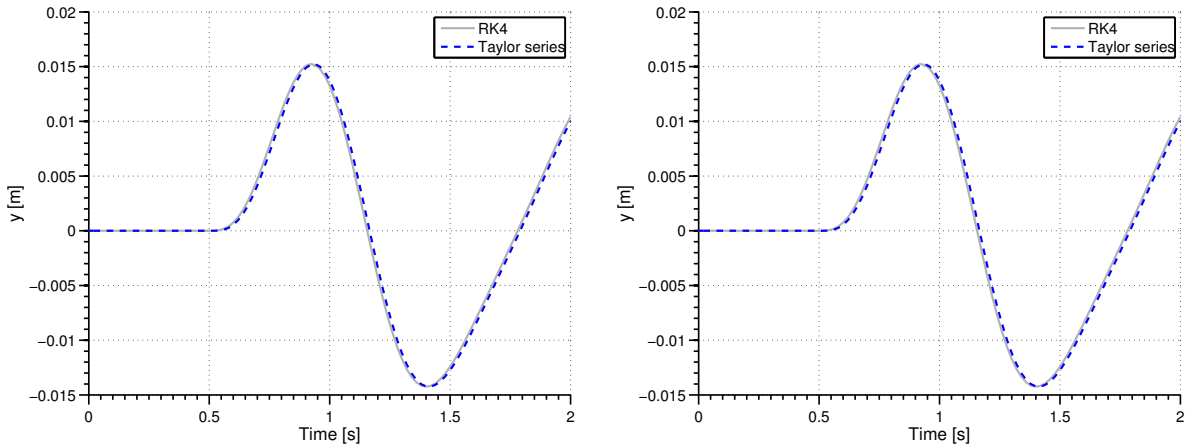
(a) Taylor series' degree = 0

(b) Taylor series' degree = 1



(c) Taylor series' degree = 2

(d) Taylor series' degree = 3



(e) Taylor series' degree = 4

(f) Taylor series' degree = 5

Figure 4.7: AD integrator example - forced dynamic system

The results are compared with the ODE integrator RK4. A perfect match is obtained with a Taylor series order greater than two, moreover, the explicit treatment of the forcing term does not introduce computational errors. This can be due to the fact that the integration time step is much smaller than the period of the forcing term, thus during a time step it will be characterized

by small variations, as a consequence, a local constant approximation is a reasonable solution to the presence of the nonautonomous term.

This consideration can be exploited also in the training of the CTRNN previosly presented, where the external input $\mathbf{u}(t)$ enters in the system dynamics. Since the neural network will work with samples that are coming from a time-accurate CFD simulation, the integration time steps will be so small that a local constant approximation of the input is more than reasonable.

Liénard's system

The final example proposed in this chapter regards the nonlinear systems known as Lièenard's system [44], which is represented by the following oscillator:

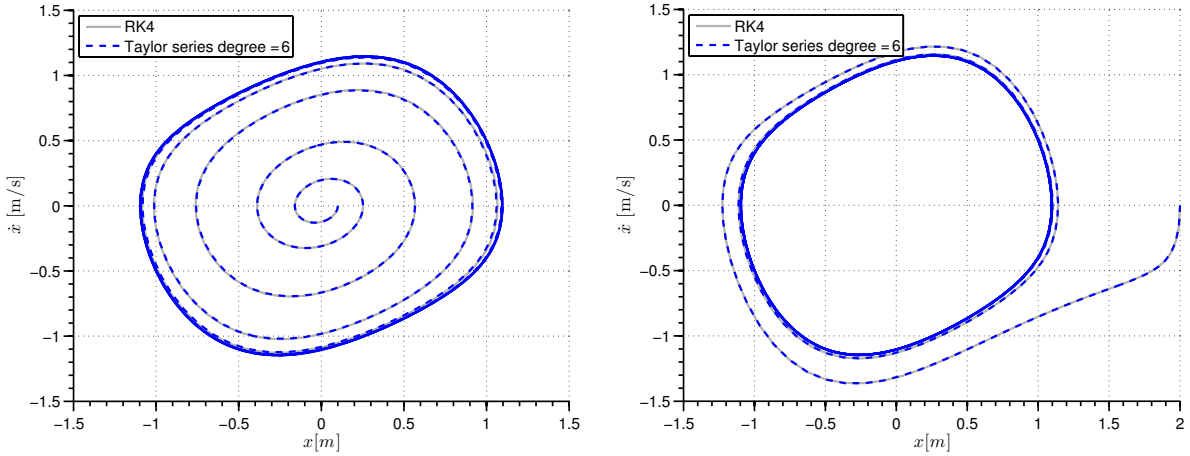
$$\ddot{x} + \left(x^2 - \mu\right)\dot{x} + x = 0 \tag{4.35}$$

This oscillator can be interpreted as a system composed by a mass, a damper and a stiffness, with the damping coefficient which is composed by a constant part and a nonlinear term dependent of the position of the mass. Thus the damping model can be considered as a nonlinear version of the proportional damping model [2]. The system can be recasted in the canonical state-space formulation:

$$\begin{cases} \dot{x} = v \\ \dot{v} = \left(\mu - x^2\right)\dot{x} - x \end{cases} \tag{4.36}$$

The origin is an equilibrium point for all values of μ and the eigenvalues of the Jacobian matrix are $\alpha(\mu) = \frac{1}{2}\left(\mu \pm j\sqrt{4-\mu^2}\right)$. Thus system (4.36) experiences a Hopf bifurcation for $-2 < \mu < 2$.

A time integration with the AD solver proposed has been performed, considering a time horizon $T_h = 100\,[s]$ and a time step $\Delta t = 0.1\,[s]$. The initial condition has been set up to $x_0 = 0.1$ and $v_0 = 0$ in the first simulation and to $x_0 = 2$ and $v_0 = 0$ in the second one. The system parameter as been chosen equal to $\mu = 0.3$, since for this value the bifurcation is supercritical and there is a stable isolated periodic orbit, therefore an LCO [44]. The Taylor series degree considered here is equal to 6. The resulting LCO is represented in the phase space, and it is shown in figure 4.8.



(a) $x_0 = 0.1$ (b) $x_0 = 2$

Figure 4.8: AD integrator example - Lièнard's system

The result obtained is compared to the integrator RK4. The LCO computed has an amplitude of $A_{\text{LCO}} = 1.1\,[m]$ and a circular frequency of $\omega_{\text{LCO}} = 0.9817\,[rad/s]$. The supercritical behaviour of the bifurcation is proved empirically, considering different initial conditions. The results are in good agreement between the two integrators proposed.

The AD integrator and the algorithms involved in the computation of the Jacobian matrix have proved to be fast and accurate, compared to a standard integration scheme. This techniques can be thus directly employed in the training of the CTRNN previously presented, with applications to both the integration of the system dynamics at each optimization iteration and to the computation of the state Jacobian matrix needed by the LM algorithm.

4.4 Network training with AD

The algorithms explained in the sections above are directly applicable to the training of the neural network previosly presented in chapter 3. The CTRNN model can be written as a general parametrization of a nonlinear system:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \theta\right) \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{4.37}$$

Let us define the *network sensitivity* as the variation of the network output against the variation of θ. Thus differentiating (4.37) with respect to θ, one obtain the following matrix ordinary differential problem:

$$\begin{cases} \dot{\mathbf{x}}_\theta = \mathbf{f}_\mathbf{x} \mathbf{x}_\theta + \mathbf{f}_\theta \\ \mathbf{y}_\theta = \mathbf{W}^c \mathbf{x}_\theta \end{cases} \tag{4.38}$$

with $\mathbf{f}_\mathbf{x} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{f}_\theta = \dfrac{\partial \mathbf{f}}{\partial \theta} \in \mathbb{R}^{n_x \times n_\theta}$, $\mathbf{x}_\theta = \dfrac{\partial \mathbf{x}}{\partial \theta} \in \mathbb{R}^{n_x \times n_\theta}$ and $\mathbf{y}_\theta = \dfrac{\partial \mathbf{y}}{\partial \theta} \in \mathbb{R}^{p \times n_\theta}$, taking the same notation of chapter 3.

The above system will be integrated along with the system dynamics (4.37), 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 $n_x + n_x \times n_\theta$ equations, and can be integrated numerically by standard and well known ODE solvers, but 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 burden part of the network training.

Moreover, system (4.37) is *nonautonomous*. The AD package used permits the computation of the Jacobian matrix only respect the state variables [67]. This problem can be solved at hand "upgrading" the nonautonomous part to a state, resulting in:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \theta) \\ \dot{\theta} = \mathbf{0} \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} \end{cases} \tag{4.39}$$

augmenting the initial data of the initial value problem with the value of θ at the beginning of the optimization iteration. It is true that the system of ODEs is augmented in size, however, the differential equations added are trivial and moreover, since the CTRNN should represent a ROM, n_θ will be usually small, thus this apporach will not influence so much the integration procedure.

Thus, at this point let us rewrite the system (4.40) as an autonomous one, defining the extended state variable $\mathbf{z}$:

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \theta \end{pmatrix} \tag{4.40}$$

system (4.37) now reads:

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}) \\ \mathbf{y} = [\mathbf{W}^c \quad \mathbf{0}_{p \times n_\theta}]\, \mathbf{z} = \hat{\mathbf{W}}^c \mathbf{z} \end{cases} \tag{4.41}$$

Introducing the normalized time $\tau = t/\Delta t$, $\hat{\mathbf{z}}' = \Delta t\, \mathbf{f}(\hat{\mathbf{z}}(\tau)) = \hat{\mathbf{f}}(\hat{\mathbf{z}}(\tau))$ thus the arguments discussed in section 4.3.1 and 4.3.2 are directly applicable. The solution of (4.41) can be expressed by the Taylor series:

$$\hat{\mathbf{z}}(\tau) = \hat{\mathbf{z}}_0 + \hat{\mathbf{z}}_1 \tau + \hat{\mathbf{z}}_2 \tau^2 + \cdots + \hat{\mathbf{z}}_d \tau^d + \mathrm{o}\left(\tau^{d+1}\right) \tag{4.42}$$

The Taylor elements of $\hat{\mathbf{z}}$ are defined by:

$$\hat{\mathbf{z}}_{i+1} = \frac{\hat{\mathbf{f}}_i}{i+1} \qquad i = 0, \ldots, \mathrm{d} - 1 \tag{4.43}$$

starting from $\hat{\mathbf{z}}_0 = \hat{\mathbf{z}}|_{\tau=0}$, all the desired elements can be computed efficiently with a standard AD package that implements the forward AD mode. The state Jacobian matrix can be computed as explained in 4.3.2. First the Jacobian matrix path of $\mathbf{f}$ must be computed:

$$\frac{\partial \hat{\mathbf{f}}}{\partial \hat{\mathbf{z}}} = \mathbf{A}_0 + \mathbf{A}_1\, \tau + \mathbf{A}_2\, \tau^2 + \cdots + \mathbf{A}_d\, \tau^d + \mathrm{o}\left(\tau^{d+1}\right) \tag{4.44}$$

the matrices $\mathbf{A}_i \in \mathbb{R}^{n_x \times n_x}$, $i = 1, \ldots, d$ can be efficiently computed by the reverse mode of AD. Once these matrices have been computed, algorithm (4.20) can be applied in order to compute the state Jacobian matrix:

$$\mathbf{B}_{i+1} = \frac{\mathrm{d}\hat{\mathbf{z}}_{i+1}}{\mathrm{d}\hat{\mathbf{z}}_0} = \frac{1}{i+1} \frac{\mathrm{d}\hat{\mathbf{f}}_i}{\mathrm{d}\hat{\mathbf{z}}_0} = \frac{1}{i+1} \sum_{j=0}^{i} \frac{\partial \hat{\mathbf{f}}_i}{\partial \hat{\mathbf{z}}_j} \frac{\mathrm{d}\hat{\mathbf{z}}_j}{\mathrm{d}\hat{\mathbf{z}}_0} = \frac{1}{i+1} \sum_{j=0}^{i} \mathbf{A}_{i-j} \mathbf{B}_j \tag{4.45}$$

At this point, dividing the simulation time in N_t uniform time steps $\Delta t = \dfrac{T_h}{N_t}$, the state and its Jacobian matrix will be known at a set of discrete time points $t_i = i\, \Delta t$, $i = 0, \ldots, N_t$, applying the definitions:

$$\begin{aligned} \hat{\mathbf{z}}(1) &= \sum_{j=0}^{d} \hat{\mathbf{z}}_j \\ \mathbf{y}(t_{i+1}) &= \hat{\mathbf{W}}^c\, \hat{\mathbf{z}}(1) \\ \mathbf{B}(1) &= \frac{\mathrm{d}\hat{\mathbf{z}}(1)}{\mathrm{d}\hat{\mathbf{z}}_0} = \sum_{j=0}^{d} \mathbf{B}_i \\ \mathbf{J}(t_{i+1}) &= -\hat{\mathbf{W}} \mathbf{B}(1) \end{aligned} \tag{4.46}$$

with $\mathbf{J}$ defined in chapter 3 and $\hat{\mathbf{W}}$ the matrix which extracts the first p rows and the last n_θ columns from $\mathbf{B}$. At each time step, the initial data will be defined as:

$$\begin{aligned} \hat{\mathbf{z}}_0 &= \hat{\mathbf{z}}(0) \\ \mathbf{B}_0 &= \mathbf{I}_{(n_x+n_\theta)} \end{aligned} \tag{4.47}$$

Chapter 5

Aerodynamic identification

In this chapter the ROM technique presented in chapter 3 is implemented to identify the aerodynamic model of a transonic flow past a pitching airfoil. First the various CTRNN models presented previously are trained to some specific training signal. Then their validation capacity is tested to an harmonically varying pitching motion at an amplitude and frequency that has not been encountered during the training.

The chapter is so structured: in section 5.1 the CFD solver is validated over a steady test case and an adapt number of finite volumes is chosen for performing the unsteady simulations. Section 5.2 treats the training of the different network parametrizations proposed with different training signals. The performance of the different networks in term of computational time required for the training and the value of the error at the end of the optimization are compared and discussed. In section 5.3 the generalization performances of the different networks are compared, presenting to the networks two input signals which has not be used in the training. Finally, in section 5.4 some consideration are given about these first performances of the different parametrizations proposed, discussing in particular the efficiency of the ROM in the representation of a large dynamic nonlinear system and the generalization properties regarding the two training input considered.

5.1 Validation of the CFD solver

The airfoil chosen for this test case is the symmetric *NACA 64A010*. This series of airfoils has been designed in order to maximize the region over which the airflow remains laminar. In this way, the drag over a small range of lift coefficients can be substantially reduced. The NACA 6-series have also nowadays a large range of applications, from aeronautics to wind turbine design. This kind of airfoils permit to reach high maximum lift coefficient, a very low drag over a small range of operating conditions and they are optimized for high speed. On the other side, they present high drag outside the optimum range of operating conditions, high pitching moments and present a poor stall behaviour.

The Mach number considered in this study is $M_\infty = 0.8$, a typical transonic flow condition, where a strong shock is usually present on the upper surface of the airfoil for positive angle of attacks. The computational mesh chosen is a *C-mesh*. Since the Euler flow model has been chosen in this analysis, an O-mesh could be used as well, but the C-type one has been selected considering a further possible analysis with a Navier-Stokes model, whereas the latter type of mesh permits to achieve a better accuracy than an O-type one, because the elements in the wake will be aligned with the primary direction of the flow. An example of C-mesh is given in figure 5.1, meanwhile the close-up of the airfoil is shown in figure 5.2. The computational mesh, which is a structured grid of quadrilateral elements, has been created with a script written in MATLAB programming language, which permits to choose the number of divisions on the

airfoil, the number of divisions along the radius of the semicircular part of the mesh and the number of divisions of the wake. The radial dimension has been set to 20 chords, meanwhile the wake dimension has been set to 30 chords. The Reynolds number can be imposed in order to determine the dimensions of the computational volumes next by the airfoil, in case of viscous simulations [20]. The output of this script is a `plot3d` mesh format that can be converted in a OpenFOAM readable mesh by the command `plot3dToFoam`.

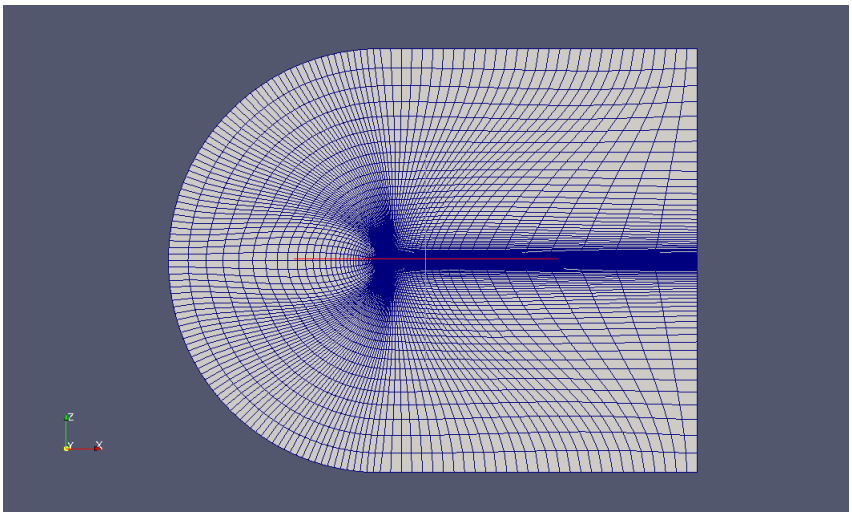


Figure 5.1: C-type mesh

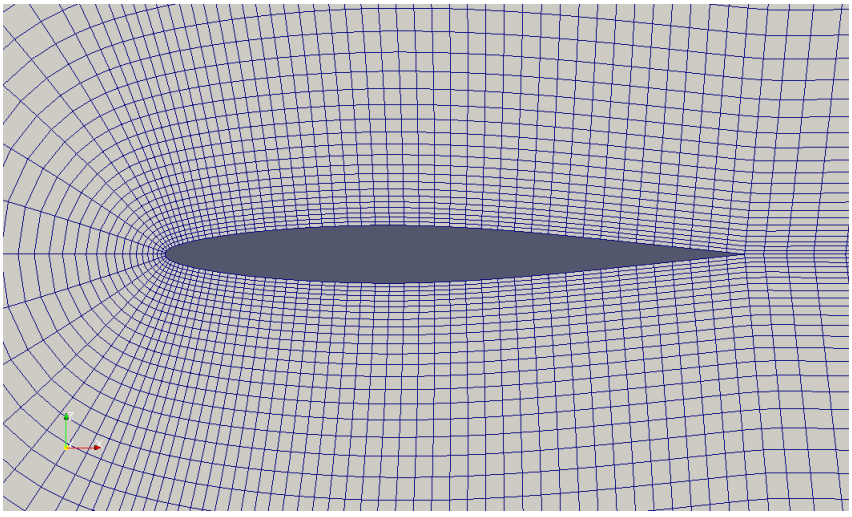


Figure 5.2: Close-up of the *NACA 64A010* airfoil

A convergence analysis as been carried out in order to choose the number of finite volumes involved in the computation. Unfortunately, the only reference result found is a RANS simulation performed at $Re = \frac{V_\infty c}{\nu} = 1 \cdot 10^6$ and $\alpha = 4\,[deg]$, found in [68]. It has been found out that this kind of airfoil has a very confusing definition, since it is often taken for the *NACA 64A010A* airfoil, a symmetric variant on the *AGARD 156* benchmark section [69]. Hovewer, three mesh of variable volumes has been tested and the results are shown in figure 5.3.

The Euler model has been adopted in these simulations. The MG solver is selected to speed-up the convergence. The time integration scheme selected is the first order accurate RK5, with an imposed time step $\Delta t = 1 \cdot 10^{-6}\,[s]$. The maximum CFL allowed is equal to 3, as indicated in the literature for this kind of integrator [20]. The solver will automatically select the correct time step: if the imposed time step is smaller than the CFL constrained one, then the smaller will be

chosen. On the other hand, when the imposed time step is greater than the one constrained by the CFL number, then the latter is chosen.

All the computations have been performed on a computer with an Intel® Core™ 2 Duo CPU with 2.93 GHz of maximum frequency, 4 GB of RAM memory, 3 MB of cache memory and a Linux operating system with kernel updated to version 2.6.32.

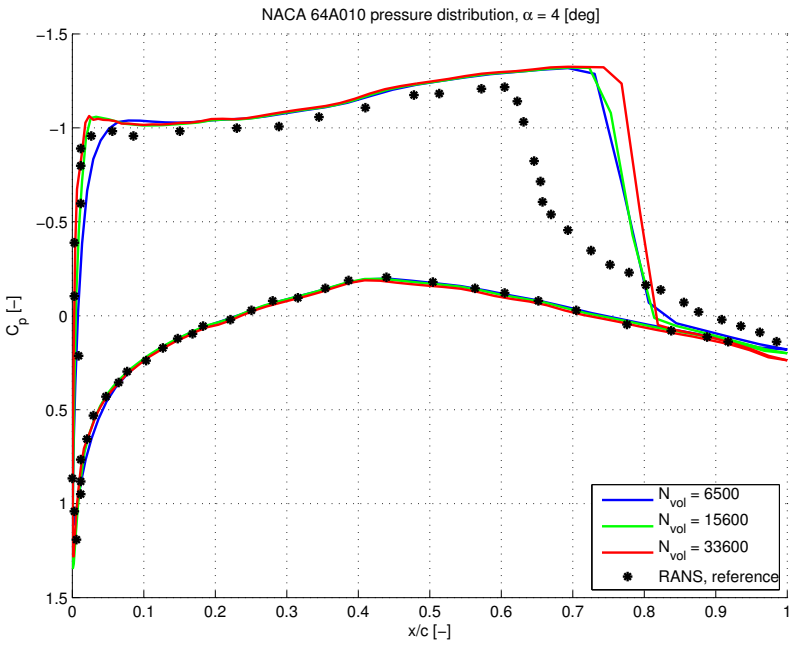


Figure 5.3: Mesh convergence test

In table 5.1 are listed the computational time requested to obtain a convergent result The simulation has been considered converged when the norm of the residual $\mathbf{R}$, presented in chapter 3, reached the value of 10^{-4}.

Number of volumes N_{vol}	Time to convergence $[h : min]$
6500	0 : 7
15600	0 : 15
33600	0 : 48

Table 5.1: Convergence time of the various meshes

Observing the results shown in figure 5.3, it can be noted that there is a good agreement with the reference RANS solver in all points except near the shock, where the effect of the viscosity can be important and thus an inviscid flow model does not permits to resolve the continuos transition between the supersonic and the subsonic region captured by the viscous solver. Moreover, since the position of the shock is given by a balance of pressures, the inviscid case places the discontinuity downward the viscous solution. However, considering a compromise of accuracy and computational time, a mesh with $N_{vol} = 12200$ finite volumes has been chosen, thus a mesh with a number of volumes between the medium and the coarse one considered in the steady convergence test. The computational time should be as small as possible, mantaining a good accuracy, considering also that the chosen mesh will be used for unsteady simulations. The chosen mesh has been tested at various angle of attack α, and the resulting pressure distributions is shown in figure 5.4. For $\alpha = 0\,[deg]$ two strong shocks are present both on the upper and the lower surface of the airfoil. For $\alpha = 4\,[deg]$ and $\alpha = 5\,[deg]$ there is a strong shock on the upper

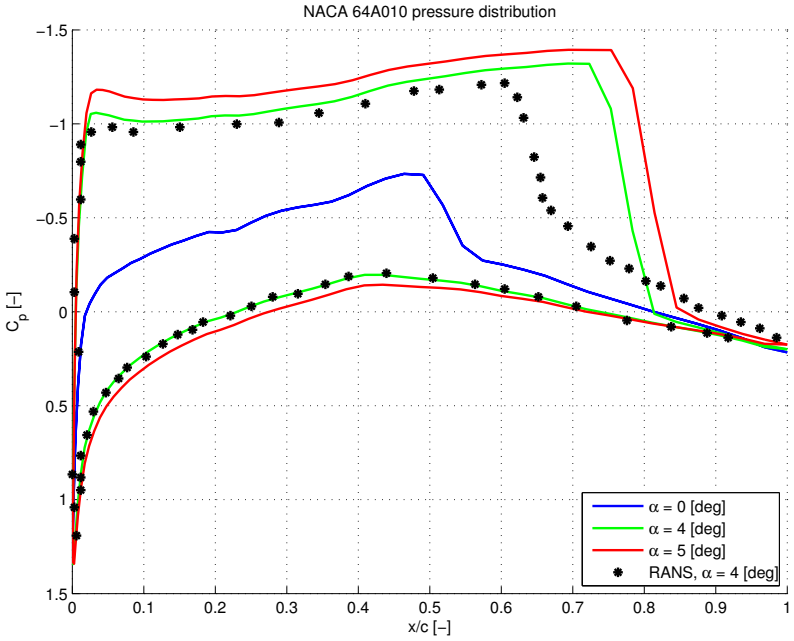


Figure 5.4: Various pressure distributions for the *NACA 64A010*

surface near the trailing edge of the airfoil, but the shock on the lower surface has vanished. The distribution of the pressure coefficient $C_p = \dfrac{p - p_\infty}{1/2\,\rho_\infty\, V_\infty^2}$ is shown in figure 5.5, where it is evident the previous consideration.

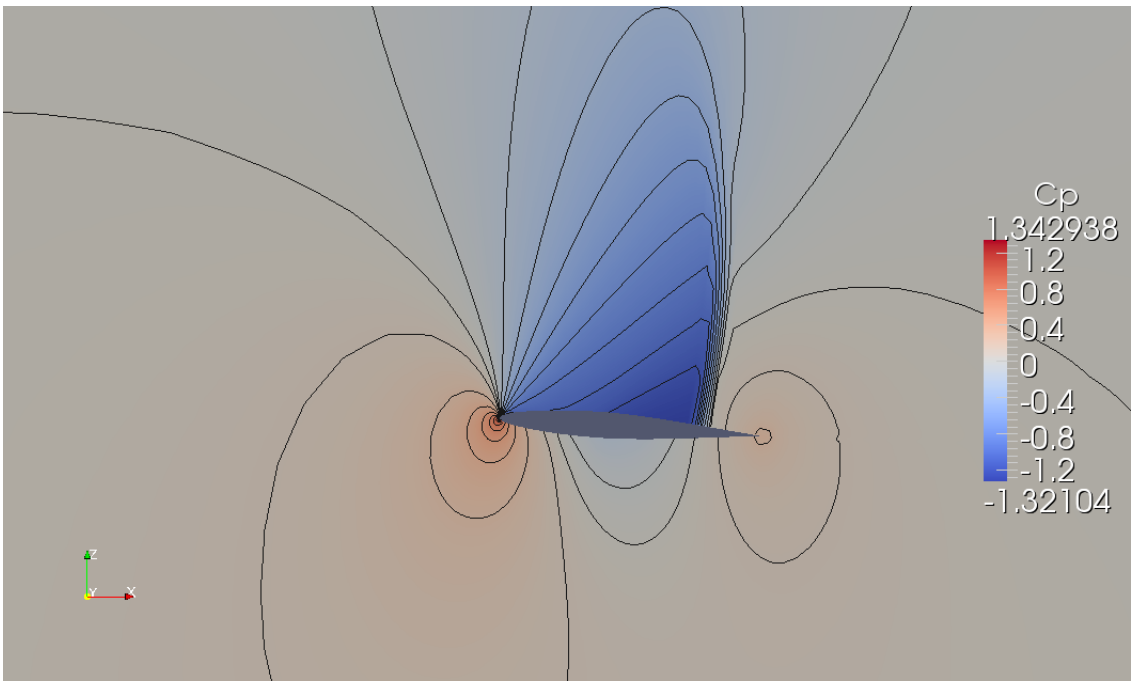


Figure 5.5: C_p contours for the *NACA 64A010* at $\alpha = 4\,[deg]$

5.2 CTRNN training

The airfoil under study is pitching around its elastic axis, positioned at 20% of the chord, at the same asymptotic condition of the steady simulation. The airfoil has an unit chord, and the asymptotic velocity is equal to $V_\infty = 272.21\,[m/s]$. Thus the input signal of the network will be the pitch angle of the airfoil, θ.

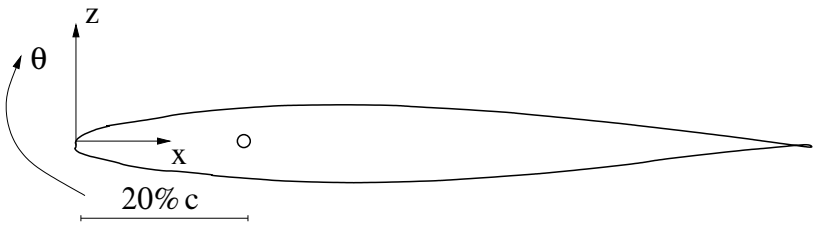


Figure 5.6: 1 DOF airfoil

The system that have to be identified is therefore a Single Input Multi Output system (SIMO), since the outputs considered are the coefficient of aerodynamic lift C_L and moment C_M computed around the rotation axis, their definition is given below:

$$C_L = \frac{L}{q_{\text{dyn}}\, c} \qquad C_M = \frac{M}{q_{\text{dyn}}\, c^2} \tag{5.1}$$

Figure 5.7: SIMO aerodynamic identification

The unsteady computation is set in AeroFoam essentially with the same setting of the steady problem, with only small differences, as explained in the following. The file `.dat` containing the arbitrary input signal is stored in `TestCase/Data`. The time solver method chosen in a the Dual Time Stepping (DTS) technique, coupled with a FAS MG strategy. In the specific, the integration method which operates in the pseudo-time domain is 1st order accurate, the threshold residual for obtaining the convergence is fixed to 10^{-3} and a maximum of 300 iterations are allowed before stopping the iterative procedure, considering the actual time step converged. The physical time step considered is equal to $\Delta t = 3 \cdot 10^{-3}\,[s]$ and the CFL number chosen is equal to 2.5. The time-marching integration scheme employed is a RK5. The computational experiment in performed considering air as working fluid, assumed in standard conditions: $p_\infty = 101325\,[Pa]$, $T_\infty = 288.15\,[K]$, $\rho_\infty = 1.225\,[kg\,m^{-3}]$, $\gamma = 1.4$.

Since the network output C_L and C_M have tipically different orders of magnitude, for example $C_L \sim \mathcal{O}\left(10^{-1}\right)$ and $C_M \sim \mathcal{O}\left(10^{-2}\right)$ as the angle of attack is small, in order to improve the convergence of the training algorithm, the outputs have been normalized by their maximum value obtained with the CFD simulation. Thus a diagonal normalization matrix $\mathbf{W}_o$ is employed in order to obtain a normalized target:

$$\hat{\tilde{\mathbf{y}}} = \mathbf{W}_o \hat{\mathbf{y}} \tag{5.2}$$

with:

$$W_{o_{ii}} = \frac{1}{\max_{t \in T_{\text{sim}}} \hat{y}_i\,(t)} \qquad i = 1, \ldots, \mathrm{N}_{\text{output}} \tag{5.3}$$

In this manner, with a normalized output, along the convergence path, the cost function F, defined in chapter 3, weights in the same manner the error of both output, thus does not incurring

in a optimization which reduces only the error on the output with the greatest order of magnitude. Therefore, the error considered in the algorithm is so defined, recalling the definition given in chapter 3:

$$\hat{\mathbf{e}}(t_i) = \mathbf{W}_o \hat{\mathbf{y}}(t_i) - \mathbf{W}^c \mathbf{x}(t_i) \qquad i = 1, \dots, N_t \tag{5.4}$$

Consequently, as the training algorithm has achieved the convergence, the output of the network can be obtained exploiting the definition of the weighting matrix itself:

$$\mathbf{y}(t) = \mathbf{W}_o^{-1} \mathbf{W}^c \mathbf{x}(t) \tag{5.5}$$

The random and hybrid signals have been chosen as training input from chapter 3, since the aim of this analysis is to predict with the ROM the unsteady aerodynamic loads acting on an harmonically pitching airfoil. These training signals contain a frequency content that should permit the CTRNN to generalize whatever unsteady input that will be presented in the validation phase. Both these training signals are tested on the three CTRNN possible parametrization presented, with a final comparison of the results.

5.2.1 Training algorithm setting

It has been said in chapter 3 that the training algorithm adopted in this work is an hybrid algorithm which utilizes the GA in the initialization phase, starting from random values of the network synaptic weights, combined to the LM optimization algorithm in the refining phase of the training. In this manner small value of the error can be achieved, considering the good convergence properties of the GA and the convergence of the second order of the LM algorithm near the optimal solution.

Let us consider the setting of the two algorithms in this chapter. The population size of the GA has been chosen equal to n_θ in order to sweep the entire $\mathbb{R}^{n_\theta}$ during the initialization phase. The probability of mutation has been fixed to 0.15 and the maximum number of generation has been selected to be 100. This value has proved to be a good compromise between a good accuracy at the end of the initialization phase and the computational time required. The range of values where the synaptic weights are allowed to vary is fixed to $[-10, 10]$, and the threshold considered for the convergence has been set to 10^{-3}.

For what regard the LM algorithm, standard values for the converging parameters has been assumed, for example the threshold for the cost function variation has been fixed to 10^{-8}, the same value has been chosen for the threshold value of the squared sum of the variation of all the synaptic weights. The maximum number of iterations allowed has been abundantly set to 400.

Finally, considering the settings of the AD integrator explained in chapter 4, it has been selected the same time step utilized by the CFD simulation, and the order of the Taylor series employed for solving the ODE system proposed in chapter 3 has been fixed to 6. All the figures presented in this section show the results obtained with the AD training.

5.2.2 Random training

The training signal chosen in this section is a random signal, whose frequency characteristics are imposed by the formula explained in chapter 3. The computational time required for the CFD simulation has been $4\,[h], 17\,[min]$. The resulting signal and frequency content are reported in figure 5.8 and 5.9.

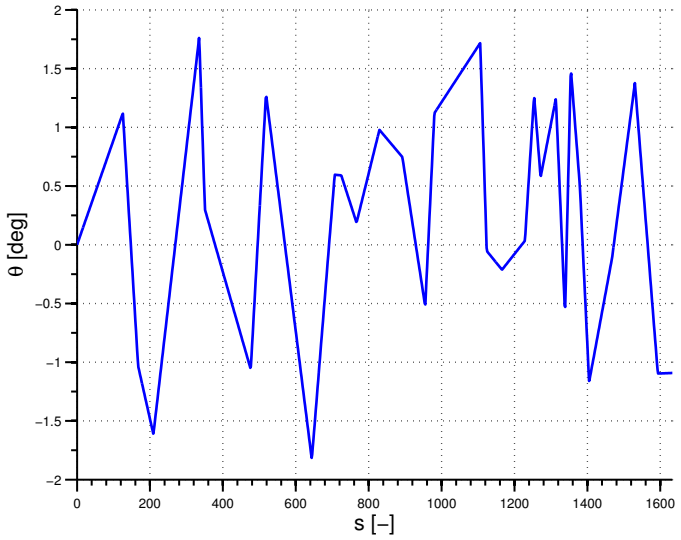


Figure 5.8: Random input for the 1 DOF airfoil

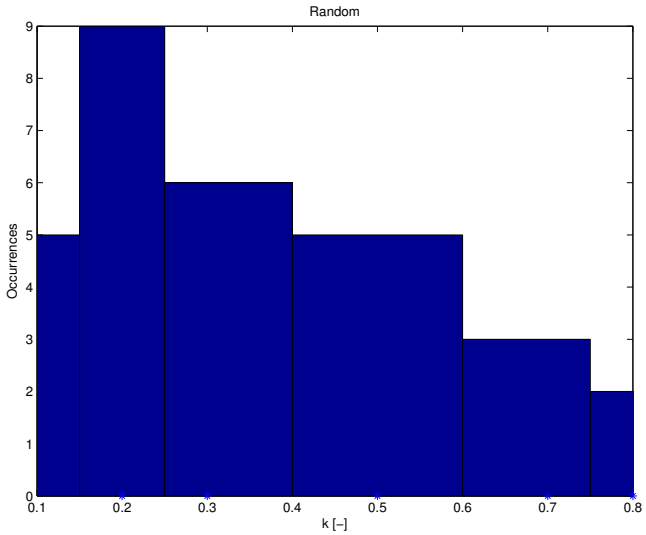


Figure 5.9: Frequency characteristics of the input signal

The training signal is composed by $N_t = 1000$ samples. In the sections below the different parametrizations presented in chapter 3 are trained considering three kind of training: a training where the Jacobian matrix needed in the LM algorithm is computed by finite difference, one where the Jacobian matrix is composed by its analytical expression and finally the AD training is considered. A comparison of computational time and convergence precision is discussed for each type of parametrization.

ECTRNN

The ECTRNN is trained with the input shown in figure 5.8, and the training results are shown in figures 5.10 and 5.11. Different network orders are considered here, and the results obtained are compared in term of accuracy and computational time. Calling the two networks A and B, their characteristics in term of system states are reported in table 5.2.

	A	B
n_x	2	5
n_h	4	4
n_θ	32	44

Table 5.2: Different ECTRNN order employed

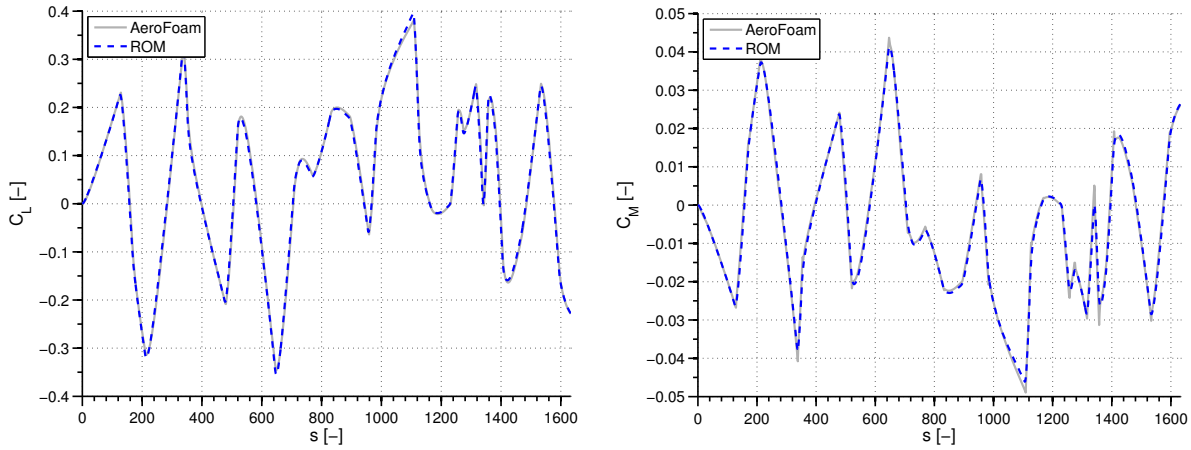


Figure 5.10: ECTRNN A, random training - 1 DOF airfoil

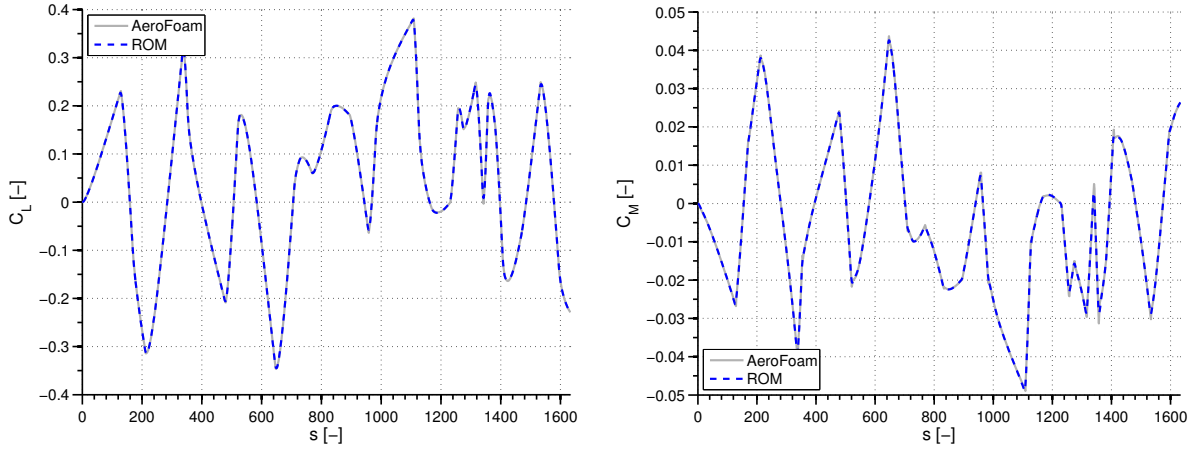


Figure 5.11: ECTRNN B, random training - 1 DOF airfoil

The training has produced good results, with a good fit of the aerodynamic response given by the solver AeroFoam. The various computational time required for the training and the order of convergence, represented by the value of the cost function F, presented in chapter 3, are given in table 5.3 for the network A and table 5.4 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	70	$\mathcal{O}\left(10^{-1}\right)$
Analytic	58	$\mathcal{O}\left(10^{-2}\right)$
AD	52	$\mathcal{O}\left(10^{-2}\right)$

Table 5.3: Convergence properties, ECTRNN A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	90	$\mathcal{O}\left(10^{-1}\right)$
Analytic	64	$\mathcal{O}\left(10^{-2}\right)$
AD	55	$\mathcal{O}\left(10^{-2}\right)$

Table 5.4: Convergence properties, ECTRNN B

As expected, the convergence time increases as the order of the network increases. However, the performances remains unaltered, which can mean that the the order of the network A is sufficient to capture the nonlinear dynamics of the aerodynamic system. From the above results, it is obvious that the analytic or the AD approach are order of magnitude more efficient than the finite difference approach in the computation of the Jacobian matrix. Maybe, the AD results can be further improved, since no code optimization has been implemented in this work.

C^{2}TRNN

The C^{2}TRNN is trained with the input shown in figure 5.8, and the training results are shown in figure 5.12 and 5.13. Different network orders are considered here, and the results obtained are compared in term of accuracy and computational time. Calling the two networks A and B, their characteristics in term of system states are reported in table 5.5.

	A	B
n_x	2	5
n_h	4	4
n_θ	25	36

Table 5.5: Different C^{2}TRNN orders employed

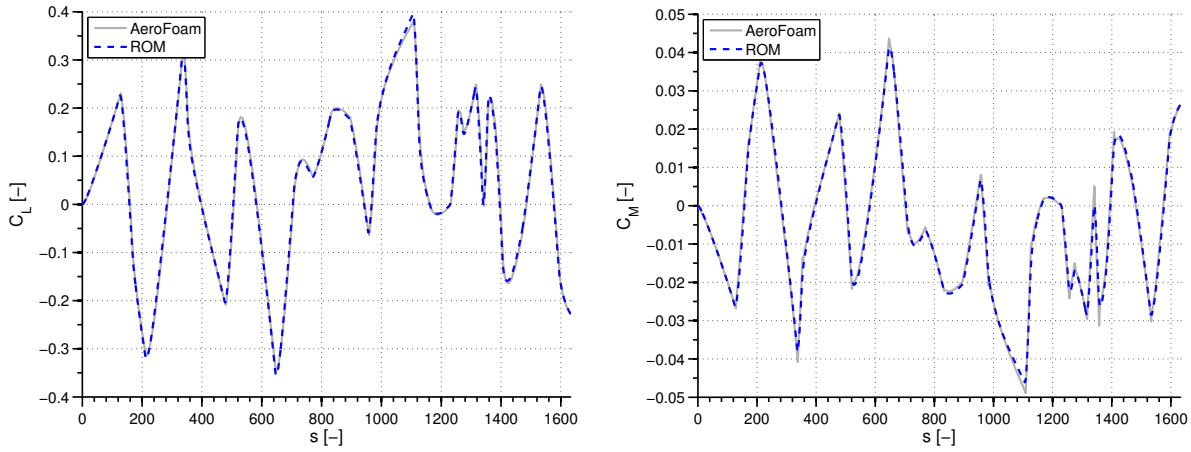


Figure 5.12: C^{2}TRNN A, random training - 1 DOF airfoil

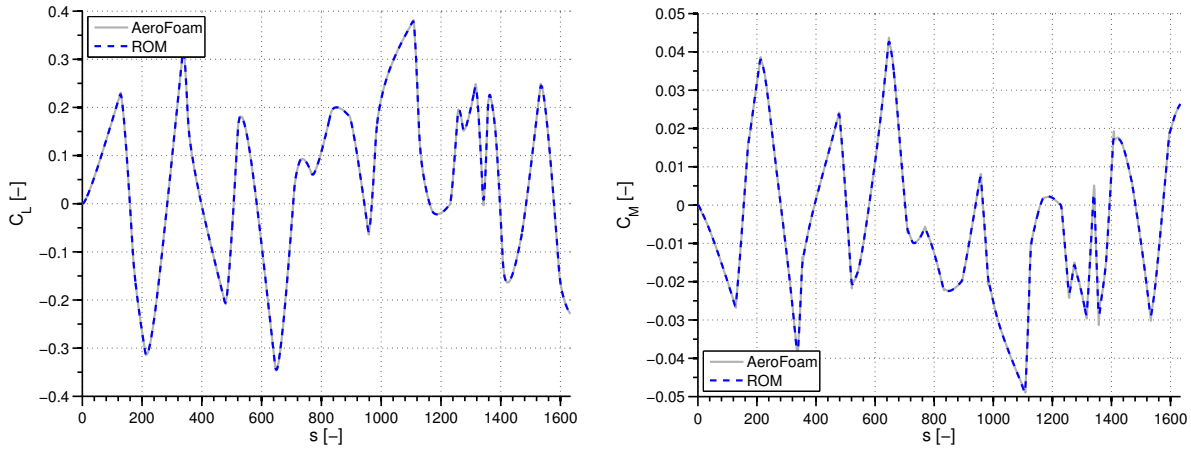


Figure 5.13: C^{2}TRNN B, random training - 1 DOF airfoil

The training has produced good results, with a good fit of the aerodynamic response given by the solver AeroFoam. The various computational time required for the training and the order of convergence, represented by the value of the cost function F, are given in table 5.6 for the network A and table 5.7 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	30	$\mathcal{O}(10^{-2})$
Analytic	12	$\mathcal{O}(10^{-3})$
AD	8	$\mathcal{O}(10^{-4})$

Table 5.6: Convergence properties, C^2TRNN A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	40	$\mathcal{O}(10^{-1})$
Analytic	22	$\mathcal{O}(10^{-3})$
AD	15	$\mathcal{O}(10^{-3})$

Table 5.7: Convergence properties, C^2TRNN B

It is very interesting to note that the computational time has been strongly decreased respect the ECTRNN case. This can be due to the smaller size of the multi-dimensional space where the optimization algorithm searches the minimum-error-solution. Smaller size, means less generation to be trained in the GA algorithm and less modifications to the current solution. It has been noted that the convergence time of the algorithm is governed by the convergence time of the initialization phase, in this case performed with the GA algorithm. Heuristically, it can be said that the computational time required for the initialization of the network weight accounts for the 60% of the total computational time. A possible solution that can be considered in the future is the computational parallelization of this phase, reducing drastically the initialization phase time.

CTRNN.LI

The CTRNN.LI is trained with the input shown in figure 5.8, and the training results are shown in figure 5.14 and 5.15. Different network orders are considered here, and the results obtained are compared in term of accuracy and computational time. Calling the two networks A and B, their characteristics in term of system states are reported in table 5.8.

	A	B
n_x	4	6
n_θ	20	42

Table 5.8: Different CTRNN.LI orders employed

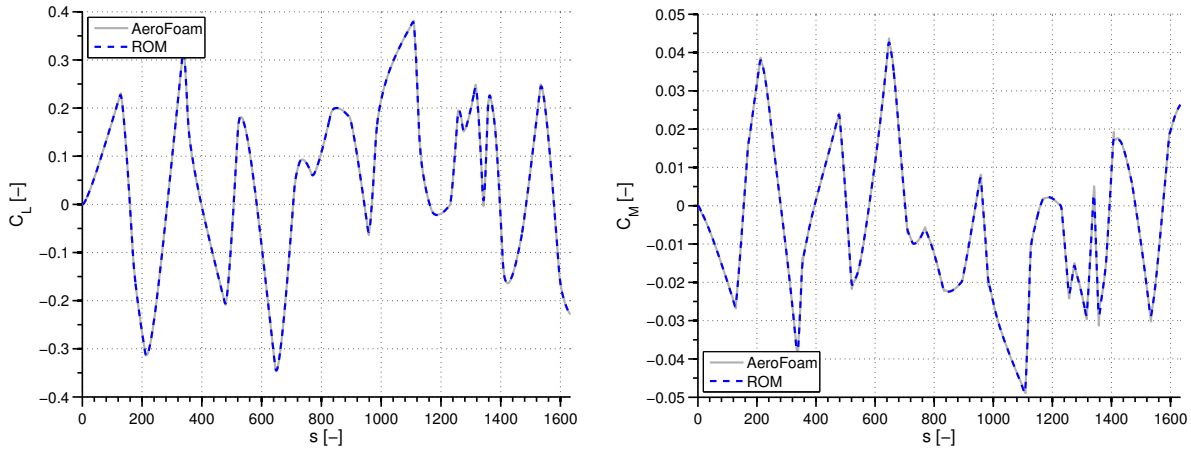


Figure 5.14: CTRNN.LI A, random training - 1 DOF airfoil

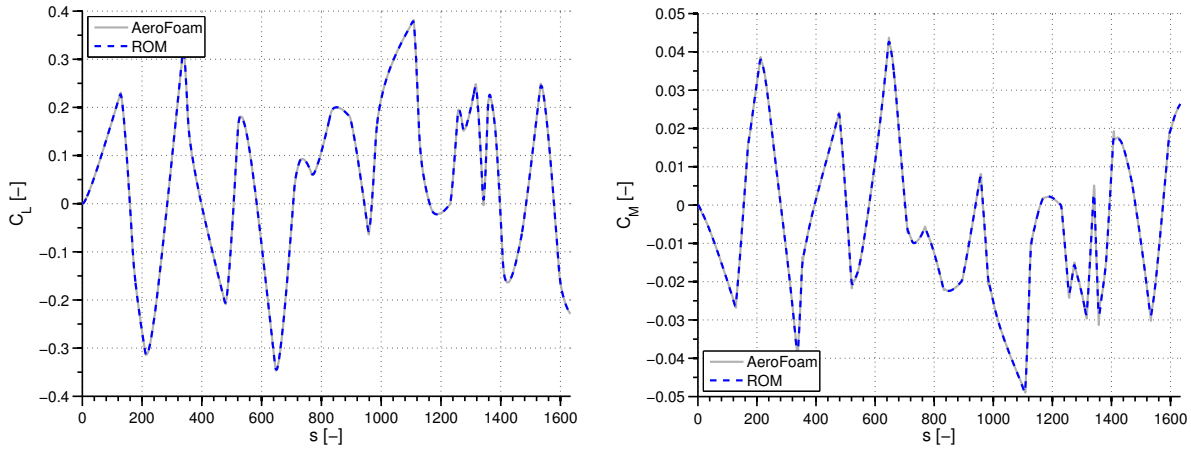


Figure 5.15: CTRNN.LI B, random training - 1 DOF airfoil

The training has produced good results, with a good fit of the aerodynamic response given by the solver AeroFoam. The various computational time required for the training and the order of convergence, represented by the value of the cost function F, are given in table 5.9 for the network A and table 5.10 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	28	$\mathcal{O}\left(10^{-2}\right)$
Analytic	18	$\mathcal{O}\left(10^{-3}\right)$
AD	10	$\mathcal{O}\left(10^{-4}\right)$

Table 5.9: Convergence properties, CTRNN.LI A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	63	$\mathcal{O}\left(10^{-1}\right)$
Analytic	40	$\mathcal{O}\left(10^{-2}\right)$
AD	32	$\mathcal{O}\left(10^{-3}\right)$

Table 5.10: Convergence properties, CTRNN.LI B

The computational time required for the training of this network was expected to be smaller than the other models. Instead, it has been found out that the training time is smaller than the ECTRNN , but it is slightly greater than the C^2TRNN . This can be due to the parametrization of the network itself, since in the other models the network input enters directly inside each neurons activation function, instead in this case the matrix $\mathbf{W}^b$ is outside the activation function. This fact can cause that the range in which the values of $\mathbf{W}^b$ can vary is modified, and this may influence the convergence time.

5.2.3 Hybrid training

The training signal chosen in this section is a hybrid signal, whose frequency and amplitude is chosen randomly as explained in chapter 3. The computational time required for the CFD simulation has been $5\,[h], 0\,[min]$. The resulting signal and frequency content are reported in figure 5.16 and 5.17.

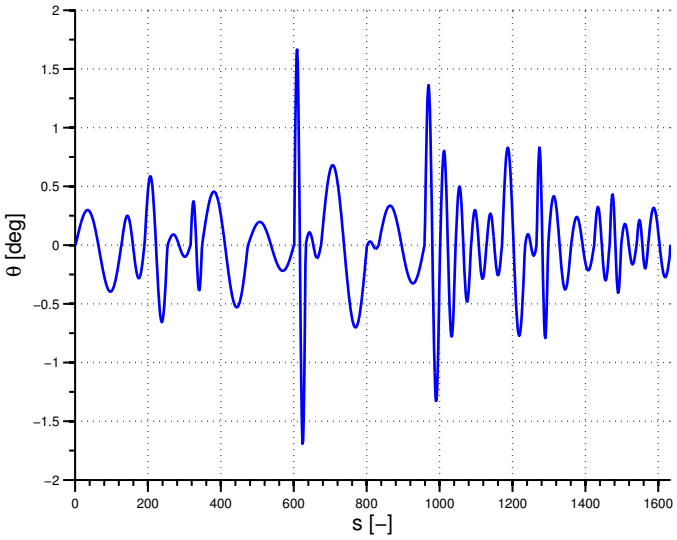


Figure 5.16: Hybrid input for the 1 DOF airfoil

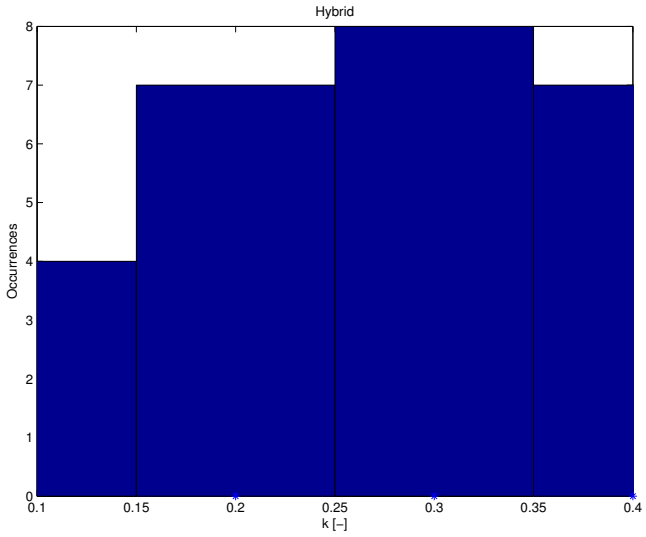


Figure 5.17: Frequency characteristics of the input signal

It is evident that the training signal contains a wide spectrum of both frequencies and amplitudes, this its should be an adapt training input for the computation of unsteady aerodynamic loads. The training signal is composed by $N_t = 1000$ samples. In the sections below the different network parametrizations presented in chapter 3 are trained considering three kind of training: a training where the Jacobian matrix needed in the LM algorithm is computed by finite difference, one where the Jacobian matrix is composed by its analytical expression and finally the AD training is considered. A comparison of computational time and convergence precision is discussed for each type of parametrization.

ECTRNN

The ECTRNN is trained with the input shown in figure 5.16, and the training results are shown in figure 5.18 and 5.19. The same networks A and B considered previously in the random training are proposed here for the hybrid training.

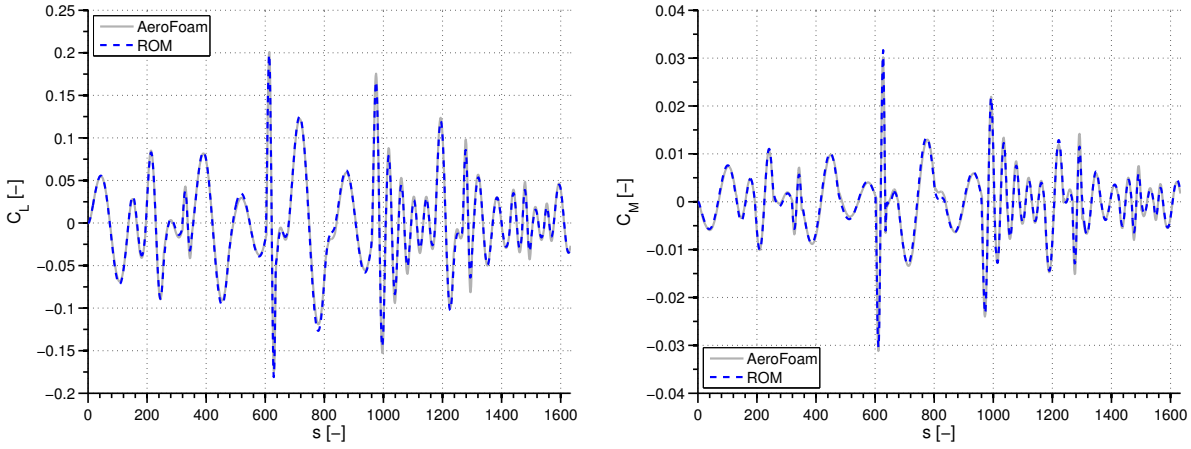


Figure 5.18: ECTRNN A, hybrid training - 1 DOF airfoil

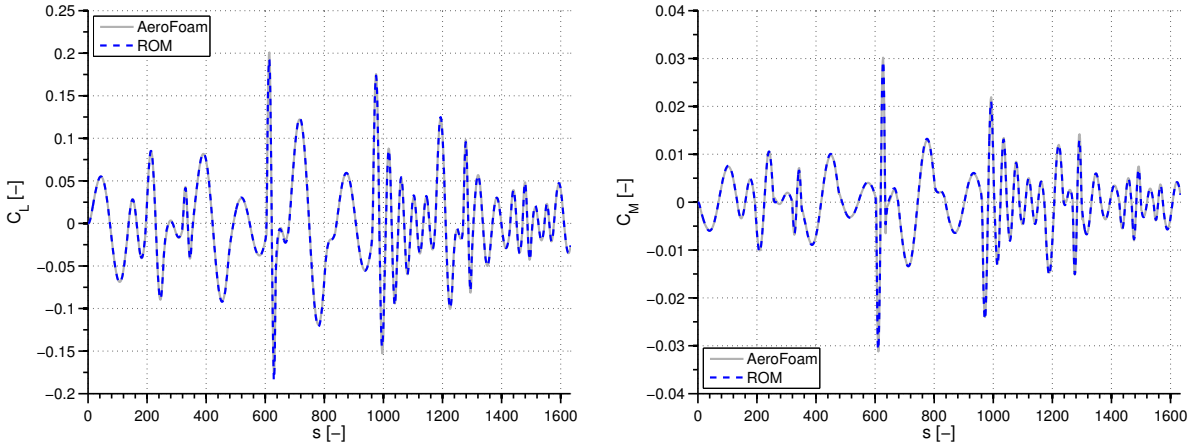


Figure 5.19: ECTRNN B, hybrid training - 1 DOF airfoil

The training has produced good results in both cases, with a good fit of the aerodynamic response given by the solver AeroFoam. Both the networks are able to capture the C_M peak at $s \approx 700$, which is a very difficult task, since the wide frequency spectra of the signal. The various computational time required for the training and the order of convergence are given in table 5.11 for the network A and table 5.12 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	82	$\mathcal{O}\left(10^{-1}\right)$
Analytic	71	$\mathcal{O}\left(10^{-2}\right)$
AD	60	$\mathcal{O}\left(10^{-3}\right)$

Table 5.11: Convergence properties, ECTRNN A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	102	$\mathcal{O}\left(10^{-1}\right)$
Analytic	87	$\mathcal{O}\left(10^{-3}\right)$
AD	71	$\mathcal{O}\left(10^{-3}\right)$

Table 5.12: Convergence properties, ECTRNN B

In this case, the training time is greater than the one required in the case of a random training. It is possible this fact is due to a more complex training input, with more and steeper variations than in the previous case, causing a more frequent variation of the network synaptic weights.

C²TRNN

The C²TRNN is trained with the input shown in figure 5.16, and the training results are shown in figure 5.20 and 5.21. The same networks A and B considered previously in the random training are proposed here for the hybrid training.

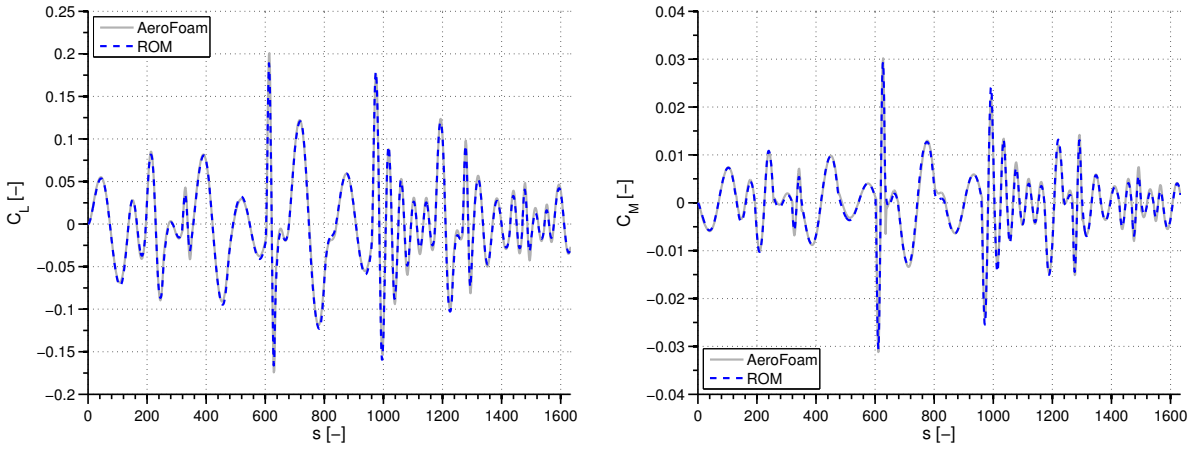


Figure 5.20: C²TRNN A, hybrid training - 1 DOF airfoil

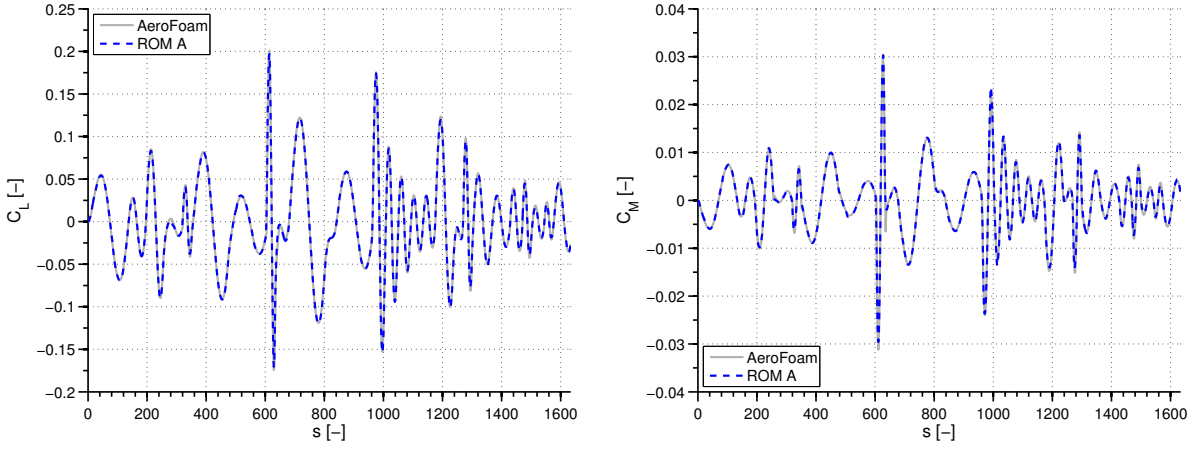


Figure 5.21: C²TRNN B, hybrid training - 1 DOF airfoil

The training has produced good results in both cases, with a good fit of the aerodynamic response given by the solver AeroFoam. The various computational time required for the training and the

order of convergence, are given in table 5.13 for the network A and table 5.14 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	41	$\mathcal{O}(10^{-1})$
Analytic	22	$\mathcal{O}(10^{-2})$
AD	15	$\mathcal{O}(10^{-2})$

Table 5.13: Convergence properties, C^{2}TRNN A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	53	$\mathcal{O}(10^{-1})$
Analytic	30	$\mathcal{O}(10^{-3})$
AD	18	$\mathcal{O}(10^{-3})$

Table 5.14: Convergence properties, C^{2}TRNN B

CTRNN.LI

The CTRNN.LI is trained with the input shown in figure 5.16, and the training results are shown in figure 5.22 and 5.23. The same networks A and B considered previously in the random training are proposed here for the hybrid training.

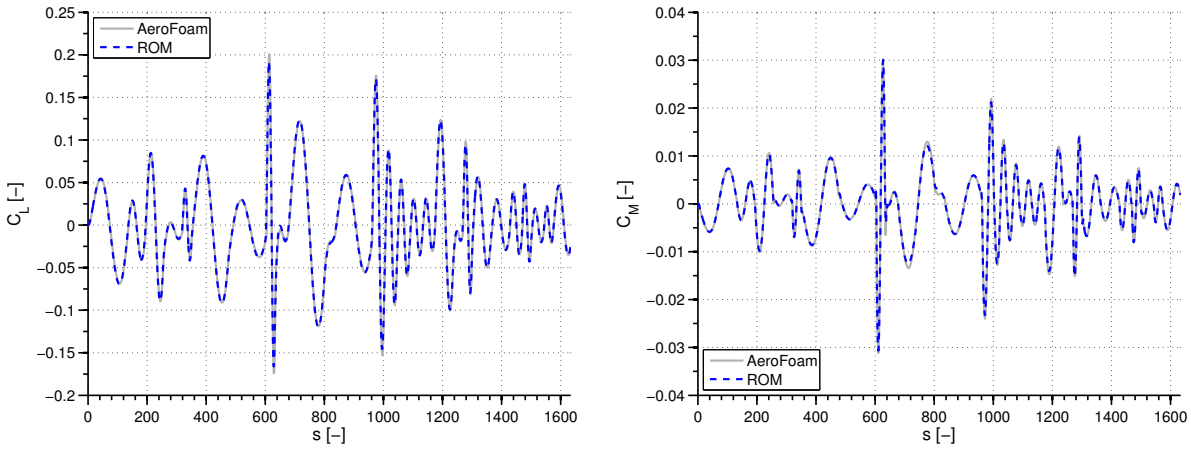


Figure 5.22: CTRNN.LI A, hybrid training - 1 DOF airfoil

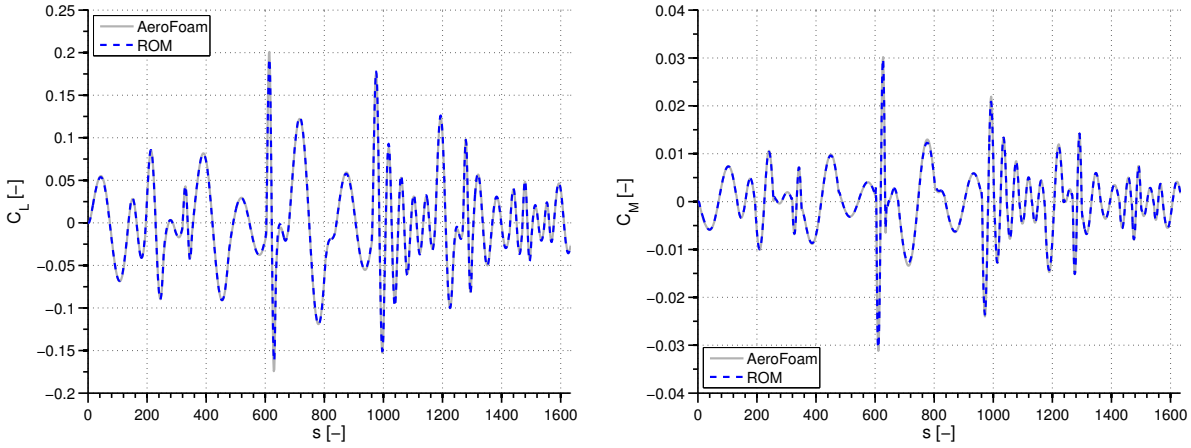


Figure 5.23: CTRNN.LI B, hybrid training - 1 DOF airfoil

The training has produced good results in both cases, with a good fit of the aerodynamic response given by the solver AeroFoam. The various computational time required for the training and the order of convergence, are given in table 5.15 for the network A and table 5.16 for network B.

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	50	$\mathcal{O}\left(10^{-1}\right)$
Analytic	26	$\mathcal{O}\left(10^{-2}\right)$
AD	17	$\mathcal{O}\left(10^{-2}\right)$

Table 5.15: Convergence properties, C^{2}TRNN A

Jacobian matrix computational method	Computational time $[min]$	Converged F
Finite difference	66	$\mathcal{O}\left(10^{-1}\right)$
Analytic	36	$\mathcal{O}\left(10^{-2}\right)$
AD	22	$\mathcal{O}\left(10^{-2}\right)$

Table 5.16: Convergence properties, C^{2}TRNN B

5.3 CTRNN validation

5.3.1 Validation with random training

The trained networks of the previous section are tested now for a two sinusoidally varing input signals, each with different frequency and amplitude. The first test case, called X, is characterized by a reduced frequency of $k_{\mathsf{X}} = 0.25$ and an amplitude of $\bar{\theta}_{\mathsf{X}} = 1\,[deg]$. The second test case, called Y has a reduced frequency of $k_{\mathsf{Y}} = 0.4$ and an amplitude of $\bar{\theta}_{\mathsf{Y}} = 0.7\,[deg]$. The computational time required to simulate the response to these input with the solver AeroFoam for three periods has been equal to $3\,[h]\,0\,[min]$ for the test X and $2\,[h]\,30\,[min]$ for the test Y. For both the CFD simulations, the time step Δt has been fixed to $\Delta t = 10^{-3}$. Thus will be interesting analyze the

validation properties of the different parametrizations trained with input signal discretized with a different time step, in the specific $\Delta t = 3 \cdot 10^{-3}$, thus with a training time step three times greater than the validation time step. Moreover, such validation signals are characterized by a linear response, while during the training phase some system nonlinearity may have been excited. This fact may influence the performance of the networks.

ECTRNN

The results obtained in the validation phase by the ECTRNN are summarized below:

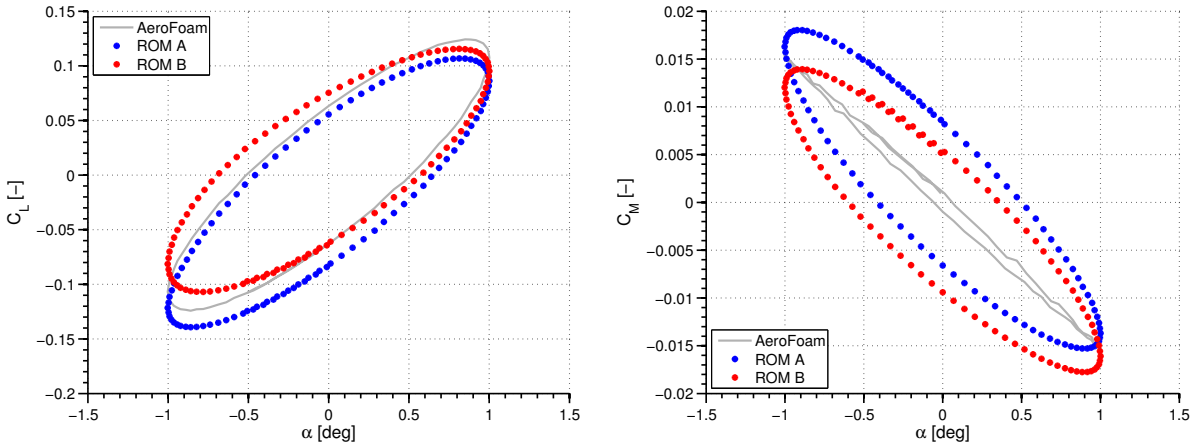


Figure 5.24: ECTRNN , random validation, test X - 1 DOF airfoil

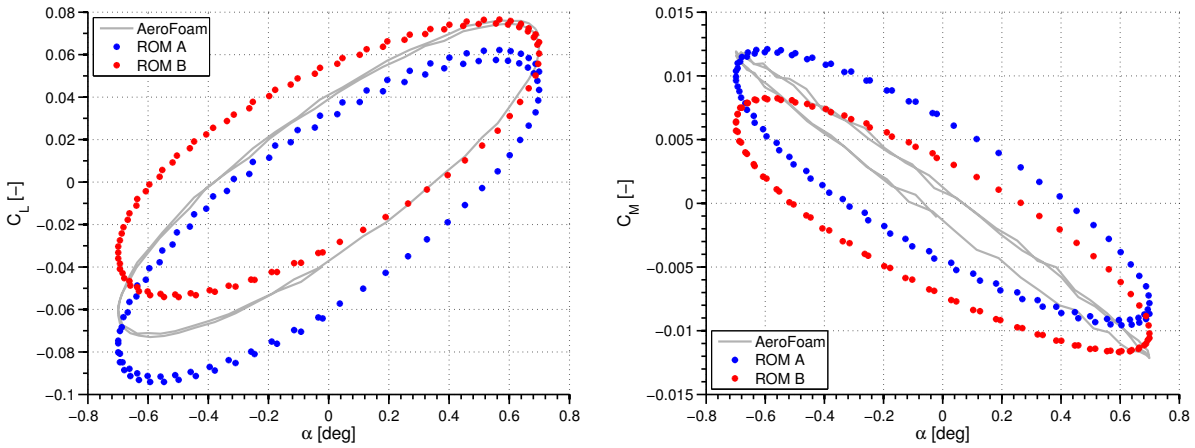


Figure 5.25: ECTRNN , random validation, test Y - 1 DOF airfoil

The generalization properties of this network are not so good as expected. The presence of the *bias* terms may influence the performances of such network. Increasing the network dynamic order does not seem to improve the results, especially for what regard the the test case Y.

C^{2}TRNN

The results obtained in the validation phase by the C^{2}TRNN are summarized below:

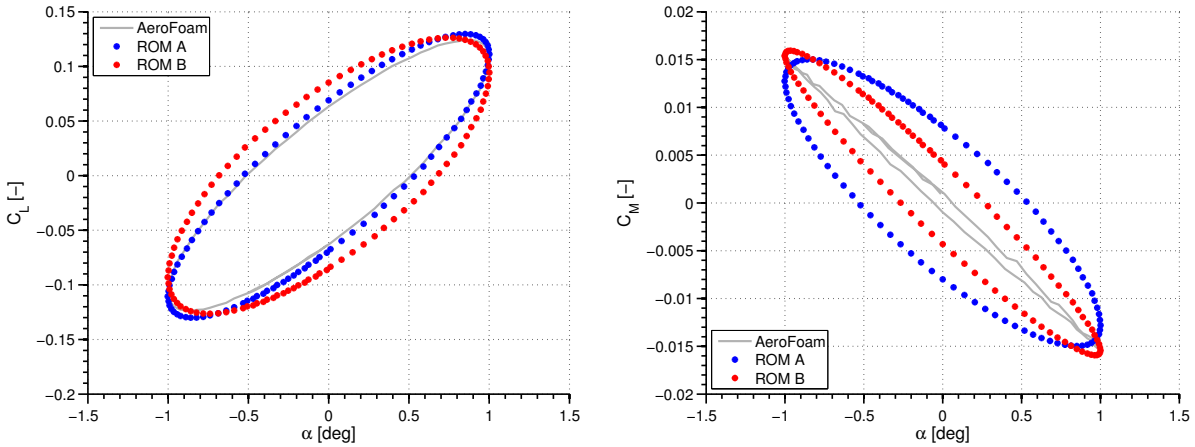


Figure 5.26: C^{2}TRNN , random validation, test X - 1 DOF airfoil

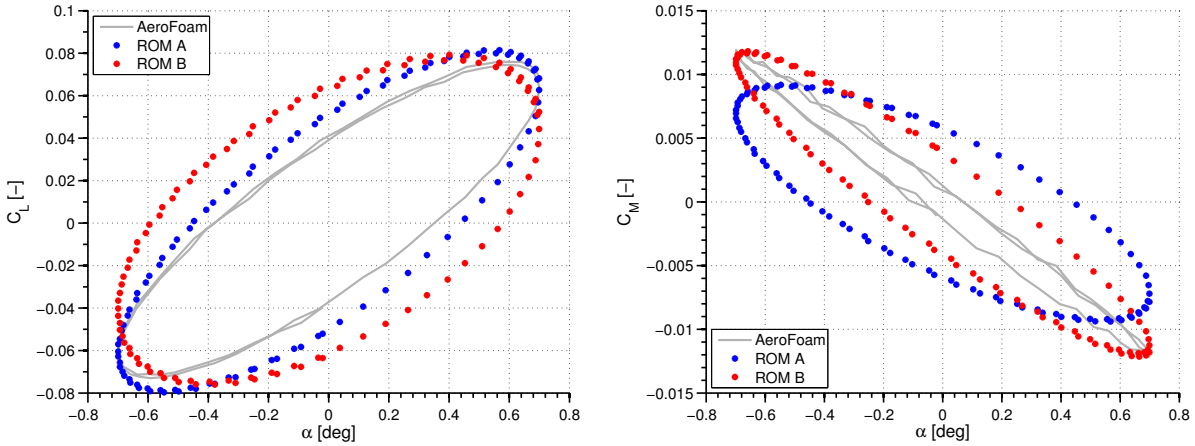


Figure 5.27: C^{2}TRNN , random validation, test Y - 1 DOF airfoil

The fitting is good but not perfect, especially for what regard the C_M. Moreover, it can be noted that as the order of the network is increased, the accuracy with respect of the C_L is decreased, meanwhile the accuracy with respect of the C_M is increased. The computational time required for both simulations with the AD integration procedure is equal to $3\,[s]$.

CTRNN.LI

The results obtained in the validation phase by the CTRNN.LI are summarized below:

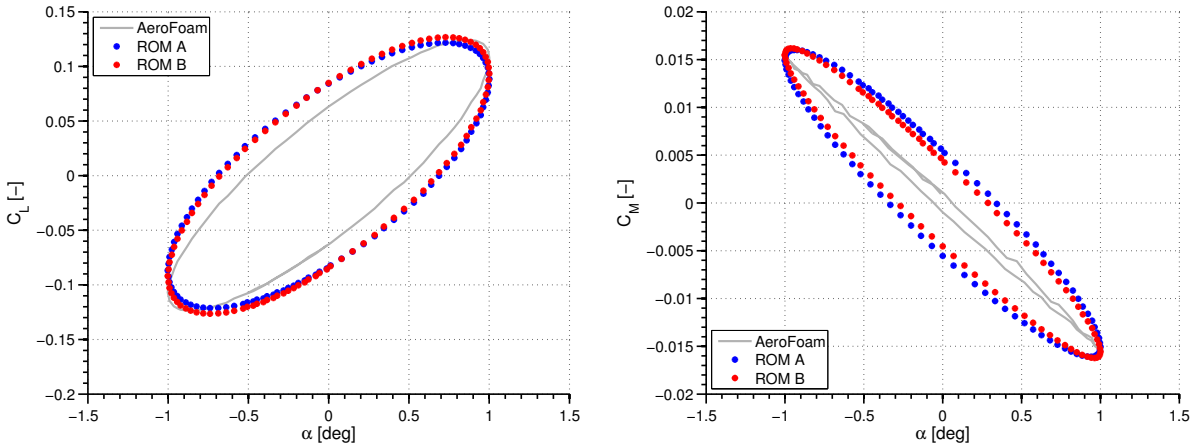


Figure 5.28: CTRNN.LI , random validation, test X - 1 DOF airfoil

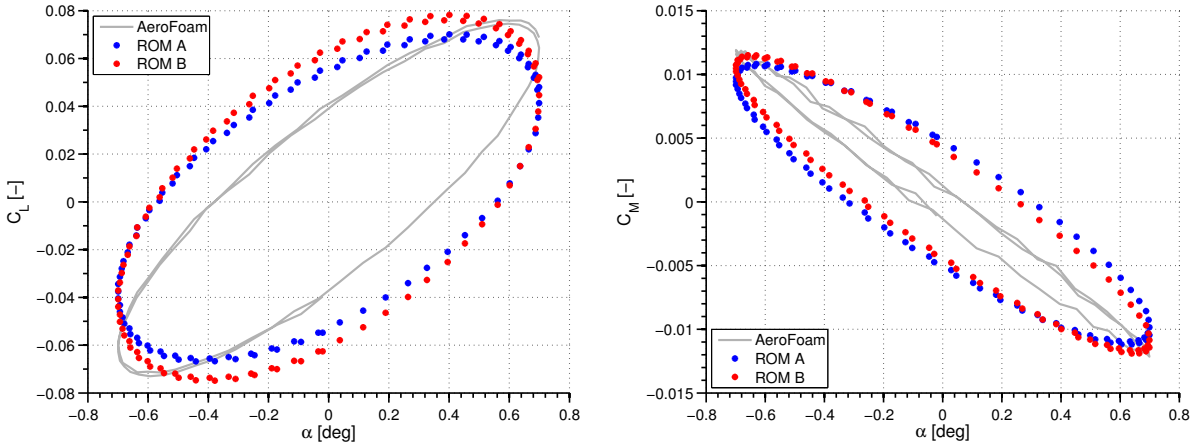


Figure 5.29: CTRNN.LI , random validation, test Y - 1 DOF airfoil

In this case the convergence of the results is as it was expected: incrementing the states of the network, the fit gets better. The generalization results for the C_M can be retained good, a little less can be said for the C_L, where the previous networks have given better results. Moreover, it seems that the CTRNN.LI network weights in the same manner both outputs, scattering the error on both the responses, and not concentrating it in only one output. The time required for the computation of the response has been equal to $3\,[s]$.

5.3.2 Validation with hybrid training

ECTRNN

The results obtained in the validation phase by the ECTRNN after the training with the hybrid signal are summarized in figures 5.30 and 5.31:

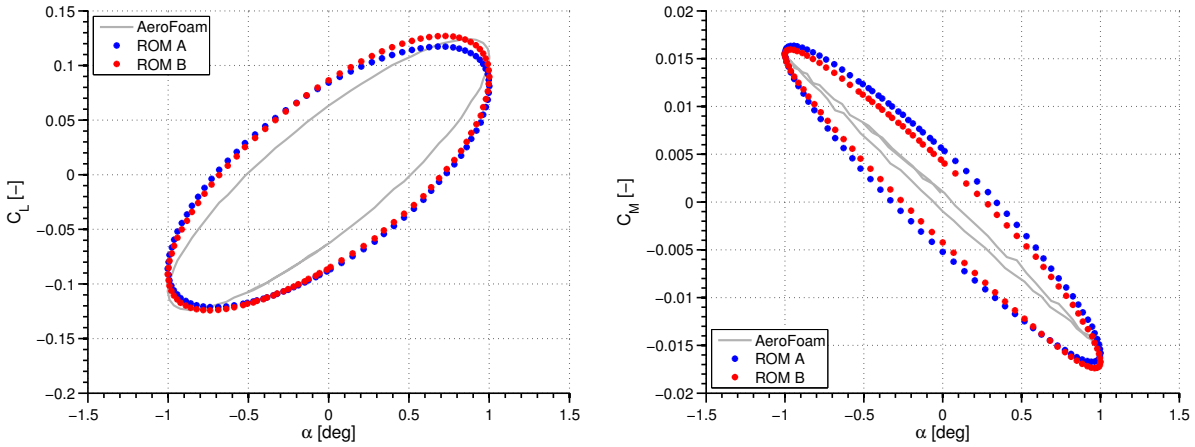


Figure 5.30: ECTRNN , hybrid validation, test X - 1 DOF airfoil

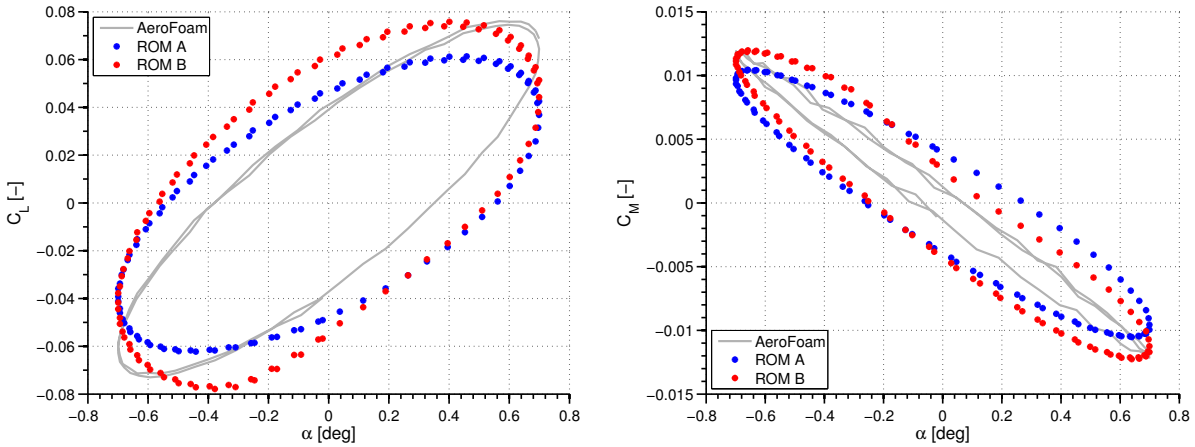


Figure 5.31: ECTRNN , hybrid validation, test Y - 1 DOF airfoil

Increasing the network order during the training comports an augmentation of fidelity in the generalization phase, at least for what regard the mean slope of the C_L and C_M curves and the maximum values. However, both the cycles are again overestimated. This means that the hybrid training does not comport a great gain in the generalization phase with respect of the random training, even if the current training is much more similar to the validation data. The computational time required for both simulations with the AD integration procedure is equal to $3\,[s]$.

C²TRNN

The results obtained in the validation phase by the C²TRNN after the training with the hybrid signal are summarized in figures 5.32 and 5.33:

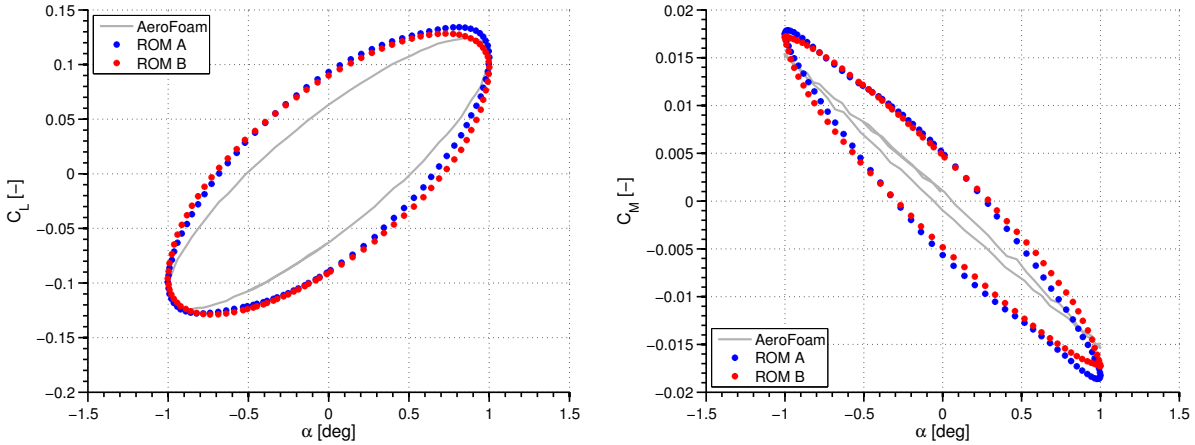


Figure 5.32: C²TRNN , hybrid validation, test X - 1 DOF airfoil

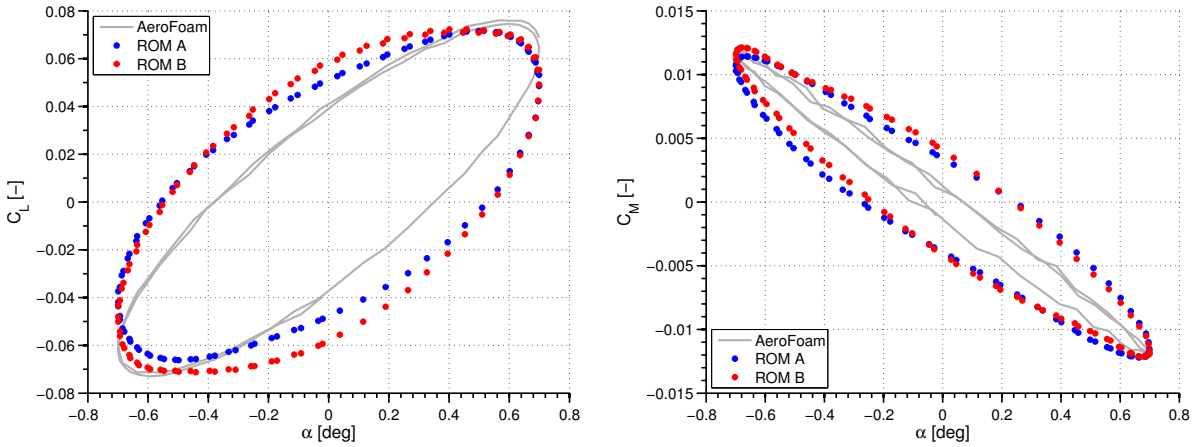


Figure 5.33: C²TRNN , hybrid validation, test Y - 1 DOF airfoil

Increasing the network dynamic order the results get better, especially for what regard the C_M. The results for the C_L are quite the same also with a change of order. The computational time required for both simulations with the AD integration procedure is equal to $3\,[s]$.

CTRNN.LI

The results obtained in the validation phase by the CTRNN.LI after the training with the hybrid signal are summarized in figures 5.34 and 5.35:

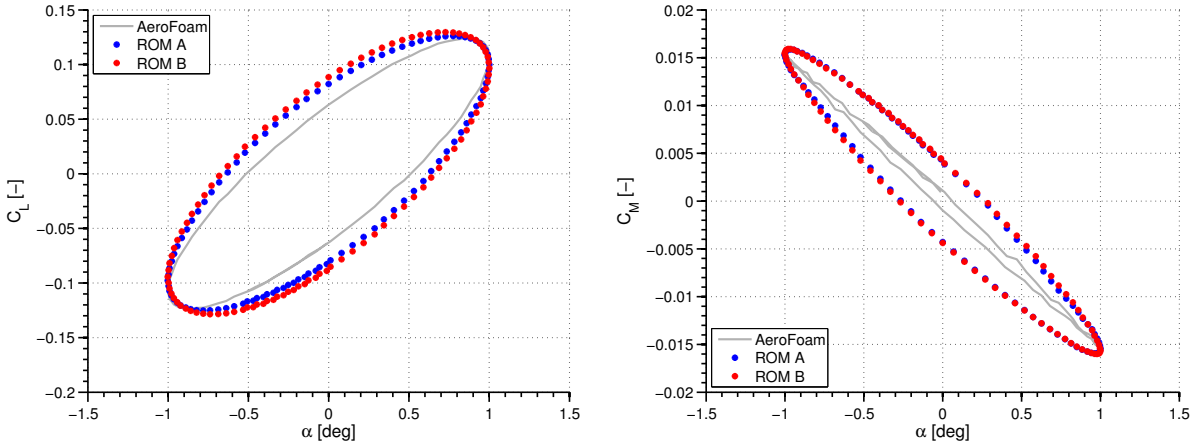


Figure 5.34: CTRNN.LI , hybrid validation, test X - 1 DOF airfoil

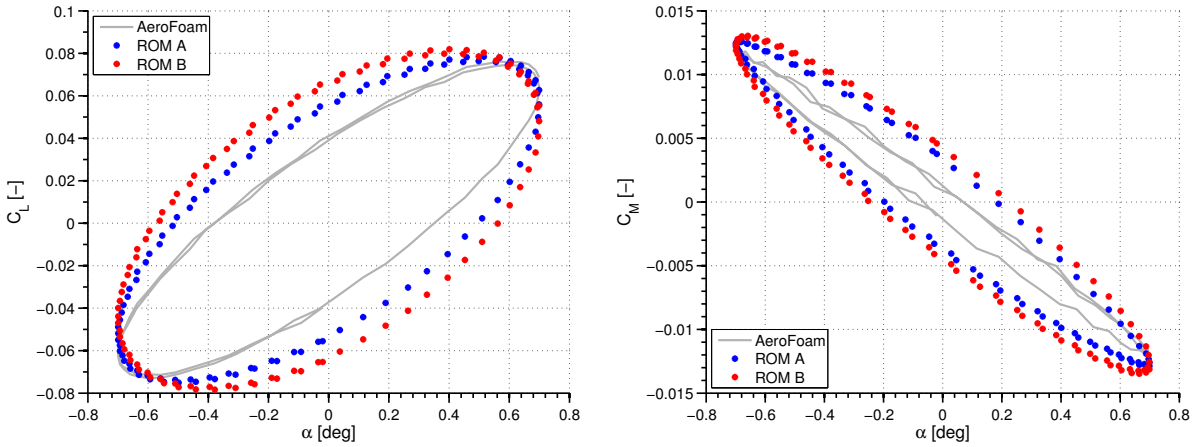


Figure 5.35: CTRNN.LI , hybrid validation, test Y - 1 DOF airfoil

Increasing the network dynamic order does not seem to improve the results. This can mean that four states are sufficient to represent the nonlinear aerodynamic system, even if the generalization performances are not perfect. The computational time required for both simulations with the AD integration procedure is equal to $2\,[s]$.

5.4 Final considerations

In the previous sections the training and generalization performances of the different network parametrizations have been presented and compared. It has been seen that the training has always resulted in very good results, characterized by low error levels. Instead, the generalization performances of all the parametrization has not proved to be perfect, with a constant overestimation of the load cycles, but an improvement of the results has been obtained with an increasing ROM order.

It is however insteresting to see that good results have been obtained with systems of very low order, the maximum order considered has been 6, against the $12200 \cdot 4 = 48800$ states required by the CFD simulation. It has been shown that after the training the computational time required for the determination of the system response has passed from the order of hours to the order of a very few seconds. This is a very simple example, but it shown the big potentialities of a ROM techniques.

Moreover, it is also interesting to analyze the generalization properties of the different networks giving as input the hybrid one for the network trained with the random input and the opposite with the networks trained with the hybrid signal. Let us consider the first case proposed. The results obtained are shown in figures 5.36, 5.37 and 5.38.

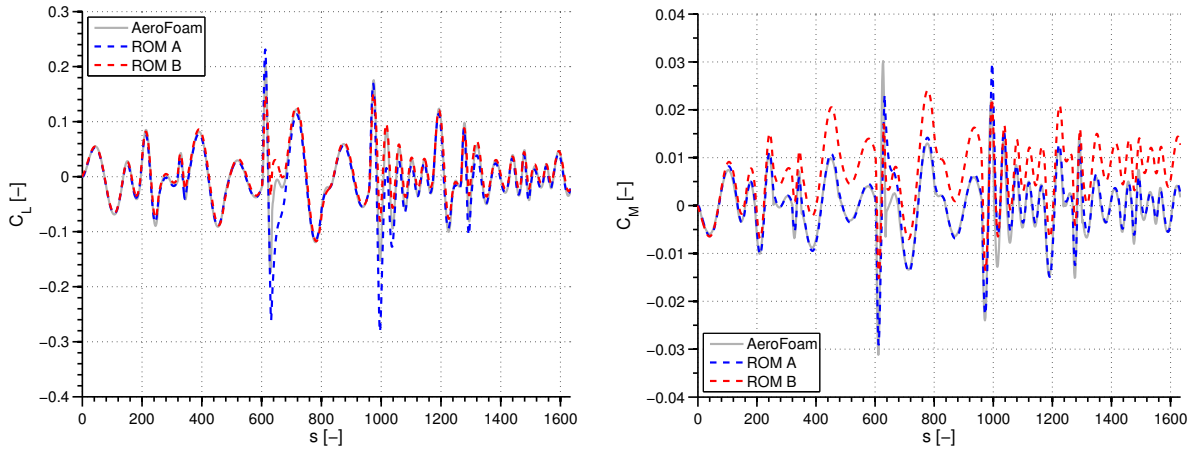


Figure 5.36: ECTRNN , random cross-validation - 1 DOF airfoil

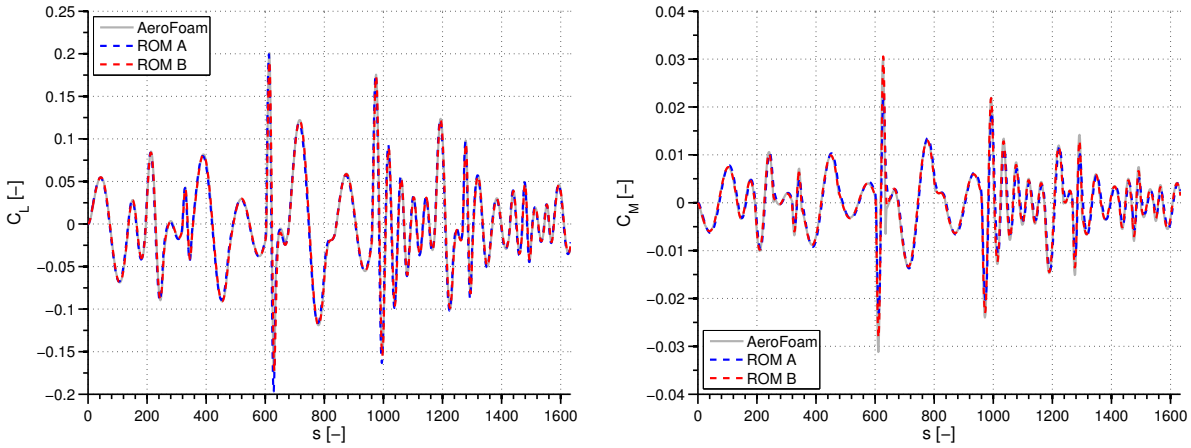


Figure 5.37: C^2TRNN , random cross-validation - 1 DOF airfoil

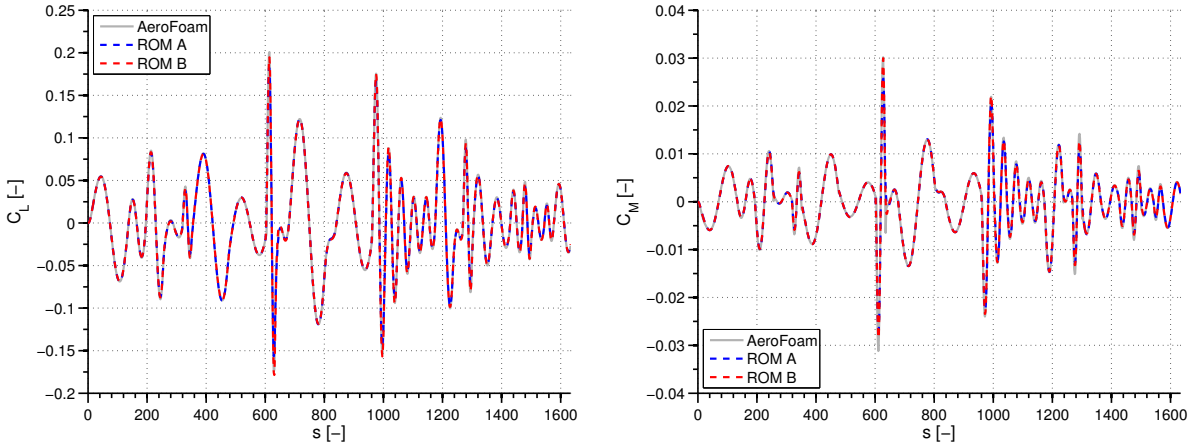


Figure 5.38: CTRNN.LI , random cross-validation - 1 DOF airfoil

The networks trained with the random signal present good generalization properties. It could be expected, since the hybrid signal is a regularized version of the random one, and moreover the frequency spectra covered by the random input is twice the one covered by the hybrid one. The only drawback presented by this type of training is shown by the ECTRNN , in particular in the C_M response. This loss of fidelity can be due to a greater number of states embedded in the ECTRNN B: thus increasing the network dimensions does not always means that the network generalization is improved. Such high order networks could include some neurons that might have been not well trained in the learning phase. Such result may introduce poor or even unstable performances. This fact should be considered in the next chapter.

Let us now consider the other cross-validation: the networks trained with the hybrid signal are tested with the random input, and the relative performances are shown in figures 5.39, 5.40, 5.41, 5.42 and 5.43.

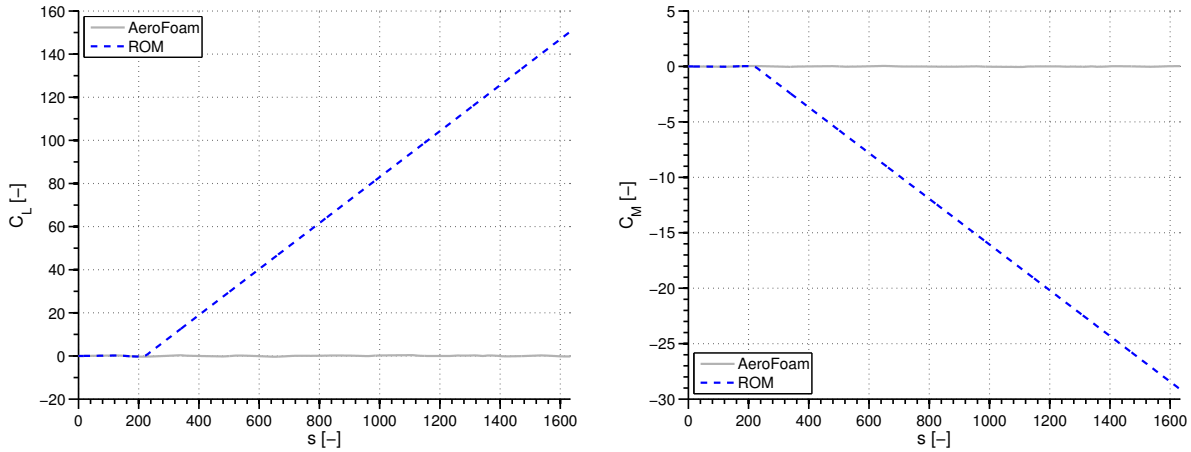


Figure 5.39: ECTRNN A, hybrid cross-validation - 1 DOF airfoil

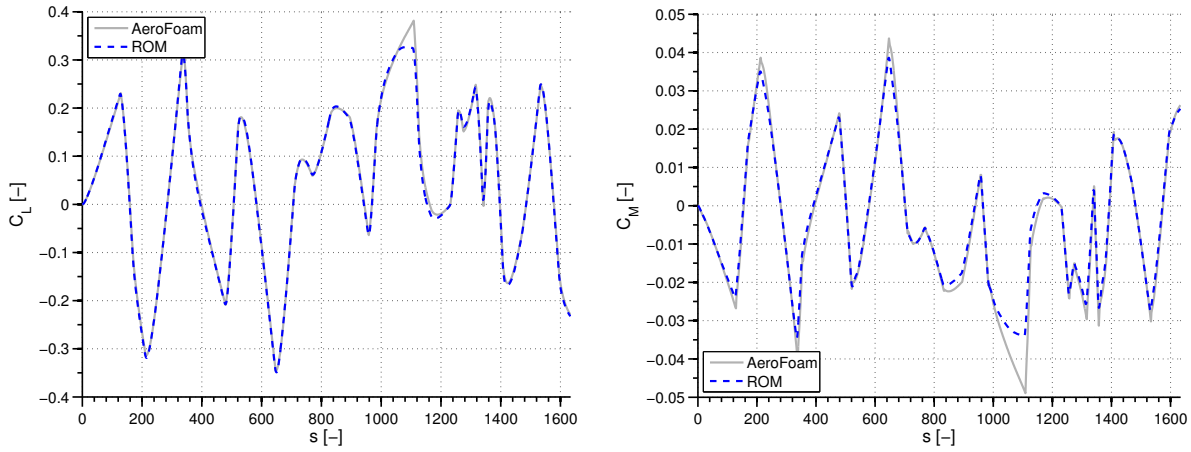


Figure 5.40: ECTRNN B, hybrid cross-validation - 1 DOF airfoil

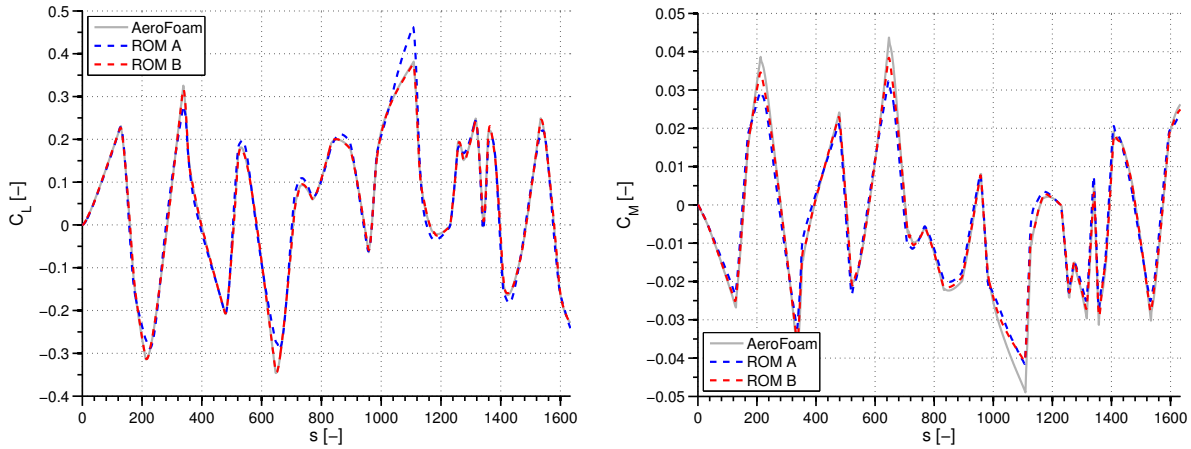


Figure 5.41: C^2TRNN , both networks, hybrid cross-validation - 1 DOF airfoil

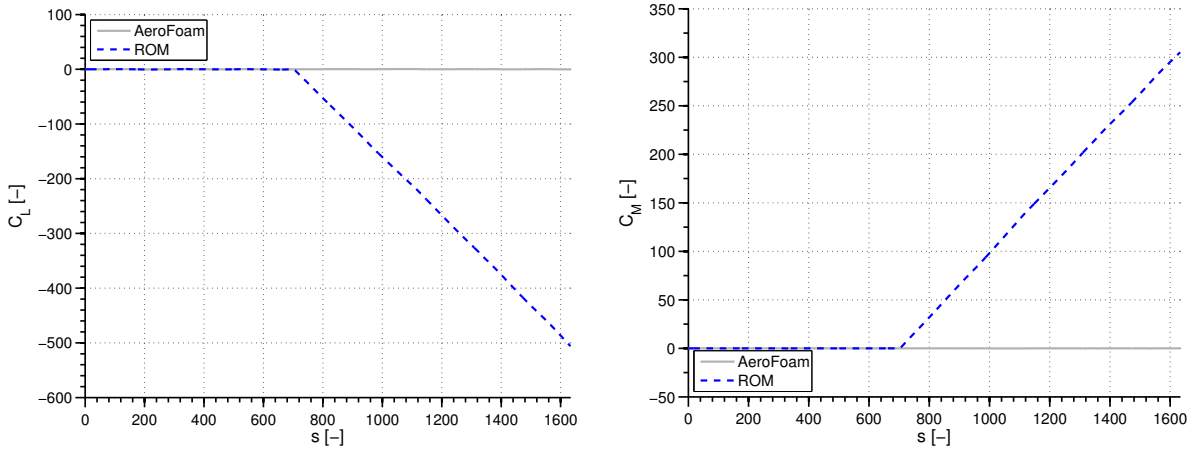


Figure 5.42: CTRNN.LI A, hybrid cross-validation - 1 DOF airfoil

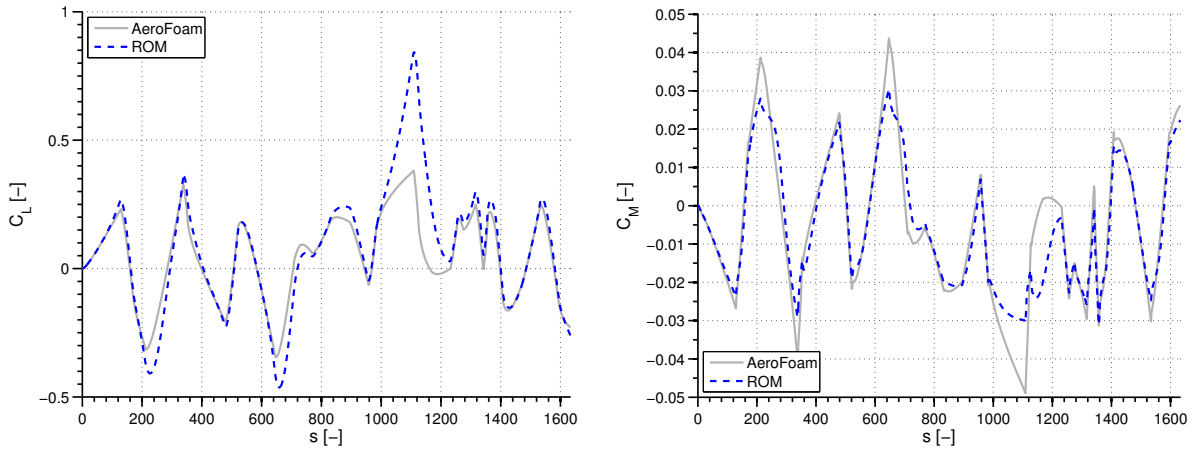


Figure 5.43: CTRNN.LI B, hybrid cross-validation - 1 DOF airfoil

It seems that the C^{2}TRNN gives the best generalization results, since both the networks A and B get a similar response to that given by the CFD simulation by AeroFoam. Instead, for the other networks it has been necessary to plot the validation results on different graphs, since for the low order network of both the parametrizations, the type A, a *saturation* problem occours, and the results obtained are very far from be accurate. This can be due to the different frequency content of the training signal, much more smaller than the frequency content of the random input: this fact can induce the network to work at some frequencies that have been not trained, thus resulting in a poor generalization behaviour. However the saturation problem is solved as the networks dynamic order is increased, thus including a greater number of dynamic states permits to obtain better generalization performances. This can be in contrast with what has been said previously, when it has been stated that is possible, for networks with a large number of states, that some nonlinear eigenvalues are not well trained and the resulting generalization performances may be poor. But poor generalization performances are of course better than no generalization performances, thus some little considerations are required when the order of the network has to be chosen.

From the literature [26], the choice of the model order is tipically carried out by a combination of prior knowledge of the system and a trial and error procedure. The choice of a higher dynamic order of the model increases the dimensionality of the problem and hence its complexity. Often one may be forced to accept significant unmodeled dynamics by choosing a too low order model. The author of [26] suggests that low order models often suffice since the model error is dominated by the approximation error caused by an inaccurate description of the high order system's nonlinearities.

Regarding the C^{2}TRNN , it has been note empirically that chosing a number of network dynamic states equal to the number of output, thus obtaining a square output matrix $\mathbf{W}^c$ is a good first guess for obtaining fair generalization results. Instead, for the other two network parametrizations both the training signal and the network dynamic order should be chosen carefully, maybe starting with a small number of different network orders and then selecting the one that gives the best generalization performances. But when the frequency characteristics of the phenomena that has to be studied is known by a previous analysis, maybe, in the case of a LCO, by a Hopf bifurcation analysis, the frequency content of the training signal can be fixed to an adapt value, removing in practice the saturation problem. Then the accuracy can be improved analyzing different network orders.

Chapter 6

Test cases

In this chapter the neural network based ROM presented in chapter 3 is adopted to build an continuous time aerodynamic model which will be coupled with the dynamics of a mechanical system. The resulting aeroelastic system considered here is a simple 2 DOFs airfoil, linked to a fixed reference frame by a torsional and a linear spring. The 2DOFs airfoil is a widely employed test case both for reduced order modeling, as can be seen in [1, 32, 21, 17, 39, 12, 31, 15] and for unsteady compressible aerodynamic modeling [19, 68]. The particular aerodynamic condition considered is characterized by the presence of strong shocks over the airfoil, and because of its motion, this shocks are subjected to large displacements. The aerodynamic flow field is thus highly nonlinear, therefore there are good possibilities that an LCO can develope during the simulation. The point of stability exchange is tracked by means of an eigenvalue analysis near the homogeneous equilibrium solution.

The present chapter follows mainly the structure of the previous one: first the aerodynamic system is identified by a neural network based ROM through a training phase, then its generalization performances are analyzed. Finally the resulting low-order aerodynamic system is coupled to the structural model in order to obtain different nonlinear aeroelastic responses, studying their dependence on different parameters.

Summing up, in section 6.1.1 the mechanical model considered is presented, introducing the dimensionless variables that are used to compare the results obtained with the ones present in the literature and the main characteristics of the 2 DOFs airfoil. In section 6.1.2 an aerodynamic reduced order model is constructed, first with a training phase, employing a training input of the kind introduced in 3 and then with a validation phase, where the aerodynamic ROM will shown its fidelity characteristics to input never presented in the training phase. All the network parametrizations presented in chapter 3 are considered, analyzing and comparing the different performances. Then, in section 6.1.5 the aerodynamic ROM is coupled with the mechanical model presented in section 6.1.1 in order to perform aeroelastic simulations, analyzing the effect of different parameters on the system response. Particular emphasis will be given to the analysis of LCOs. The performance of the ROM will be compared to the aeroelastic simulation performed with a CFD model employed to represent the aerodynamic system, with both accuracy and computational time being considered. Moreover, both the methods described in 2.1.1 and 2.1.2 will be employed in sections 6.1.6 and 6.1.7 rispectively, enhancing the ability of the ROM in capturing the LCO behaviour. Finally, in section 6.3 some considerations about the results obtained are made, discussing some features of the proposed ROM in this particular application.

6.1 Two degrees of freedom typical section

6.1.1 Mechanical model

The 2 DOFs airfoil is again a *NACA 64A010*, and a schematic representation of such model is given in figure 6.1. The two DOFss considered are the vertical plunge h, positive upward, and

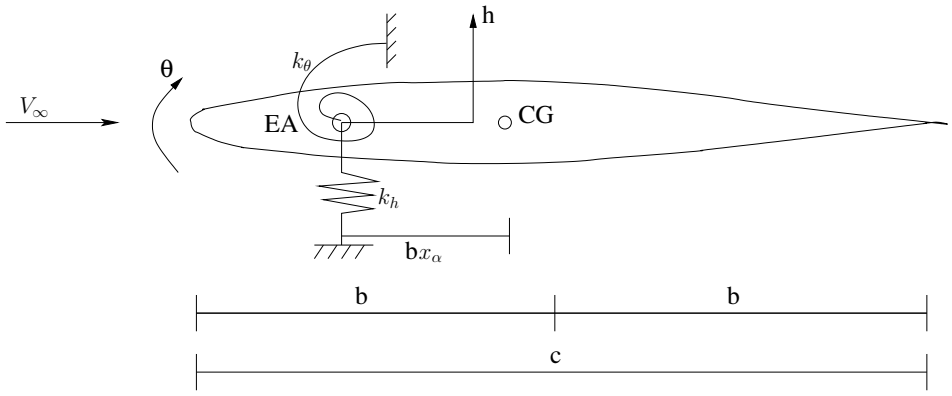


Figure 6.1: 2 DOFs airfoil

the rotational pitch θ, positive clock-wise. The airfoil is connected to a fixed reference frame by a linear and torsional springs. The position of the Elastic Axis (EA) is taken as in the previous chapter at the 20% of the airfoil chord, meanwhile the position of the Center of Gravity (CG) is determined by the value of the variable x_α. The variable b represent the half chord and it is employed for non-dimensionalising the dynamic system.

The related equations of motion are:

$$\begin{bmatrix} m & S_\theta \\ S_\theta & J_\theta \end{bmatrix} \begin{pmatrix} \ddot{h} \\ \ddot{\theta} \end{pmatrix} + \begin{bmatrix} k_h & 0 \\ 0 & k_\theta \end{bmatrix} \begin{pmatrix} h \\ \theta \end{pmatrix} = \begin{pmatrix} F_h \\ M_\theta \end{pmatrix} \tag{6.1}$$

The force and moment acting on the airfoil are the lift and the aerodynamic moment produced by the interaction of the flow with the airfoil itself. The modeling of the aerodynamic load is left to the next section, where a neural network based ROM is employed in order to extract the main nonlinear behaviour from the high-order system represented by the CFD modeling.

The two characteristic circular frequencies ω_h and ω_θ are defined as:

$$\omega_h = \sqrt{\frac{k_h}{m}} \qquad \omega_\theta = \sqrt{\frac{k_\theta}{J_\theta}} \tag{6.2}$$

Introducing the following adimensional variables:

$$\begin{aligned} \mu &= \frac{m}{\pi \rho_\infty b^2} && \text{Fluid-to-mass ratio} \\ V^* &= \frac{V_\infty}{\omega_\theta b \sqrt{\mu}} && \text{Reduced velocity} \end{aligned} \tag{6.3}$$

Introducing an adimensional time $\tau = \omega_\theta t$, the dynamic system 6.1 can be rewritten in adimensional form:

$$\begin{bmatrix} 1 & x_\theta \\ x_\theta & r_\theta^2 \end{bmatrix} \begin{pmatrix} (h/b)'' \\ \theta'' \end{pmatrix} + \begin{bmatrix} (\omega_h/\omega_\theta)^2 & 0 \\ 0 & r_\theta^2 \end{bmatrix} \begin{pmatrix} h/b \\ \theta \end{pmatrix} = \frac{(V^*)^2}{\pi} \begin{pmatrix} C_L \\ 2\,C_{M_{EA}} \end{pmatrix} \tag{6.4}$$

Where:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} &= (\cdot)' \\ x_\theta &= \frac{S_\theta}{mb} \\ r_\theta &= \sqrt{\frac{J_\theta}{mb^2}} \end{aligned} \tag{6.5}$$

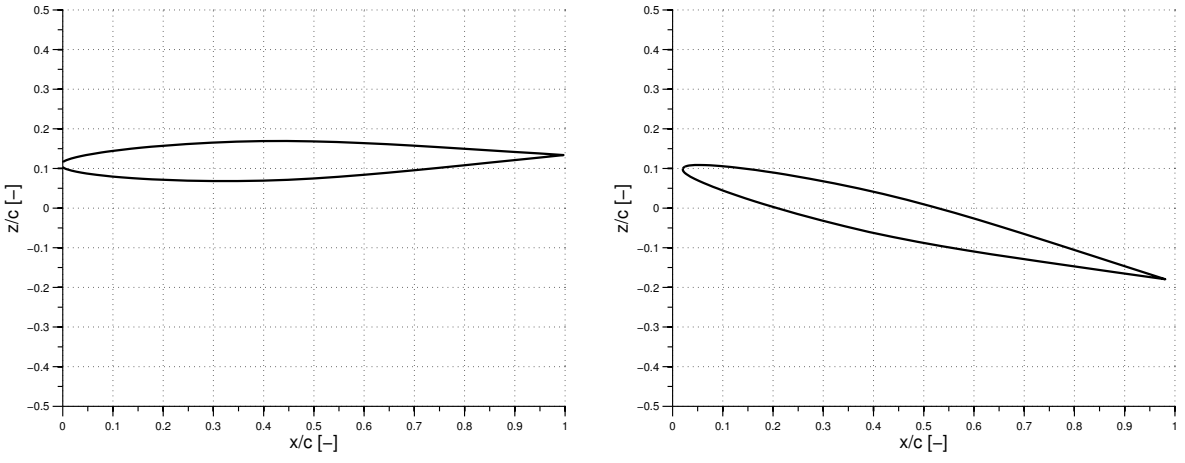
and $C_{M_{EA}}$ is the aerodynamic moment coefficient computed about the elastic axis.

The test case considered is characterized by the following values:

M_∞	0.8
x_θ	0.25
r_θ^2	0.75
ω_h/ω_θ	0.5
μ	75
ρ_∞	$1.225\,[kg\,m^{-3}]$
T_∞	$288.15\,[K]$

Table 6.1: 2 DOFs airfoil data

Since the asymptic value of the airflow speed will be considered fixed, the stiffness characteristics of the mechanical system can be varied along with the value of the reduced velocity, according to relation 6.3. For example, for $V^* = 0.739$, a value that will be considered in the validation phase of the aeroelastic system, the natural frequencies of the system are $f_h = 6.68\,[Hz]$ and $f_\theta = 14.35\,[Hz]$. The normal modes of such system are shown in figure 6.2.



(a) Mode 1 (b) Mode 2

Figure 6.2: No wind normal modes of the 2 DOFs airfoil, $V^* = 0.739$

Clearly, the first mode is mainly plunging, meanwhile the second is mainly pitching. The two movements are coupled because of the presence of the static unbalance x_θ.

At this point, an aerodynamic model is needed, to compute the aeroelastic response of the system. The CTRNN will be employed to represent such a system and the different parametrizations proposed in chapter 3 will be considered in the following sections.

6.1.2 Aerodynamic reduced order modeling

In AeroFoam, the 2 DOFs airfoil represented in figure 6.1 is forced in rigid plunge and pitch with the training input designed in chapter 3. Thus, the CFD simulation represent a purely unsteady aerodynamic problem: no mass and stiffness characteristics of the aeroelastic system are specified and the airfoil is treated like a rigid body with plunge and pitch DOFs. A schematic representation of the aerodynamic simulation is given in figure 6.3.

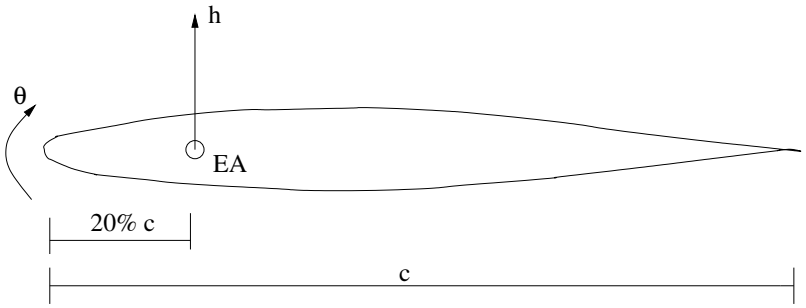


Figure 6.3: 2 DOFs airfoil, model employed in the CFD simulation

The computational mesh is the same of section 5.1, since good results have been obtained with the 1 DOF experiment.

In this chapter the aerodynamic system to be identified is characterized by two input and two output, thus the identification of a Multi Input Multi Output (MIMO) is considered in this section, and in figure 6.4 the schematic role of the CTRNN as aerodynamic ROM model is shown.

Figure 6.4: Work of the CTRNN in the 2 DOFs airfoil case

The unsteady computation is set in AeroFoam essentially with the same setting of the steady problem, with only small differences, as explained in the follow. The file `.dat` containing the arbitrary input signal is stored in `TestCase/Data`. The time solver method chosen in a the Dual Time Stepping (DTS) technique, coupled with a FAS MG strategy. In the specific, the integration method which operates in the pseudo-time domain is 1st order accurate, the threshold residual for obtaining the convergence is fixed to $2.5 \cdot 10^{-3}$ and a maximum of 300 iterations are allowed before stopping the iterative procedure and consider the actual time step converged. The physical time step considered is equal to $\Delta t = 2 \cdot 10^{-3}\,[s]$ and the CFL number chosen is equal to 2.5. The time-marching integration scheme employed is a RK5. The computational experiment in performed with the same settings of section 5.2.

Since the network output C_L and C_M have tipically different orders of magnitude, for example $C_L \sim \mathcal{O}\left(10^{-1}\right)$ and $C_M \sim \mathcal{O}\left(10^{-2}\right)$ as the airfoil pitch and the plunge velocity are small, in order to improve the convergence of the training algorithm, the output have been normalized by their maximum value obtained with the CFD simulation, as in section 5.2. Moreover, in this case, since two input with different order of amplitude are employed, i.e. typical plunge values are in the order of $\mathcal{O}\left(10^{-1}\right)$, meanwhile typical pitch values are in the order of $\mathcal{O}\left(10^{-2}\right)$, since they

are expressed in radians, the input have been normalized by their maximum value. Therefore, defining a normalization input matrix $\mathbf{W}_i$, which is a diagonal matrix such that:

$$W_{i_{ii}} = \frac{1}{\max_{t \in T_{\text{sim}}} u_i(t)} \qquad i = 1, \dots, \text{N}_{\text{input}} \tag{6.6}$$

The training algorithm employs an input matrix $\hat{\mathbf{W}}^b$ which is defined as:

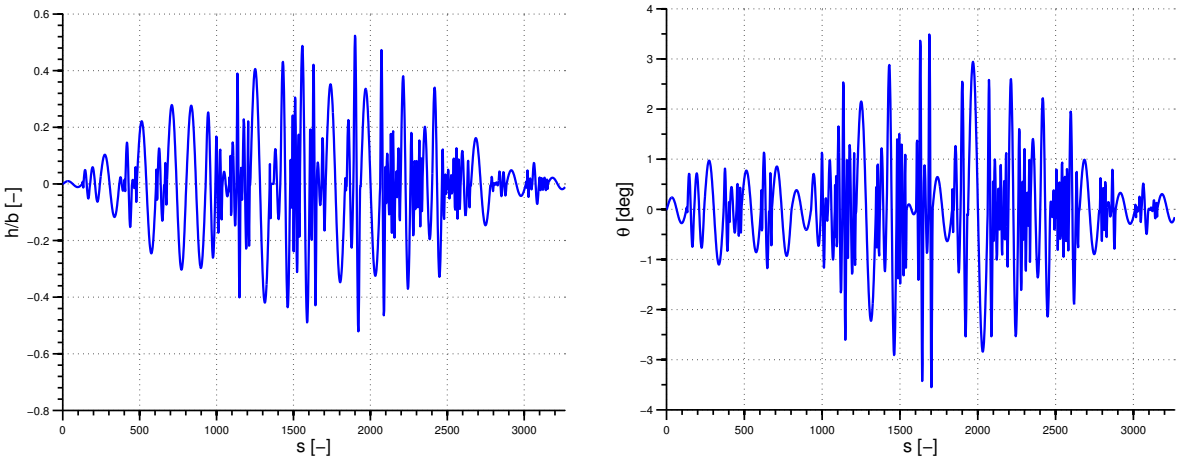
$$\hat{\mathbf{W}}^b = \mathbf{W}^b \mathbf{W}_i \tag{6.7}$$

Thus the different input will be weighted by the algorithm in the same manner. With a little abuse of notation, in the following parts of this chapter, the normalized input and output matrices will be written again as $\mathbf{W}^b$ and $\mathbf{W}^c$, with the aim of not making the mathematical notation too heavy.

An hybrid signal with a wide frequency content has been chosen as training input from chapter 3, since the final scope of this analysis is to predict with the aerodynamic ROM the unsteady aerodynamic loads acting on the airfoil during a LCO. Thus no consideration will be given to input signals that contain steady input. Moreover this kind of signal is assumed to be equivalent to the random signal since it can be interpreted as a regularized version of it. This training signal is characterized by a frequency spectra that should permit the CTRNN to infer whatever unsteady input will be presented to it in the validation phase. The training signal is given as input to the three CTRNN possible parametrization presented in chapter 3, with a final comparison of the relative performances. In this case, only one CTRNN model order will be selected for training, generalization and the aeroelastic simulation phases.

6.1.3 Training phase

With the training signal chosen, the computational time required for computing the aerodynamic response with a CFD simulation has been $24\,[h]\,1\,[min]$. The resulting signal and frequency content are reported in figure 6.5 and 6.6.



(a) $h(s)$ (b) $\theta(s)$

Figure 6.5: Hybrid input for the 2 DOFs airfoil

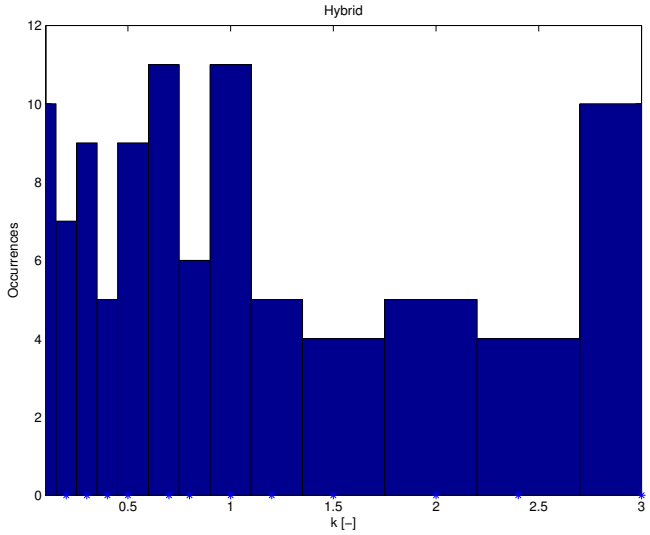


Figure 6.6: Frequency characteristics of the input signal

The training signal is composed by $N_t = 3500$ samples. The frequency spectra of the training input covers a fairly large band, in order to capture all the possible phenomena of interest. In the sections below the different parametrizations presented in chapter 3 are trained considering three kind of training: a training where the Jacobian matrix needed in the LM algorithm is computed by finite difference, one where the Jacobian matrix is composed by its analytical expression and finally the AD training is considered. A comparison of computational time and convergence precision is discussed for each type of parametrization. The settings of the optimization algorithm are the same of section 5.2.1.

ECTRNN

The ECTRNN is trained with the input signal shown in figure 6.5. Considering the results obtained in the previous chapter, the network dimensions are fixed to a number of states $n_x = 3$, a number of hidden neurons $n_h = 5$, therefore a total number of unknowns of $n_\theta = 47$. The results obtained from the training are shown in figure 6.7.

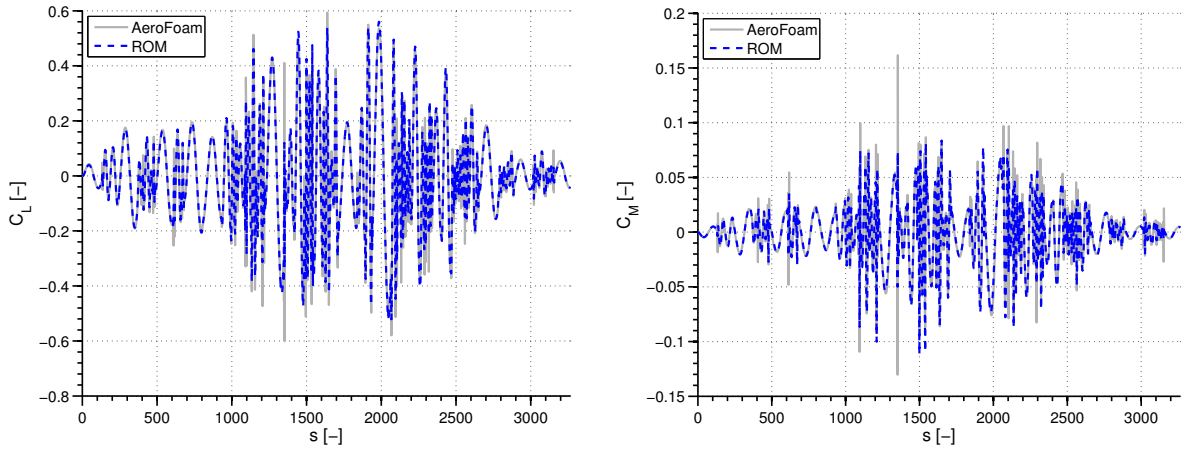


Figure 6.7: ECTRNN , training - 2 DOFs airfoil

Very good results have been obtained from the training phase, capturing most of the relevant peaks in the response. It should be noted that in chapter 5 this kind of network has also shown very good training properties. The computational time and the level of convergence are shown in 6.2.

Jacobian matrix computational method	Computational time	Converged F
Finite difference	Not converged	[–]
Analytic	7 [h] 28 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	6 [h] 58 [min]	$\mathcal{O}\left(10^{-1}\right)$

Table 6.2: Convergence properties, ECTRNN

The adoption of the AD training technique permits to reduce considerably the convergence time. The finite difference approach has been proved again inefficient, with the loss of the convergence properties of the training algorithm. This fact can be used as an hint for further analysis or in general for the application of optimization algorithms: when the parametrization of a nonlinear system is simple enough, the computation of the analytic Jacobian matrix or the adoption of more sophisticated techniques like AD is higly preferable than the mere application of finite difference techniques in the computation of it, especially when the system is characterized by strong nonlinearities.

C^{2}TRNN

The C^{2}TRNN is trained with the input signal shown in figure 6.5. The same dimensions of the ECTRNN employed previously are assumed here, resulting in a total number of unknowns of $n_\theta = 40$. The results obtained from the training are shown in figure 6.8.

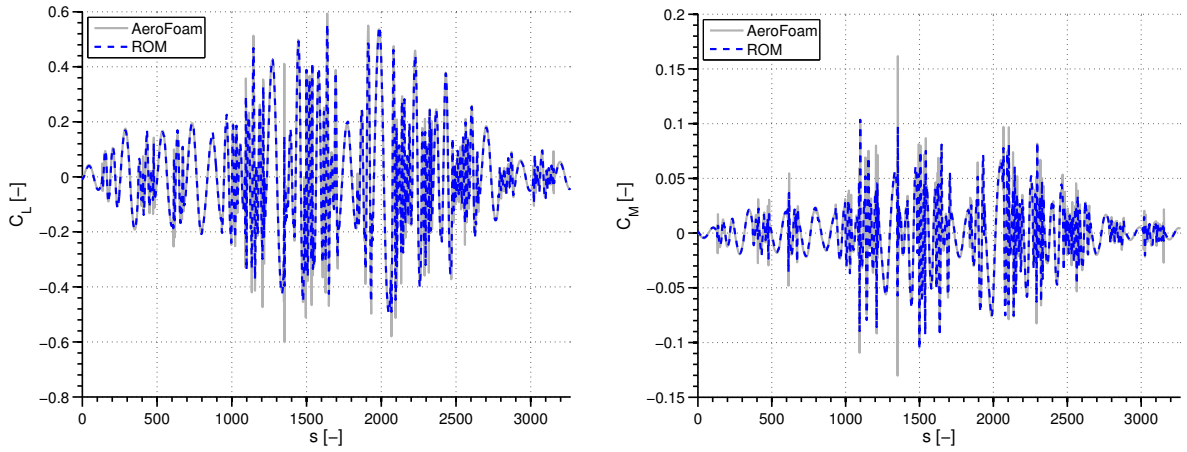


Figure 6.8: C^{2}TRNN , training - 2 DOFs airfoil

Very good results have been obtained also from the training phase. It should be noted that the unresolved peaks occour at a reduced frequency which is very high, that will never be reached during the LCO simulation. However, the resulting neural model presents a very good fit for small reduced frequency, as can be seen from 6.8, and this can be interpreted as a good results in view of the aeroelastic simulation. The computational time and the level of convergence are shown in 6.3.

Jacobian matrix computational method	Computational time	Converged F
Finite difference	Not converged	[–]
Analytic	6 [h] 39 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	6 [h] 01 [min]	$\mathcal{O}\left(10^{-1}\right)$

Table 6.3: Convergence properties, C^{2}TRNN

The adoption of the AD training technique permits again to reduce considerably the convergence time. A faster training has been obtained with respect of the one occourred to the ECTRNN , of course this is due to the smaller number of unknowns that have to be computed by the training algorithm.

CTRNN.LI

The CTRNN.LI is trained with the input signal shown in figure 6.5. In this case the dimensions selected are $n_x = 6$, resulting in a total number of unknowns equal to $n_\theta = 48$. The results obtained from the training are shown in figure 6.9.

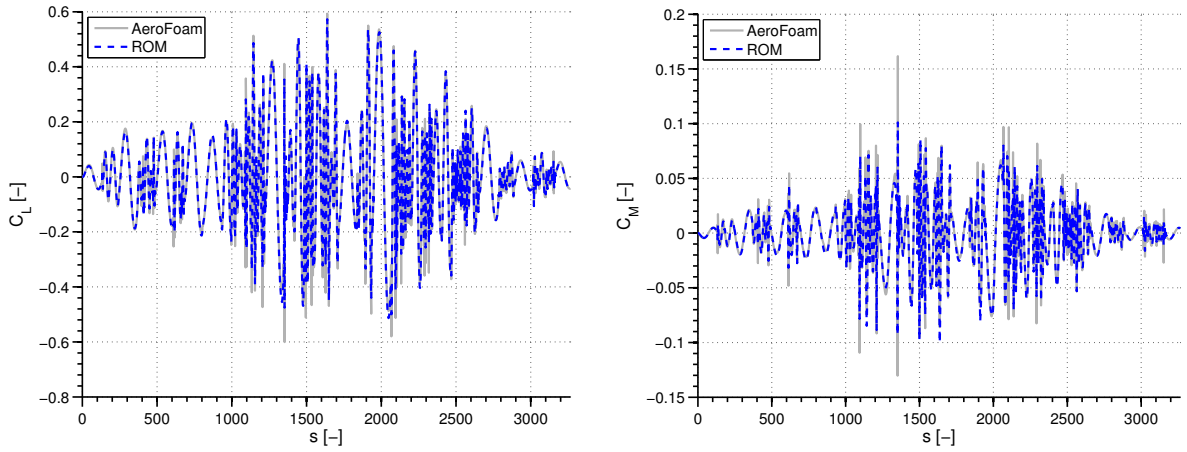


Figure 6.9: CTRNN.LI , training - 2 DOFs airfoil

A good fit of the CFD dynamic response has been obtained with such kind of network. The computational time and the level of convergence are shown in 6.4.

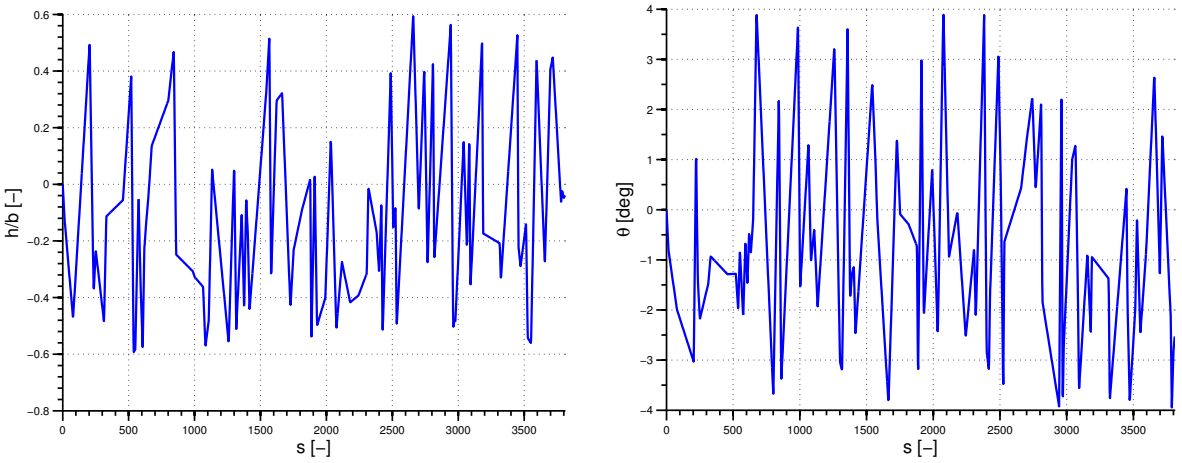
Jacobian matrix computational method	Computational time	Converged F
Finite difference	Not converged	[–]
Analytic	7 [h] 53 [min]	$\mathcal{O}\left(10^{-1}\right)$
AD	7 [h] 14 [min]	$\mathcal{O}\left(10^{-1}\right)$

Table 6.4: Convergence properties, CTRNN.LI

The adoption of the AD training technique permits again to reduce considerably the convergence time. A faster training has been obtained with respect of the one occourred to the ECTRNN , of course this is due to the smaller number of unknowns that have to be computed by the training algorithm.

6.1.4 Validation phase

The input signals selected for the generalization phase are a random signal and an hybrid signal, both described in chapter 3. The frequency content of such signals is limited with respect of the one employed in the training phase. In what follows, the random signal is called signal X, meanwhile the hybrid signal is called signal Y. The computational time required for the CFD simulations has been recorder to $24\,[h]\,51\,[min]$ for signal X and $26\,[h]\,02\,[min]$ for signal Y. These signals and their relative frequency content is shown in figures 6.10, 6.11, 6.12 and 6.13.



(a) $h(s)$ (b) $\theta(s)$

Figure 6.10: Test input X for the 2 DOFs airfoil

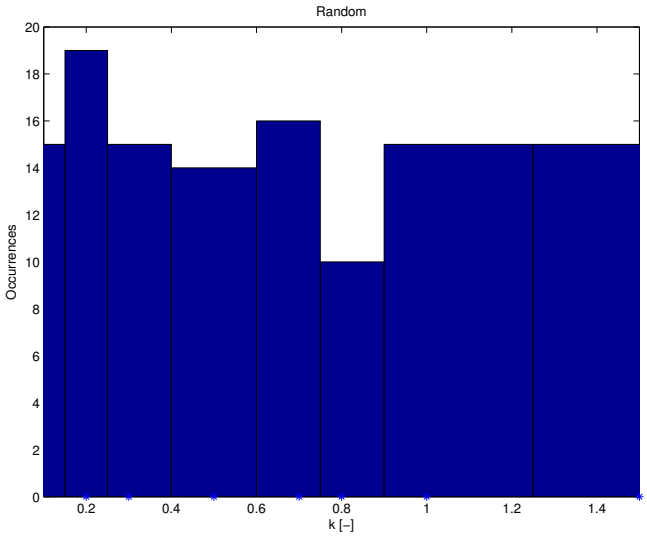
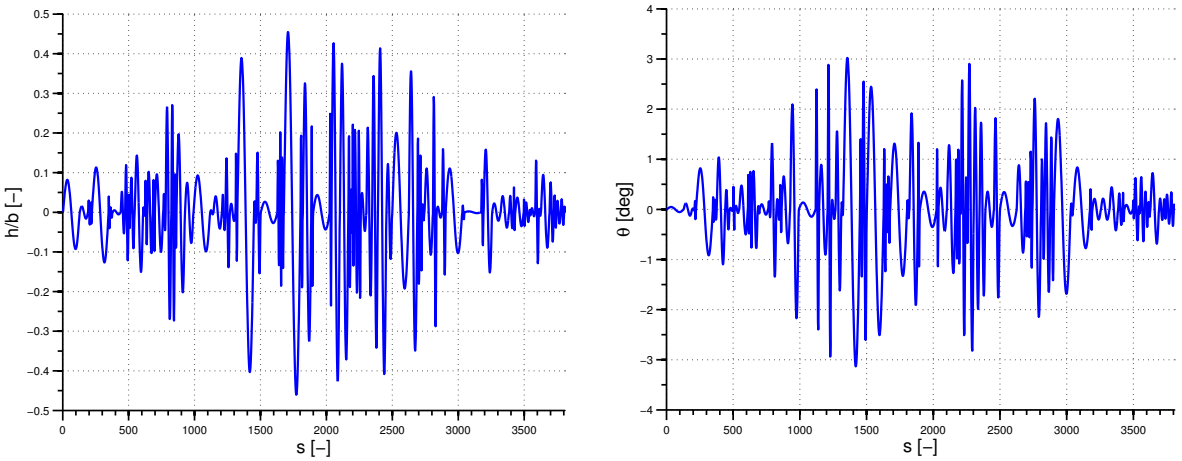


Figure 6.11: Frequency characteristics of the test signal X



(a) $h(s)$ (b) $\theta(s)$

Figure 6.12: Test input Y for the 2 DOFs airfoil

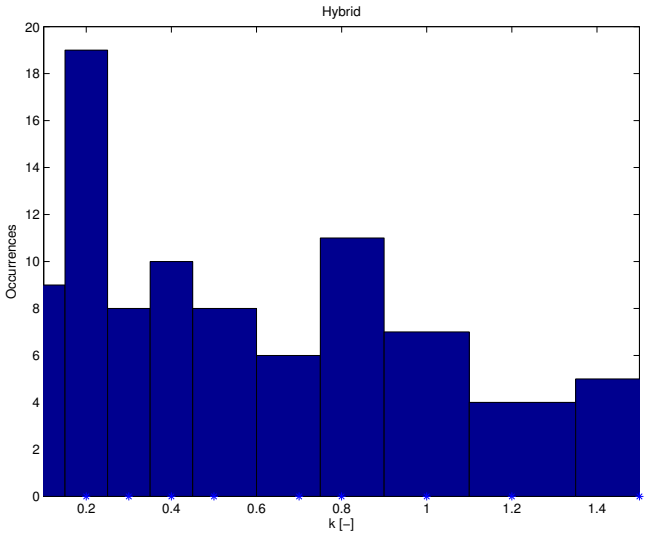


Figure 6.13: Frequency characteristics of the test signal Y

The previously trained networks should be able to generalized such input signals, since the maximum frequency of both is fixed to $k = 1.5$, the half of the maximum one allowed in the training signal.

ECTRNN

The generalization results obtained with the trained ECTRNN are reported in figures 6.14 and 6.15.

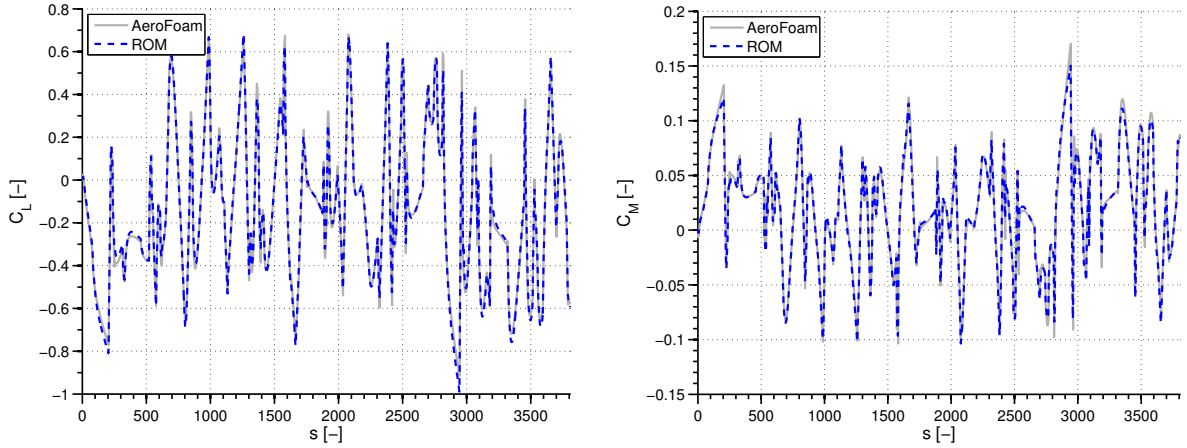


Figure 6.14: ECTRNN , test X - 2 DOFs airfoil

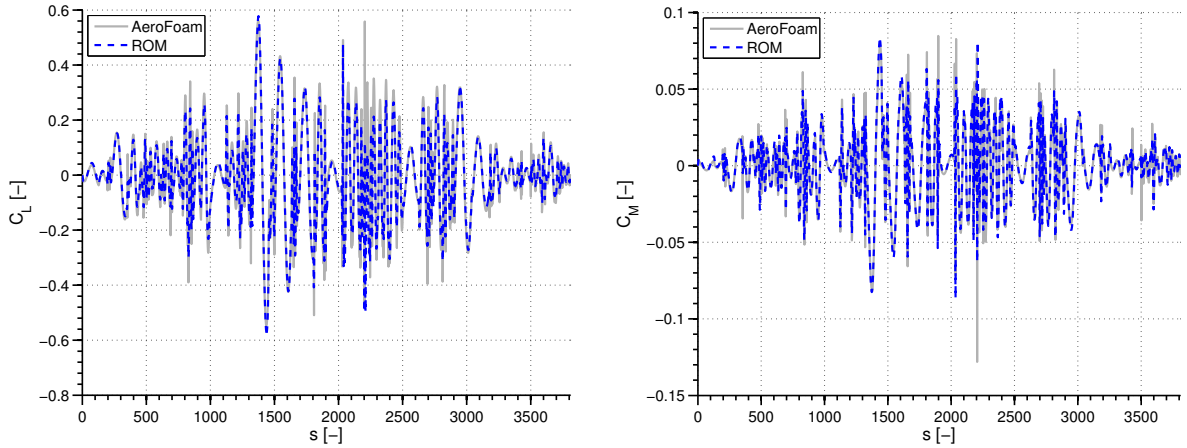


Figure 6.15: ECTRNN , test Y - 2 DOFs airfoil

In both cases, the resulting response presents some kind of numerical instability, it can be seen in test X in various points, where the network response is strongly oscillating and in test Y, near $s = 2250$ the peak of the response is overestimated. However, such network seems work well, with good generalization properties and a small error over the whole simulation interval. The computational time required for such responses are of $4\,[s]$ each.

C^{2}TRNN

The generalization results obtained with the trained C^{2}TRNN are reported in figures 6.16 and 6.17.

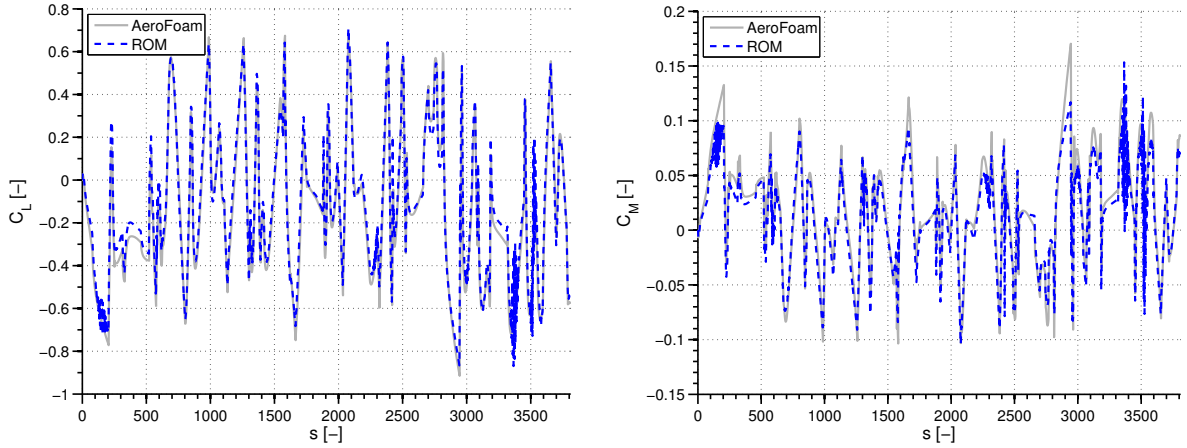


Figure 6.16: C^{2}TRNN , test X - 2 DOFs airfoil

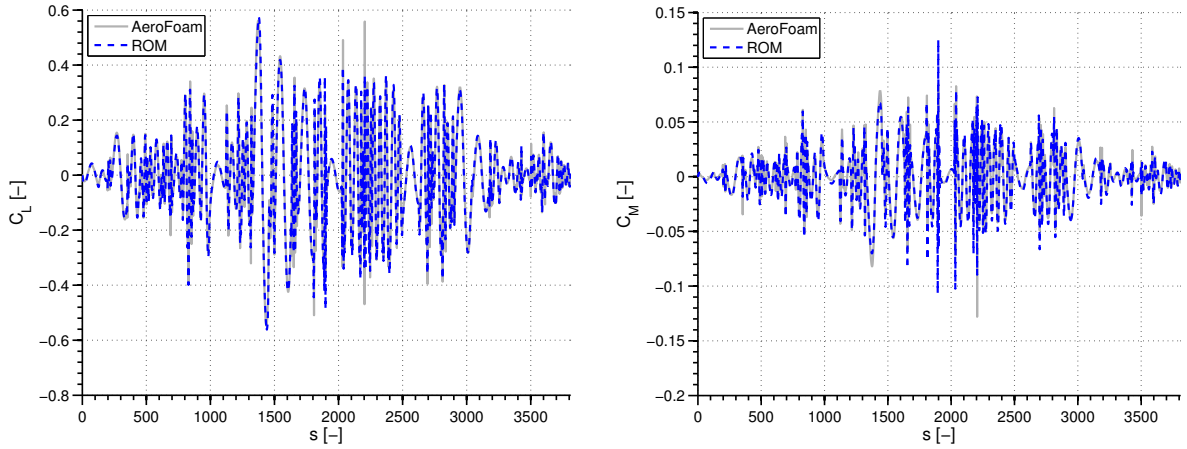


Figure 6.17: C^{2}TRNN , test Y - 2 DOFs airfoil

The C^{2}TRNN seems to have better generalization properties of the previous network presented. The numerical oscillations in test X are present again, but in this case they are much more smaller. Instead, the generalization for input Y seems to be very good. Such better performances may be due to the absence of the bias terms $\mathbf{b}_1$ and $\mathbf{b}_2$, which can induce poor generalization performances since their constant value computed for only one training signal. Such problem may be resolved adopting a training signal which is composed by both the hybrid and the random signals, but in this case this solution has not been followed, since the network performances are retained good. The computational time required for such responses are of 4 $[s]$ each.

CTRNN.LI

The generalization results obtained with the trained CTRNN.LI are reported in figures 6.18 and 6.19.

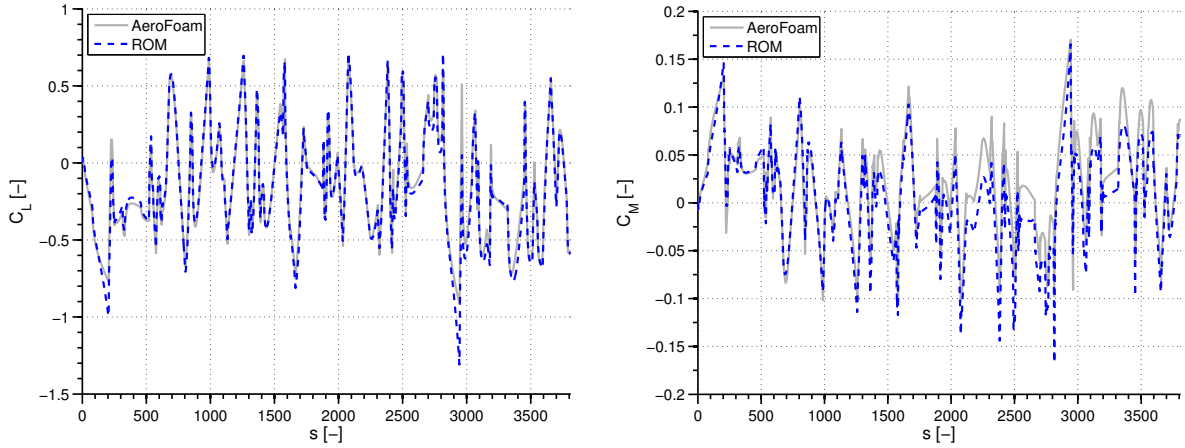


Figure 6.18: CTRNN.LI , test X - 2 DOFs airfoil

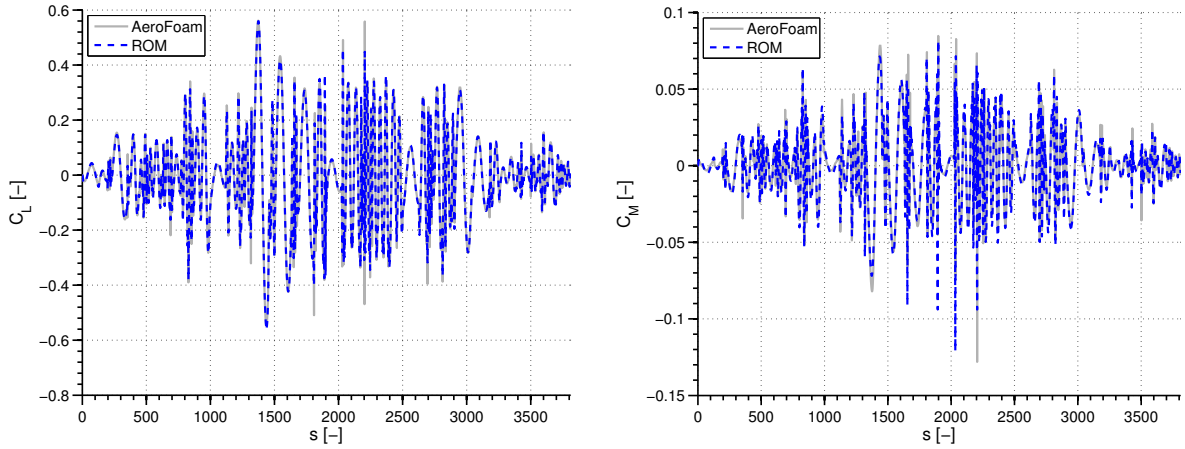


Figure 6.19: CTRNN.LI , test Y - 2 DOFs airfoil

6.1.5 Aeroelastic simulation

The trained CTRNNs are now tested in the prediction of the aerodynamic loads acting on the typical section shown in figure 6.1 in a transonic flow condition. The inertial and stiffness characteristics of the airfoil have been summarized in table 6.1.

A CFD simulation has been performed by AeroFoam at the reduced velocity $V^* = 0.739$. The simulation time has been fixed to $T_S = 4.5[s]$ with a physical time step of $\Delta t = 1 \cdot 10^{-3}[s]$. The Euler flow model has been selected. The selected solver is `TimeStepping`, thus no MG method is employed in this simulation. The interface selected is `Modal`, therefore in the folder `TestCase/Data/` are present the files containing the modal shapes of the mechanical system. A DTS strategy is employed, with the pseudo-time step solver characterized by a first order accuracy, 10^{-3} as treshold value for the residual of the pseudo-time problem and a maximum number of iterations fixed to 500. A RK4 time-marching scheme is selected, with a maximum CFL number allowed equal to 0.8. The following initial condition has been imposed: $h_0/b = 0.1$ and $\theta_0 = -0.1[deg]$.

The problem is integrated forward in time, starting from the assigned initial condition: the aerodynamic loads are computed for the initial condition, then, after having applied an interpolation operator that traduce the aerodynamic loads Q_a evaluated in the aerodynamic mesh in the ones acting on the structural mesh Q_s, the response of the system, in terms of plunge and pitch displacement and velocity, represented by the vector q_s in figure 6.20 is computed solving 6.4. With this new boundary conditions q_a, translated to the aerodynamic mesh by means of an adequate interpolation operator, the aerodynamic problem is solved again, and so on in a closed-loop fashion. A graphical representation is given in figure 6.20.

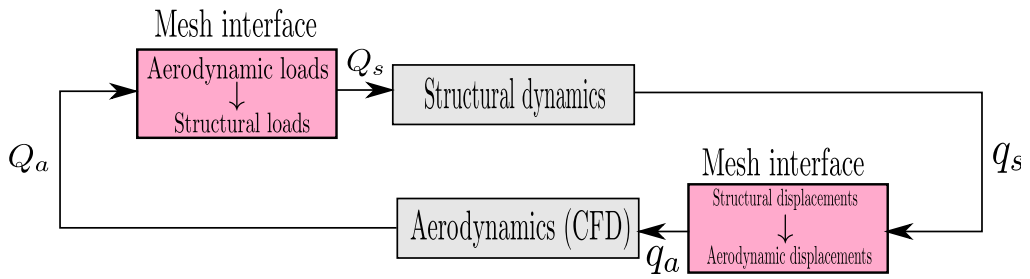


Figure 6.20: CFD closed loop workflow

The computational time required has been of $40\,[h]\,42\,[min]$. Most of the burden of this computation is due to the solution of the aerodynamic problem, since at each time step $4 \cdot 12200 = 48800$ unknowns have to be computed, instead in this case the structural dynamic model is composed by only two second order ODEs, thus four ODEs of the first order.

For these settings of the aeroelastic system a LCO condition is encountered, as can be seen in [31]. The aeroelastic response obtained by AeroFoam is shown in figure 6.21, meanwhile the loads are shown in figure 6.22.

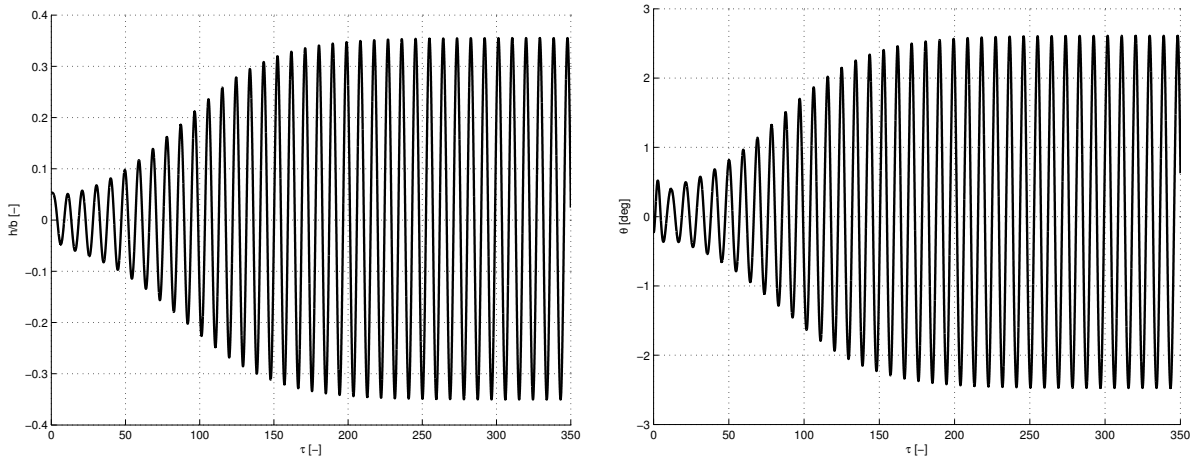


Figure 6.21: LCO condition obtained by AeroFoam, $V^* = 0.739$

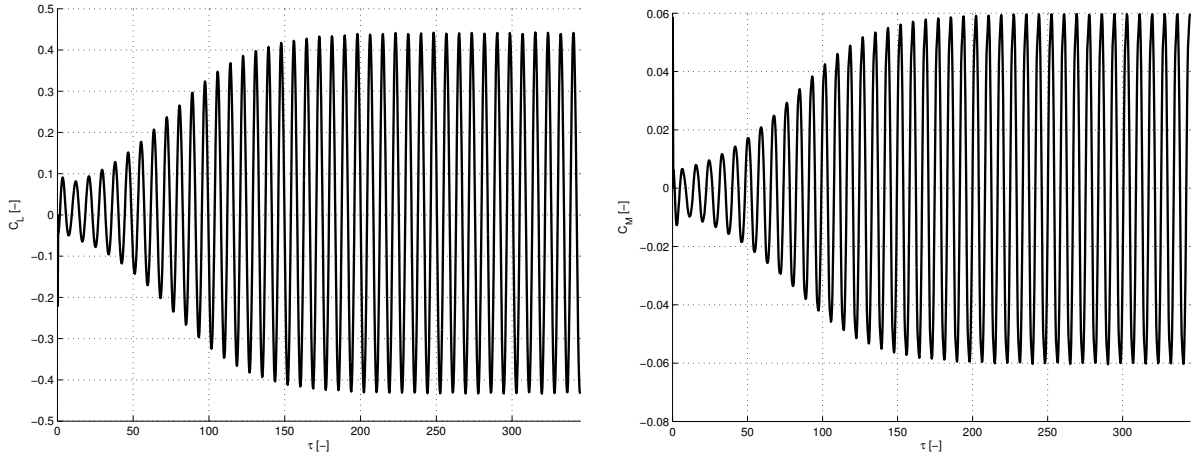
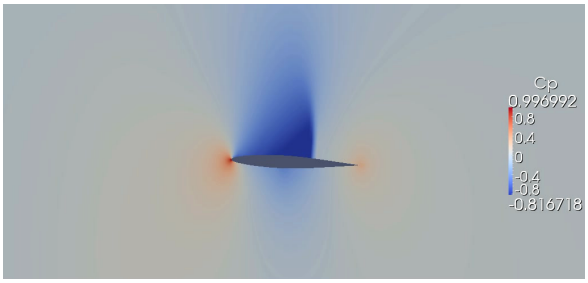


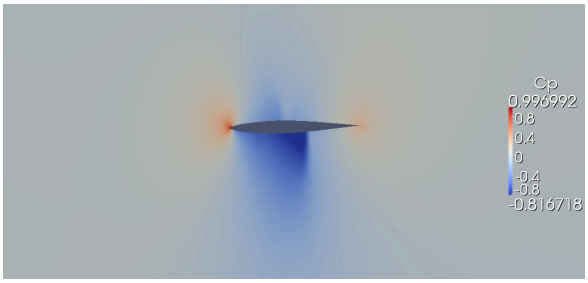
Figure 6.22: LCO loads obtained by AeroFoam, $V^* = 0.739$

The obtained LCO is characterized by the amplitudes $\bar{h}/b = 0.34$ and $\bar{\theta} = 2.61[deg]$, with a reduced frequency of $k_{\mathrm{LCO}} = 0.2077$. The reduced frequency of interest is significantly smaller than the maximum frequency allowed in the training signal, therefore, if the trained CTRNN has been able to include in its implicit dynamics such an attractor, it will interpolate the response from the trained one.

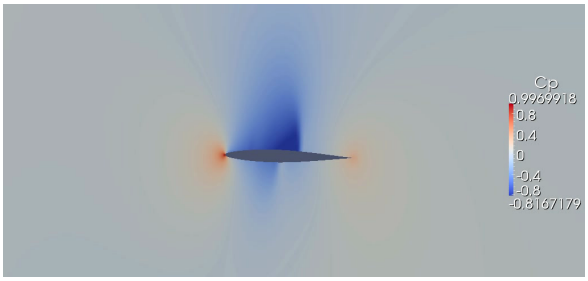
This working condition is characterized by a strongly nonlinear fluid flow condition, since the shocks on the upper and the lower surface of the airfoil experience a displacement of $\Delta s = 21\%\, c$ during half of the period. Such cycle is shown in figure 6.23.



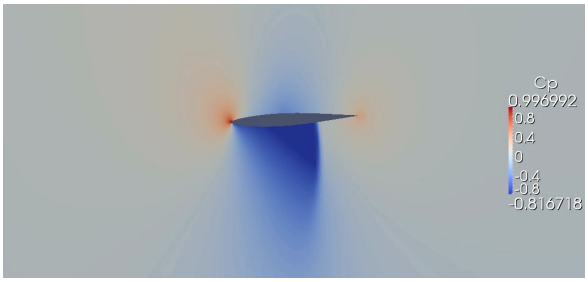
(a) Maximum position reached by the upper shock, the lower shock is absent



(b) Minimum position reached by the upper shock, the lower shock has already developed



(c) Minimum position reached by the lower shock, the upper shock has already developed



(d) Maximum position reached by the lower shock, the upper shock is absent

Figure 6.23: Shock's displacement cycle for the LCO computed at $V^* = 0.739$

As can be seen, not only the shock experiences a large displacement, but it also disappear and reappear during an LCO period.

The different CTRNNs trained in the previous section with the training signal shown in figure 6.5 are now coupled to the structural model presented in equation 6.4. The resulting aeroelastic system should be able to behave like the CFD model employed before, thus, having captured the main nonlinearities of the flow, the LCO can be computed using a more compact model than the coupled structural-CFD model. In what follows, the mass matrix will be represented with the symbol $\mathbf{M}$, the stiffness matrix by the symbol $\mathbf{K}$ and the DOFs vector by the symbol $\mathbf{u}$. A graphical representation of the aeroelastic ROM is given in figure 6.24.

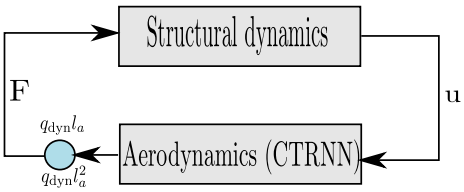


Figure 6.24: ROM closed loop workflow

Such workflow shown the efficacy of the CTRNN in the representation of the aerodynamic problem: there is no need of interpolation operators to interface the aerodynamic to the structural system, and the resulting dynamic model is characterized by a very low dynamic order, say $\mathcal{O}\left(10^0\right)$ with respect $\mathcal{O}\left(10^5\right)$ of the CFD model. Calling $\mathbf{B}$ the diagonal matrix defined as:

$$\mathbf{B} = \mathrm{diag}\left(\mathrm{q_{dyn}c}, \mathrm{q_{dyn}c^2}\right) \tag{6.8}$$

the aerodynamic load acting on the structure can be written as:

$$\mathbf{F} = \mathbf{B}\mathbf{W}^c\mathbf{x} \tag{6.9}$$

recalling the definition of the network output from chapter 3.

For the time integration of the ROM dynamic system, the AD explicit technique has not been adopted, but an implicit time-marching scheme developed in [70] and well explained in [2], is adopted, since a much more great time step can be adopted in the solution of the ODE system associated to the ROM. In practice the ODE solver used here is a simplified version of the one presented in the reference, since in this case the time step is mantained constant. Let us consider the generic ODE:

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, t) \tag{6.10}$$

The integration method presented here is A-L stable, multi-step with two variable steps:

$$\mathbf{z}_{k+1} = a_0\mathbf{z}_k + a_{-1}\mathbf{z}_{k-1} + b_1\mathbf{f}(\mathbf{z}_{k+1}) + b_0\mathbf{f}(\mathbf{z}_k) + b_{-1}\mathbf{f}(\mathbf{z}_{k-1}) \tag{6.11}$$

where the subscript k characterizes the solution $\mathbf{z}$ at the k-th time step. The coefficients can be expressed in terms of the time step and the desired level of numerical dissipation $\tilde{\rho}$:

$$\begin{aligned}
\beta &= \frac{3(1-\tilde{\rho})^2 + 4(2\tilde{\rho}-1)}{4-(1-\tilde{\rho})^2} \\
\delta &= \frac{(1-\tilde{\rho})^2}{2\left[4-(1-\tilde{\rho})^2\right]} \\
a_0 &= 1-\beta \\
a_{-1} &= \beta \\
b_1 &= \Delta t\left(\delta + \frac{1}{2}\right) \\
b_0 &= \Delta t\left(\frac{\beta}{2} + \frac{1}{2} - 2\delta\right) \\
b_{-1} &= \Delta t\left(\frac{\beta}{2} + \delta\right)
\end{aligned} \tag{6.12}$$

Such implicit solver has been considered for two reasons: first the ODE system can be integrated forward in time with a greater time step than the one employed in the CFD simulation. Second, it is a good test for the capability of the CTRNN to work on a much larger time step then the one on which it has been trained. For these reasons, the selected time step for the implicit integrator scheme is equal to $\Delta t = 10^{-2}[s]$ and the level of numerical dissipation has been fixed to $\tilde{\rho} = 0.8$. In this way the numerical scheme behaves closely to a Crank-Nicholson (CN) implicit scheme than a BDF2 scheme, thus mantaining a good level of accuray while introducing a small numerical dissipation. Since the adopted integration method needs the previous two time steps solution in order to compute the one at the current time step, the start-up of the integrator requires an ad-hoc strategy, since at the first time step only the initial condition of the problem is known. This problem is solved here by exploiting the CN scheme at the first time step:

$$\mathbf{z}_1 = \mathbf{z}_0 + \frac{\Delta t}{2}\left[\mathbf{f}(\mathbf{z}_0) + \mathbf{f}(\mathbf{z}_1)\right] \tag{6.13}$$

In this way at the second step the solution at the two previous time steps is known, and the proposed integration procedure can be applied.

ECTRNN simulation

The ECTRNN trained in section 6.1.2 is coupled tp the dynamical model representing the 2 DOFs airfoil. The resulting aeroelastic system is represented by the following system of ODE:

$$\begin{cases} \mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{B}\mathbf{W}^c\mathbf{x} \\ \dot{\mathbf{x}} = \mathbf{W}^x\mathbf{\Phi}\left(\mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u} + \mathbf{b}_1\right) + \mathbf{b}_2 \end{cases} \tag{6.14}$$

Following the approach explained in section 2.1.1, system 6.14 is forced for a fixed time by the following motion:

$$\begin{cases} h(t) = 0.1\ \sin(\omega_f\, t) \\ \theta(t) = 1.4\ \sin(\omega_f\, t) \end{cases} \tag{6.15}$$

where ω_f is the frequency associated to the reduced frequency $\hat{k} = 0.3$, thus $\omega_f = \dfrac{V_\infty\, \hat{k}}{c}$. The results obtained are shown in figures 6.25 and 6.26.

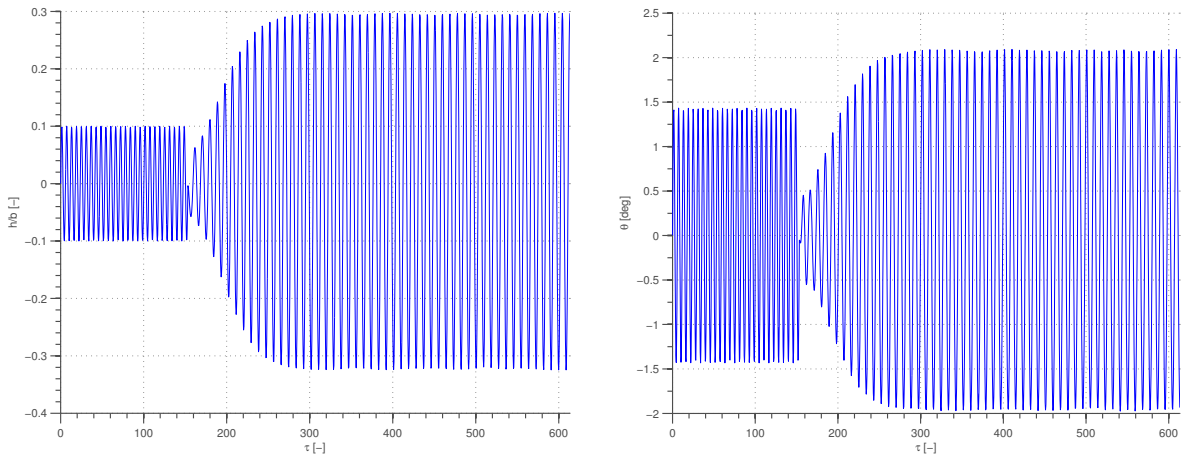


Figure 6.25: LCO condition obtained by ECTRNN , $V^* = 0.739$

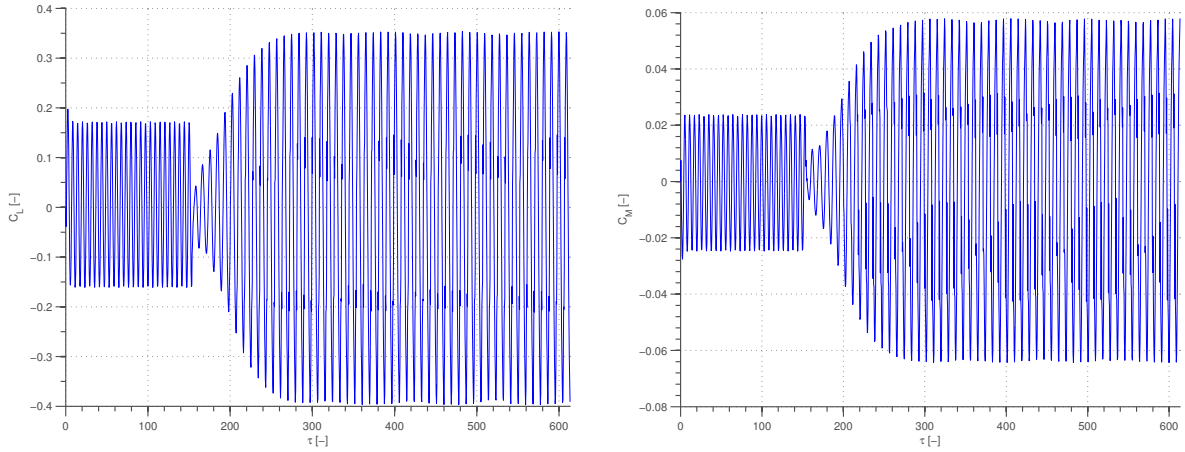


Figure 6.26: LCO loads obtained by ECTRNN , $V^* = 0.739$

The ROM experienced a LCO which is slightly different from the LCO obtained by the CFD simulation. First of all, it is slightly non-simmetric. This can be due to the presence of the bias which introduce always a constant contribution to the response of the system. Thus the value of

$\mathbf{b}_1$ and $\mathbf{b}_2$ resulting from the training may introduce a not null mean value in the response. The amplitudes of the LCO are also different from the CFD simulation, the plunge amplitude is equal to $\bar{h}/b = 0.3$, meanwhile the pitch one is equal to $\bar{\theta} = 2[deg]$. Thus a substantial error of $0.6[deg]$ (about 23%)is present in the evaluation of the LCO amplitude. However, the estimated reduced frequency is equal to $k_{\text{LCO}} = 0.2074$, thus no error in the estimation of the reduced frequency has been encountered. The computational time of the response has been of $43\,[s]$.

C^{2}TRNN simulation

The C^{2}TRNN trained in section 6.1.2 is coupled to the dynamical model representing the 2 DOFs airfoil. In this case the resulting aeroelastic system is represented by the following system of ODE:

$$\begin{cases} \mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{BW}^c\mathbf{x} \\ \dot{\mathbf{x}} = \mathbf{W}^x\mathbf{\Phi}\left(\mathbf{W}^a\mathbf{x} + \mathbf{W}^b\mathbf{u}\right) \end{cases} \tag{6.16}$$

Following the approach explained in section 2.1.1, system 6.16 is forced for a fixed time by the movement employed in the previous ECTRNN example. The results obtained are shown in figures 6.27 and 6.28.

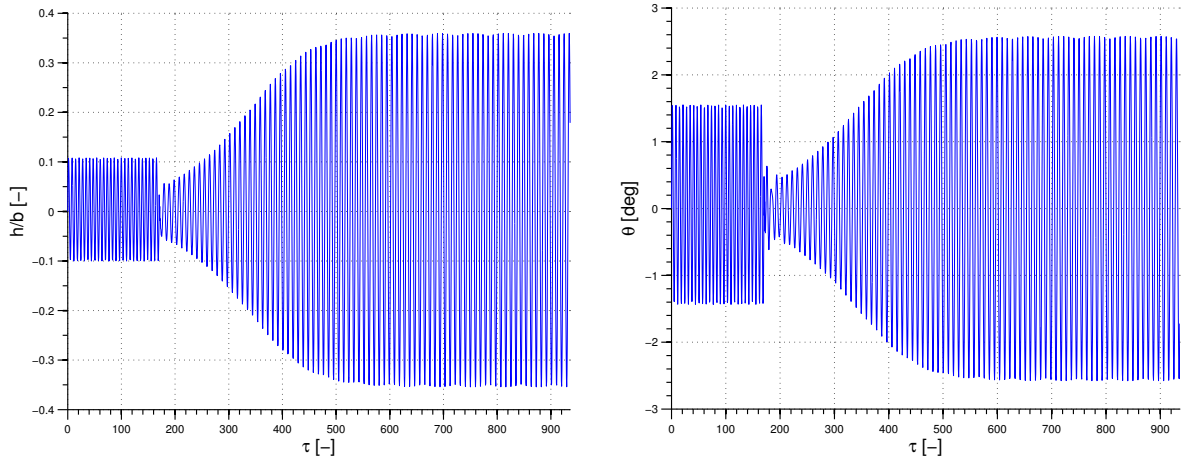


Figure 6.27: LCO condition obtained by C^{2}TRNN , $V^* = 0.739$

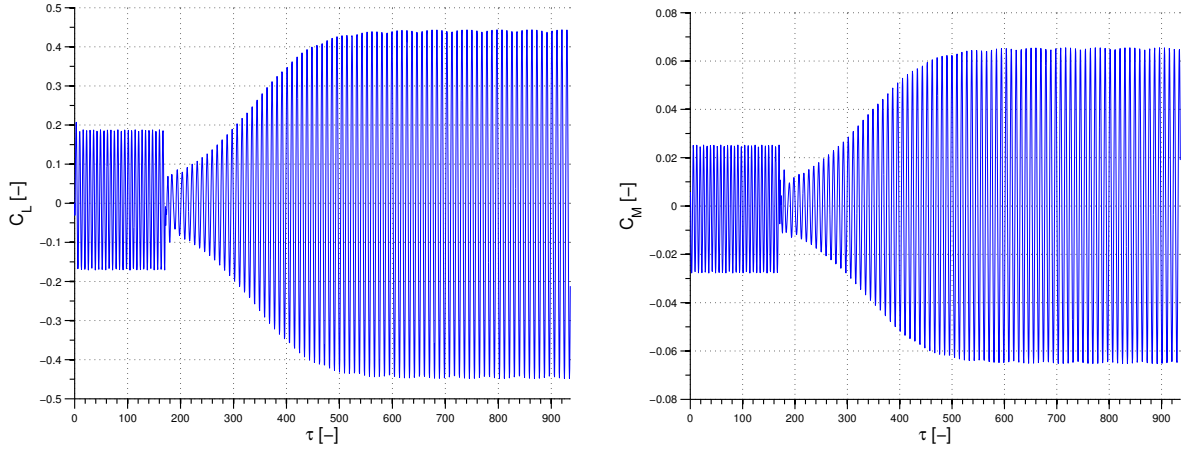


Figure 6.28: LCO loads obtained by C^{2}TRNN , $V^* = 0.739$

Much better results have been obtained in this case. The obtained LCO is symmetric, the

amplitudes are $\bar{h}/b = 0.34$ and $\bar{\theta} = 2.6[deg]$. In this case a total agreement between the ROM and the CFD results have been obtained: this fact can be influenced by the choice of the network's parametrization, since the C^{2}TRNN can be interpreted as a *"symmetric"* network, and therefore a symmetric solution has been forced in some way by imposing a prior knowledge of the system. The reduced frequency obtained is $k_{\mathrm{LCO}} = 0.2077$, thus the LCO frequency has been correctly predicted. The computational time of the response has been of 43 $[s]$.

CTRNN.LI simulation

The CTRNN.LI trained in section 6.1.2 is coupled to the dynamical model representing the 2 DOFs airfoil. In this case the resulting aeroelastic system is represented by the following system of ODE:

$$\begin{cases} \mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{B}\mathbf{W}^c\mathbf{x} \\ \dot{\mathbf{x}} = \mathbf{W}^a\mathbf{\Phi}\left(\mathbf{x}\right) + \mathbf{W}^b\mathbf{u} \end{cases} \tag{6.17}$$

Following the approach explained in section 2.1.1, system 6.17 is forced for a fixed time by the movement employed in the previous ECTRNN example. The results obtained are shown in figures 6.29 and 6.30.

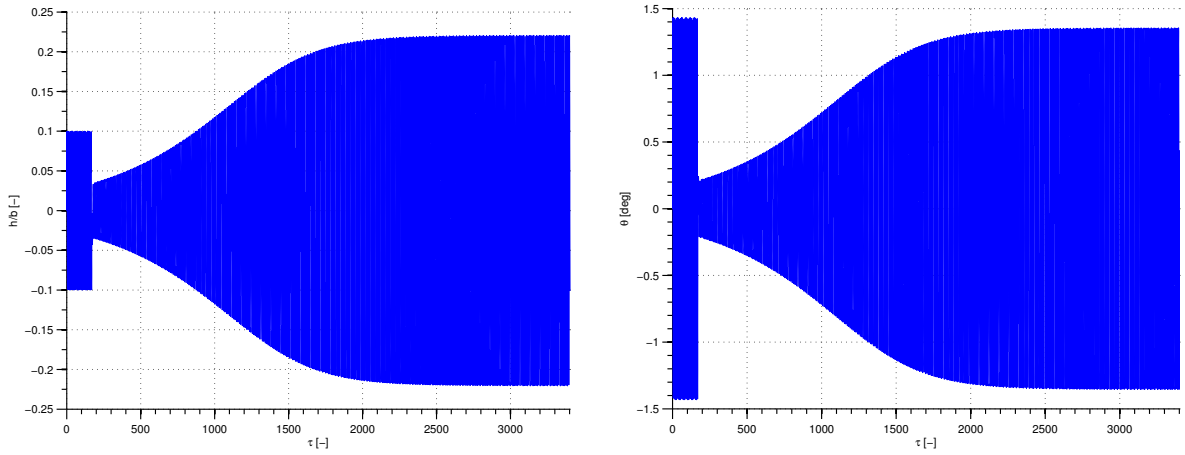


Figure 6.29: LCO condition obtained by CTRNN.LI , $V^* = 0.739$

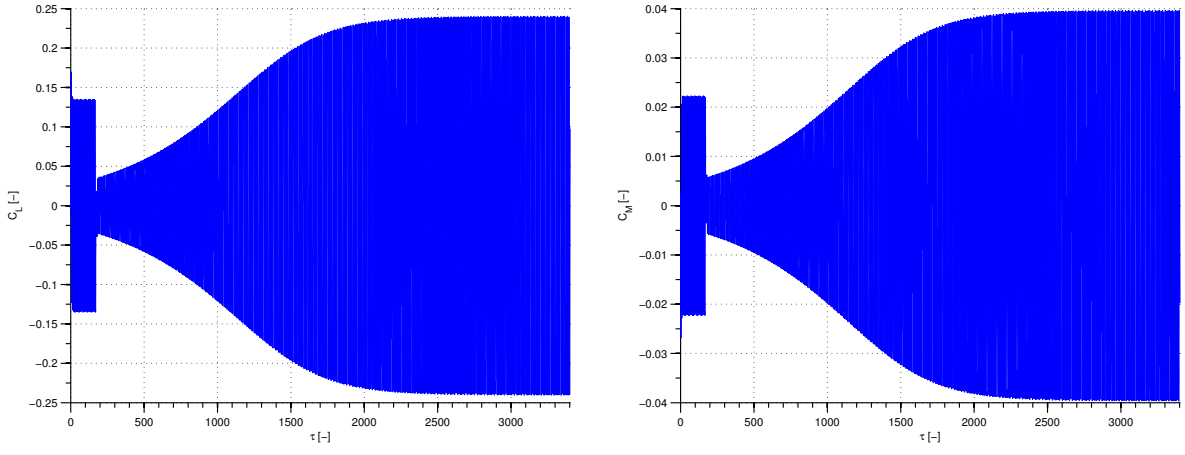


Figure 6.30: LCO loads obtained by CTRNN.LI , $V^* = 0.739$

The predicted LCO is symmetric, but the amplitudes differ from the CFD simulation. The

predicted amplitudes are $\bar{h}/b = 0.22$ and $\bar{\theta} = 1.35[deg]$. The amplitude error in this case is substantial in both pitch and plunge responses. However, also in this case, the predicted LCO frequency is equal to $k = 0.2061$, thus very near to the one computed by the CFD analysis. The computational time of the response has been of $1\,[min], 34\,[s]$, since in order to capture the full developed LCO a greater time has been required. This means that the CTRNN.LI model has captured a different nonlinear behaviour from the training signal than the one captured by the other two parametrizations.

6.1.6 Envelope analysis by time marching simulations

The efficiency of the developed ROM can be exploited in the computation of the LCO envelope over different values of the torsional stiffness k_θ, which translates into variations of the reduced velocity V^*. Such analysis can be performed in a very small time with respect to the computational time required to perform the same analysis with a CFD solver. Thus at this point various values of the reduced velocity are considered, and the corresponding LCO amplitudes and reduced frequency are recorded. Finally the results obtained are compared with some CFD solution obtained with AeroFoam and one present in the literature [33]. In the considered reference the LCO envelope is computed by means of a DTRNN with RBF activation functions, for this reason the results obtained by such work are marked by the name of its activation functions. The results obtained are very similar, as can be seen from figures 6.31 and 6.32.

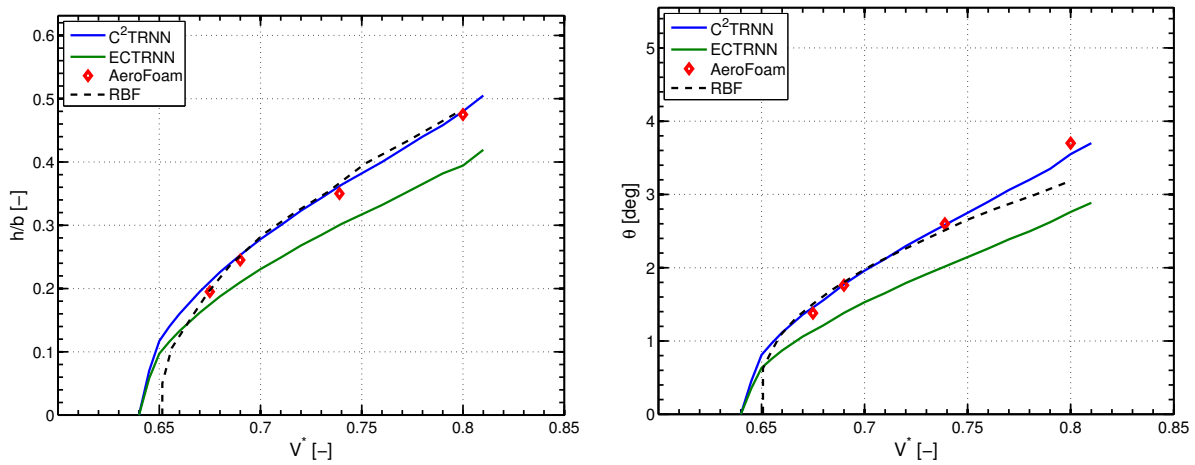


Figure 6.31: LCO envelope of amplitudes versus reduced velocity computed by different approaches

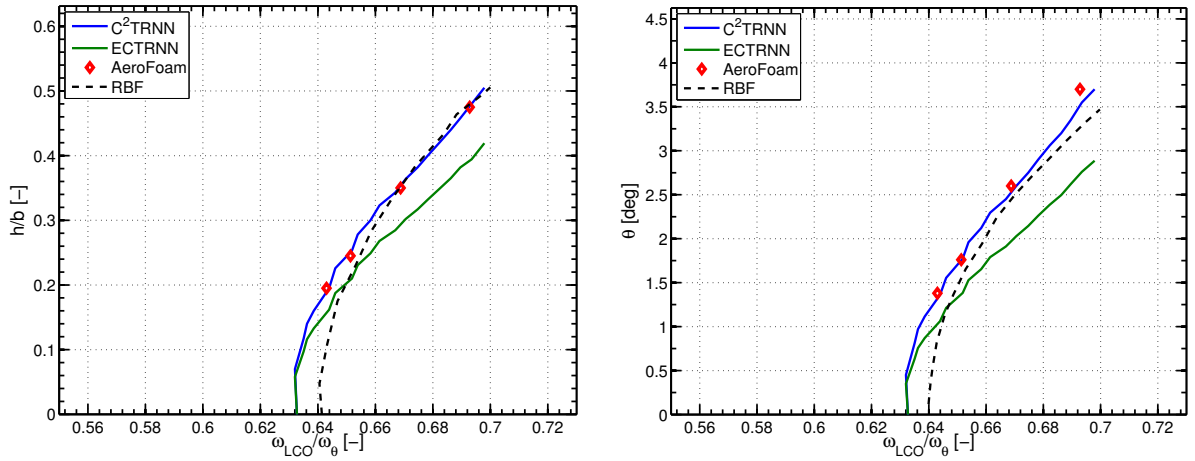
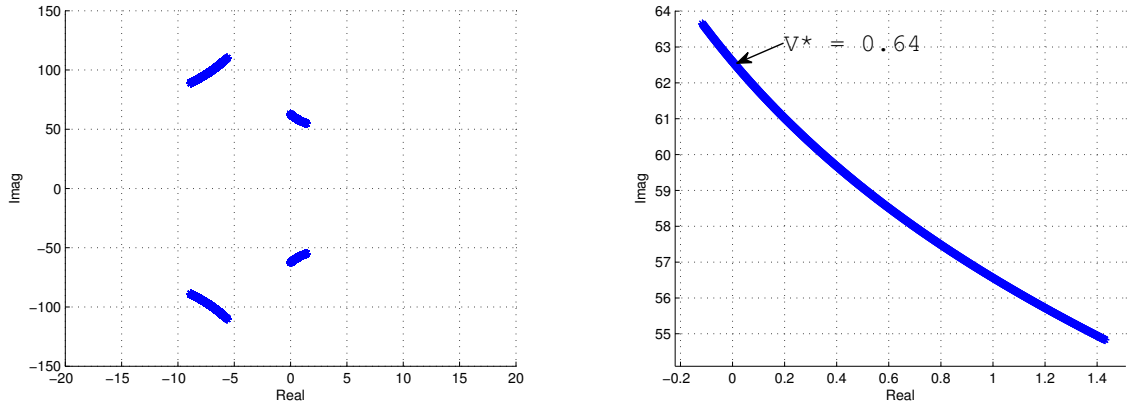


Figure 6.32: LCO envelope of amplitudes versus LCO frequency ratio computed by different approaches

The bifurcation point has been computed very quickly by an eigenvalue analysis of the relative Jacobian matrix evaluated at the origin with varying reduced velocity. The results obtained are identical for both the ECTRNN and C^{2}TRNN , and the eigenvalue evolution found near the origin with the ECTRNN is shown in figure 6.33.



(a) Zoom near the origin of the eigenvalue space

(b) Zoom near the bifurcation point

Figure 6.33: Bifurcation analysis

Quite satisfactory results have been obtained for both reduced velocity and frequency ratio envelope, especially for what regard the comparison of the results obtained with AeroFoam. Unsatisfactory results have been obtained from the CTRNN.LI , since the bifurcation point computed by an eigenvalue analysis of the related Jacobian matrix is located at $V^* = 0.705$ at a reduced frequency of $k = 0.2106$. The results obtained in this case are shown in figure 6.34.

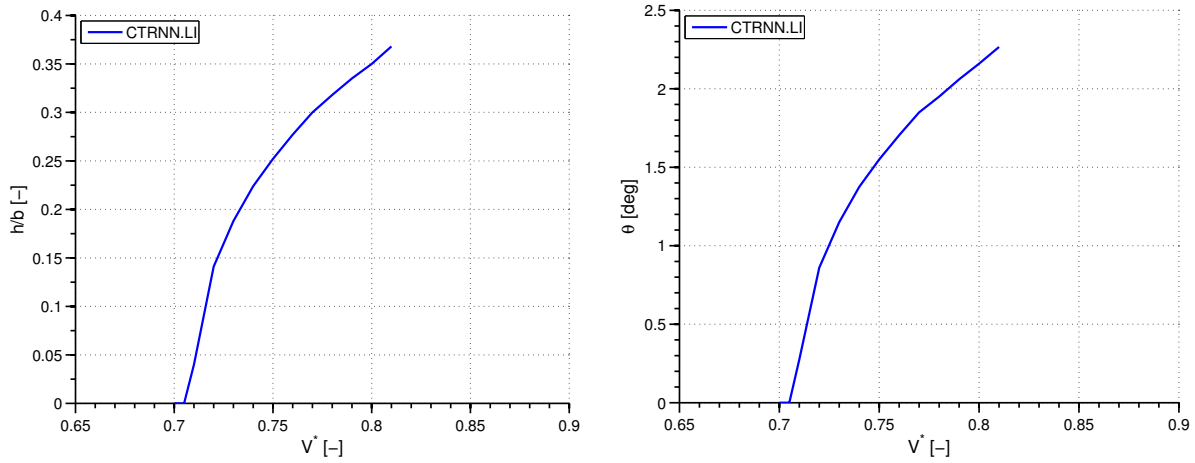


Figure 6.34: LCO envelope computed by CTRNN.LI

This kind of network has not been able to capture the true nonlinear behaviour of the system, since it presents a different bifurcation point and a different behaviour in front of a varying reduced velocity with respect of the other two parametrizations. Some considerations on this behaviour will be given in the next section, but for now such kind of neural network is not considered in the following sections.

Nevertheless, the results obtained with the ECTRNN and C^{2}TRNN parametrizations, in particular the latter, show a good agreement with the CFD results obtained with AeroFoam: which means that the scope of work prefixed which was a ROM working as a "*mini*" AeroFoam solver has been reached. Very good results have been obtained in the both comparisons with the LCO amplitudes and frequency ratio. It is interesting to compare the computational time required for the evaluation of such LCO envelope first by the CFD code and then by the ROM. Such comparison is presented in tables 6.5 and 6.6.

Computational time required for five time accurate simulations	1 [h] 12 [min] + 40 [h] 02 [min] + 40 [h] 42 [min] + 41 [h] 00 [min] + 41 [h] 10 [min]
Total time required	164 [h] 06 [min]

Table 6.5: CFD-based simulation, computational time

	ECTRNN	C^{2}TRNN
Generation of the training input-output data pair by AeroFoam	24 [h] 01 [min]	
Training time	6 [h] 58 [min]	6 [h] 01 [min]
Envelope simulation time (20 points)	20 [*min*]	20 [*min*]
Total time required	31 [h] 19 [min]	30 [h] 22 [min]

Table 6.6: ROM-based simulation, computational time

The ROM constructed by the response of the aerodynamic system at a fixed velocity can be thus be used to perform different dynamic analyses for different values of the torsional stiffness

Moreover, the efficiency of the ROM is proved, showing a computational time saving in the order of 160 hours with respect of the CFD-based simulations.

6.1.7 Envelope analysis by periodic collocation in the time domain

It is now interesting perform the computation of the LCO envelope by the periodic collocation procedure detailed in section 2.1.2. Since a first evaluation of such envelope is available from the previous analyses performed considering the initial value problem associated to the aeroelastic system, the stored data can be considered here for a sensitivity analysis of the solution with respect of the initial guess of the period. Moreover, also the influence of the initial tentative imposed state on the converged solution may be studied. Therefore, first a sensitivity analysis of the solution with respect to the initial guess will be considered, in particular the already mentioned condition at $V^* = 0.739$ will be analyzed. Then the computation of the LCO envelope will be carried out with this technique.

For all the analyses considered, the mixed method composed by the MPR and the BDF2 will be considered, with the mixing coefficent $\beta = 0.4$. Thus the damping of the BDF2 method is preferred over the accuracy of the MPR, in order to avoid spurious oscillating solutions. The method starts with 16 subdivisions of the initial guess of the period, then a refinement of the number of time nodes is carried out as the residual of the estimated period diminishes.

As said previously, the test case $V^* = 0.739$ is considered here for a sensitivity analysis of the solution with respect of the initial guess on the period and the fixed state value. For what regard the C^2TRNN , three sensitivity analyses have been carried out, considering:

- Case A: various amplitudes for the fixed value of the h/b variable, with an initial guess of the period equal to $T_0 = 0.09$;
- Case B: various amplitudes for the fixed value of the θ variable, with the same initial guess for the period as before;
- Case C: it is considered the influence of the initial guess of the period on the converged solution, fixing the first component of the state at 0.1 and varying the initial guess on T.

The results obtained are shown in tables 6.7, 6.8, 6.9.

Case A			
h/b imposed	LCO amplitude h/b	LCO amplitude $\theta\,[deg]$	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 6.7: LCO sensitivity analysis with respect to the value of the fixed h/b variable

Case B			
θ imposed	LCO amplitude h/b	LCO amplitude $\theta\,[deg]$	LCO period $[s]$
1.5		Unstable LCO	
1.8		Unstable LCO	
2		Unstable LCO	
2.2	0.34	2.6	0.1111
2.4	0.34	2.6	0.1111
2.5	0.34	2.6	0.1111

Table 6.8: LCO sensitivity analysis with respect to the value of the fixed θ variable

Case C			
Initial guess on T	LCO amplitude h/b	LCO amplitude $\theta\,[deg]$	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 6.9: LCO sensitivity analysis with respect to initial guess of the period

Therefore the method seems to be robust in the face of various perturbations of the LCO parameters. Such robustness is shown also in the following analysis, where the imposed value of h/b is equal to 0.2 and the initial guess of the period is $T_0 = 0.075\,[s]$. The results of such analysis are shown in figure 6.35.

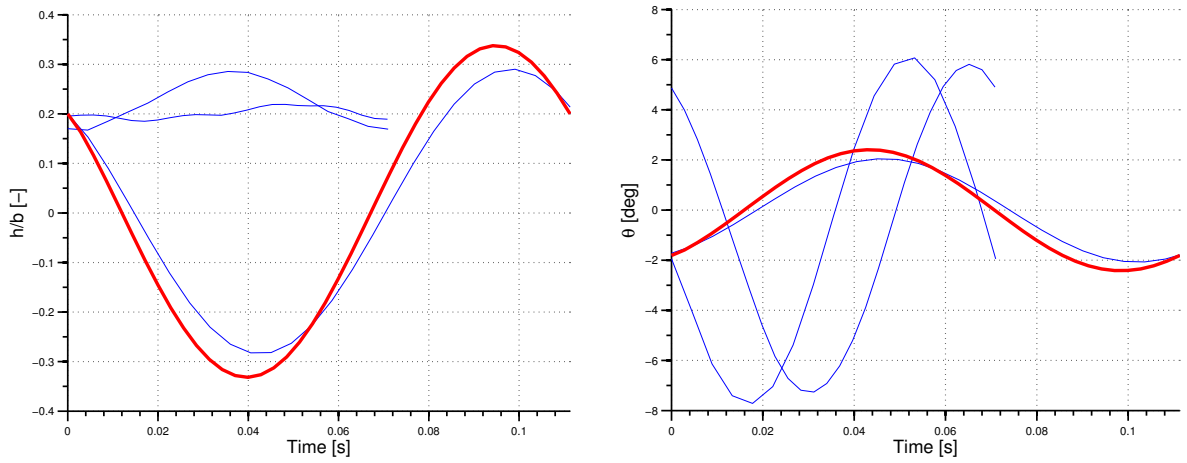


Figure 6.35: LCO computation by period collocation in the time domain, initial guess of the period $T_0 = 0.075\,[s]$. The converged solution is indicated as (—)

In this case the number of sudivisions has been increased from the 16 of the first iteration to the 66 of the last one, with an increasing of 5 intervals in each iteration. However a convergent results has been obtained after only four iterations, this means that the LCO behaviour is well embedded into the constructed ROM.

At this point the evaluation of the LCO envelope can be carried out, starting from the results of the time marching procedure, with a continuation technique: the current solution is used as initial guess for the iterations at the next value of the reduced velocity and so on. A precise agreement of the results has been achieved, as expected. The results are shown in figures 6.36 and 6.37 for both the ECTRNN and C^2TRNN parametrizations.

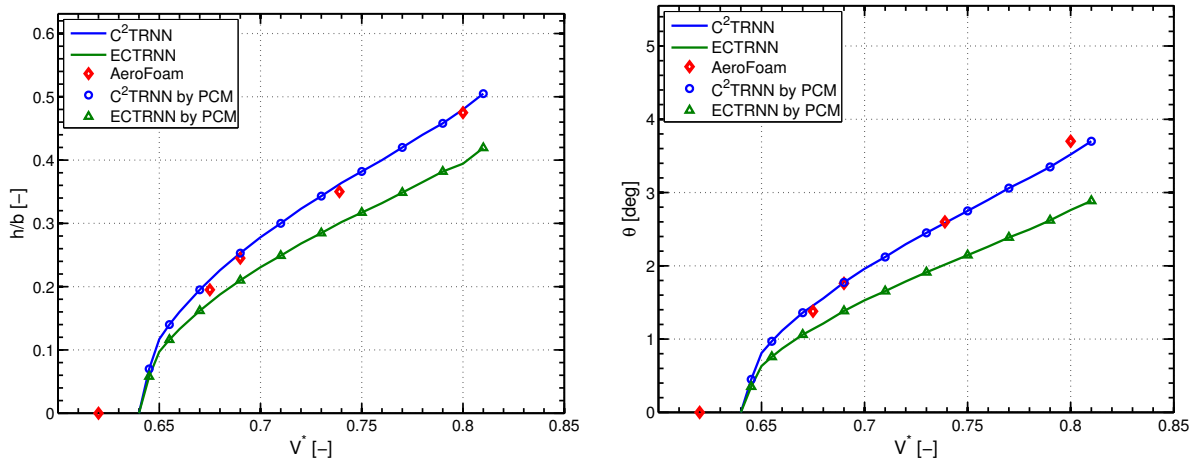


Figure 6.36: LCO envelope of amplitudes versus reduced velocity, comparison between the time marching method and the periodic collocation (indicated as PCM)

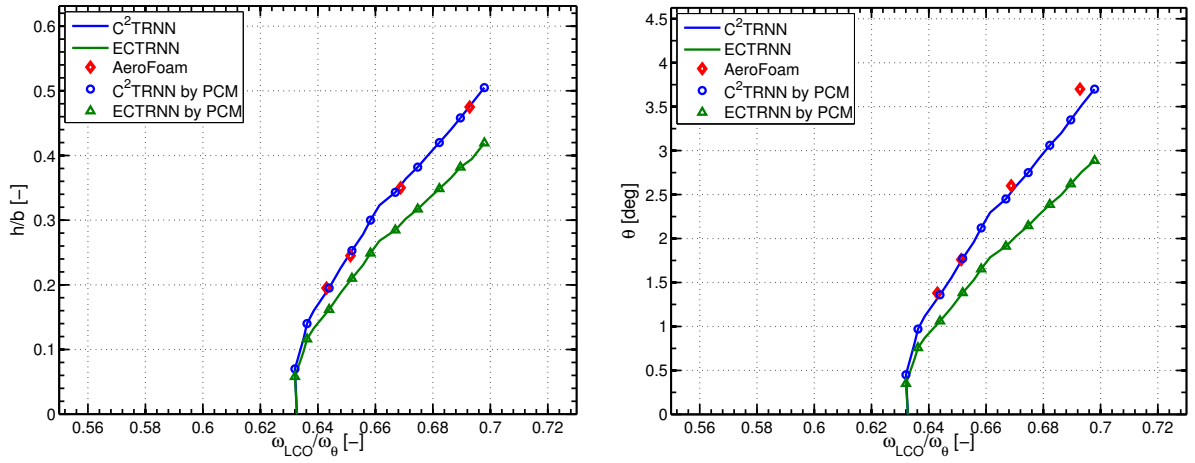


Figure 6.37: LCO envelope of amplitudes versus LCO frequency ratio, comparison between the time marching method and the periodic collocation (indicated as PCM)

Finally, the computational time required for performing the envelope analysis with the time periodic collocation is shown in table 6.10.

	Computational time required (10 envelope points)
ECTRNN	26 $[min]$
C^2TRNN	22 $[min]$

Table 6.10: Computational time required for the computation of the LCO envelope by the periodic collocation method

The computational time required to construct the LCO envelope is slightly greater than the one experienced for the initial value problem simulation. However this characterization of the LCO is much more interesting, since the interpretation of the LCO as a boundary value problem permits to analyze such behaviour with a more rigorous mathematical approach than the initial value problem formulation. Here below are finally shown the results obtained for the C^2TRNN case at reduced velocities equal to $V^* = 0.7$, $V^* = 0.77$ and $V^* = 0.8$.

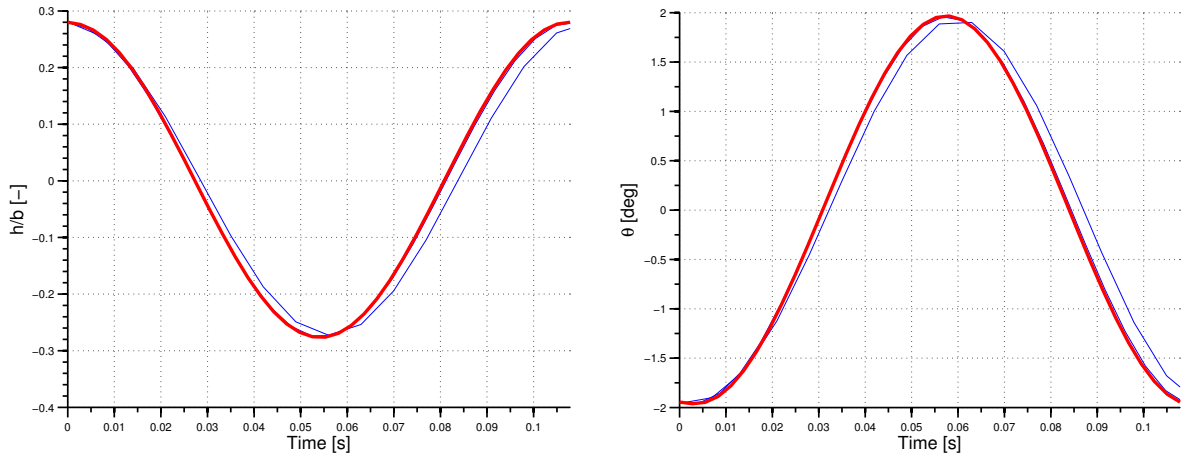


Figure 6.38: LCO computation by period collocation in the time domain, $V^* = 0.7$, initial guess of the period $T_0 = 0.1, [s]$. The converged solution is indicated as (—)

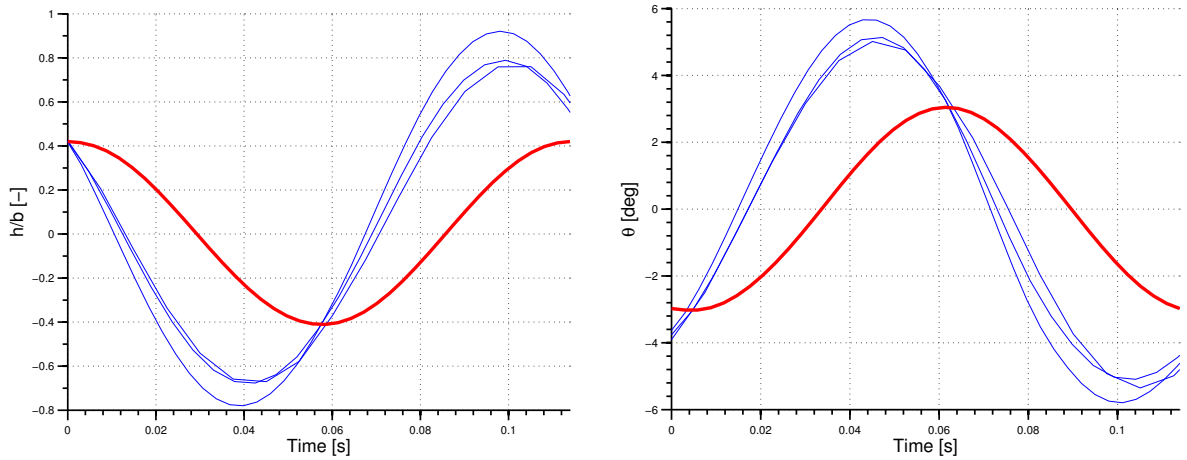


Figure 6.39: LCO computation by period collocation in the time domain, $V^* = 0.77$, initial guess of the period $T_0 = 0.1, [s]$. The converged solution is indicated as (—)

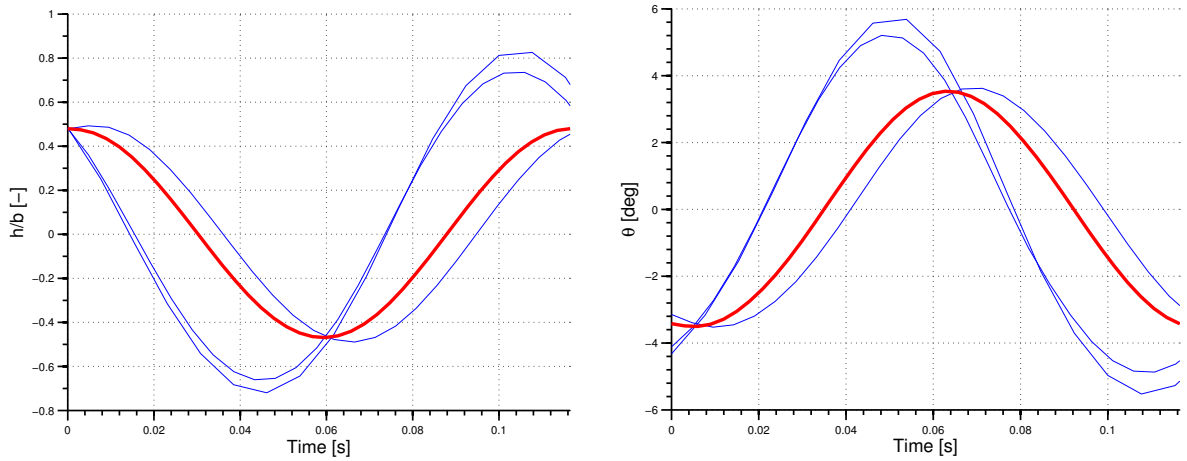


Figure 6.40: LCO computation by period collocation in the time domain, $V^* = 0.8$, initial guess of the period $T_0 = 0.11, [s]$. The converged solution is indicated as (—)

6.2 BACT wing

The other test case considered in this chapter is the Benchmark Active Control Technology (BACT) wing [71, 72]. Such a case is considered here only partially and it is currently under development. The BACT project is part of NASA Langley Research Center's Benchmark Models Program for studying transonic aeroelastic phenomena. The BACT wind-tunnel model was developed to collect high quality unsteady aerodynamic data, in terms of pressure and loads, near transonic flutter conditions and demonstrate flutter suppression using spoilers [71].

Such a model is a rigid, rectangular wing with a NACA 0012 airfoil section. It is equipped with trailing-edge control surface and upper/lower-surface spoilers that can be controlled independently via hydraulic actuators. The control surfaces are assumed blocked in the present analysis. The wing is mounted on a device called the Pitch And Plunge Apparatus (PAPA), which is designed to permit the motion along the two principal modes indicated by its name. A representation of the system is given in figure 6.41.

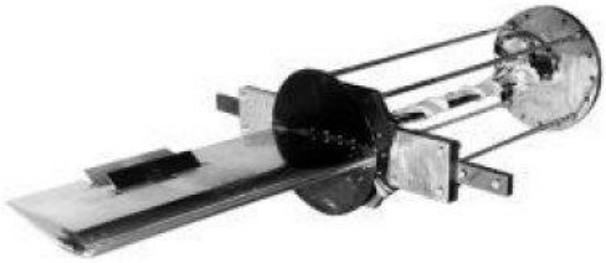


Figure 6.41: BACT system

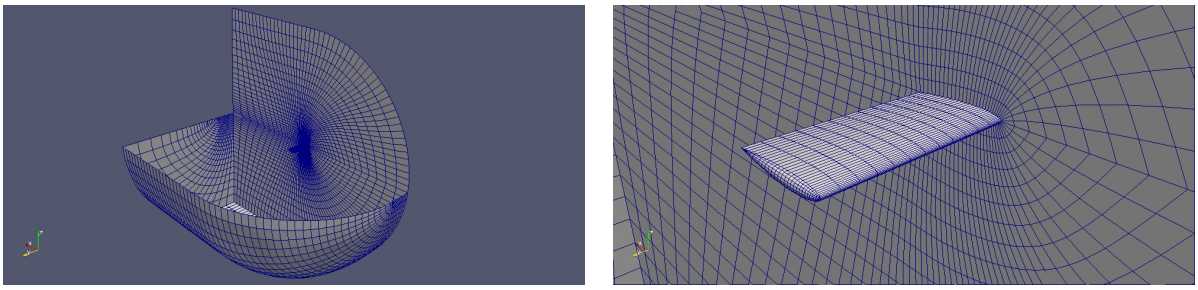
The BACT system, composed by the BACT wing and the PAPA, has a dynamic behaviour very similar to the classical 2 DOFs problem in aeroelasticity, as just shown in figure 6.1. The main difference is the complexity of the aerodynamic behaviour and the presence of additional structural modes. The finite span and low aspect ratio of the BACT wing introduce significant three dimensional flow effects. Higher frequency structural degrees of freedom are associated with the PAPA and the fact that the wing section is not truly rigid. The PAPA system allow the wing to pitch around the mid chord. In the following analysis the BACT will be treated as a 2 DOFs airfoil from the mechanical point of view. This assumption implies that the other structural modes of the BACT wing are not significant. Investigation of the structural vibration characteristics of the PAPA mount was shown to support this assumption. In fact the next lowest frequency for any transverse mode was more than six times the frequency of the pitch and plunge modes and well outside the frequency range of interest [71]. The main characteristics of the BACT system are presented in table 6.11.

M_∞	0.8
x_θ	0.0035
r_θ^2	1.036
ω_h	21.01 $[rad/s]$
ω_θ	32.72 $[rad/s]$
μ	4284
Chord, c	0.4064 $[m]$
Span, s	0.8128 $[m]$
Surface, S	0.3303 $[m^2]$

Table 6.11: BACT wing data

With the fluid to mass ratio now defined as $\mu = \dfrac{m}{\pi \rho_\infty s b^2}$, being b the semi-chord.

The solver AeroFoam is validated through a steady simulation. The computational mesh considered is again a C-mesh generated by a script written in MATLAB programming language. The one adopted in the current simulation and the close-up near the wing is shown in figure 6.42.



(a) Complete mesh (b) Close-up

Figure 6.42: Computational mesh considered for the BACT wing

The Euler flow model has been selected for the present analysis. The same settings of the previous case are assumed. Different mesh sizes have been considered: 40k, 50k and 70k finite volumes. After a convergence analysis, where the reference results are taken from the CFD simulations performed at $\alpha = 0\,[deg]$ in [73], the medium mesh has been selected. This choice has been primarily influenced by the computational time required for achieving the steady state value of the loads. With the selected mesh, about 1 hour has been necessary for obtaining such results, employing 2 CPUs with the same characteristics of the previous analysis. The C_p distribution about the 60% of the span and its contour lines in the flow field are shown in figures 6.43 and 6.44.

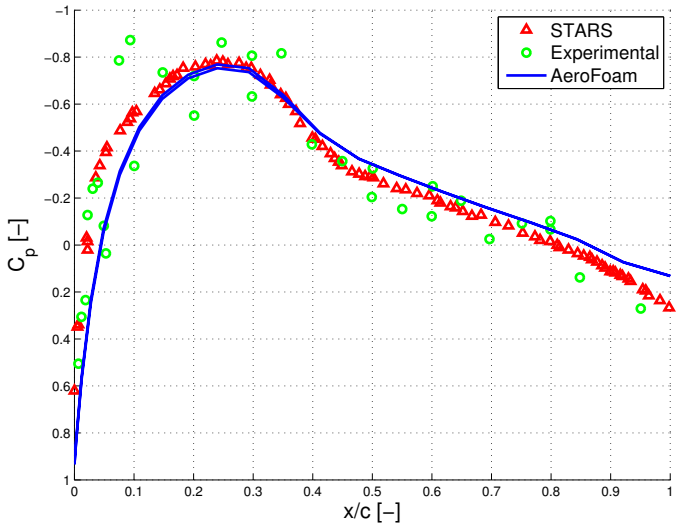


Figure 6.43: $C_p - x$ distribution for the BACT wing

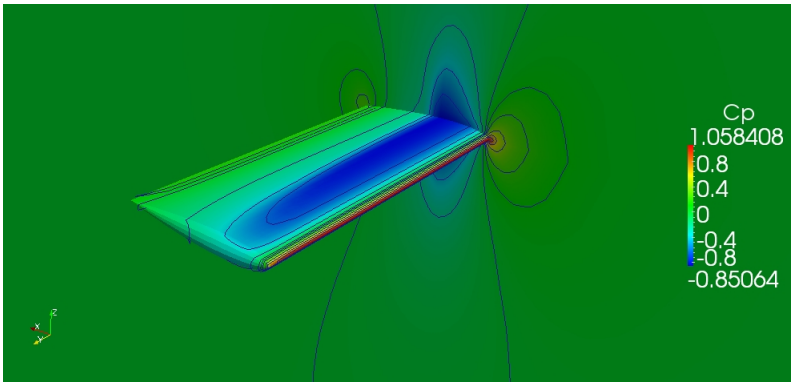


Figure 6.44: C_p contours

The bifurcation point has been computed through a linear flutter analysis by the standard industrial tool MSC.Nastran. The aerodynamic model considered in this case is the Doublet Lattice Method, which works fine for subsonic flows but it presents strong theoretical limitations regarding analyses involving transonic flows. However such a method is considered here for comparing its results to the experimental data, collected in [72], verifing the relative accuracy. The $V_\infty - g$ and $V_\infty - f$ diagrams obtained are shown in figure 6.45.

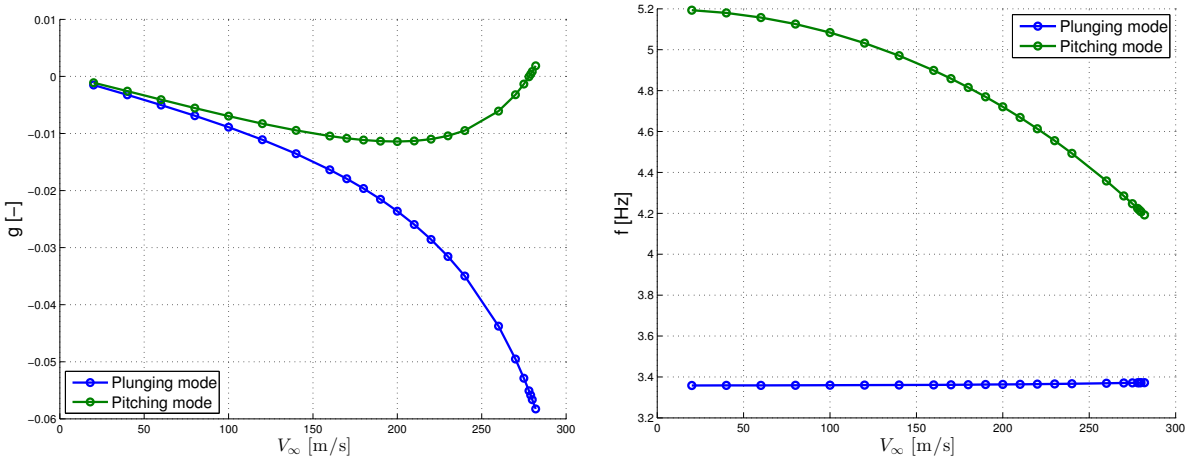


Figure 6.45: Linear flutter analysis for the BACT wing, at $M_\infty = 0.8$

The reduced flutter velocity is equal to $V_f^* = 0.64$, meanwhile the experimental result is slightly smaller, being equal to $V_{f_{\text{exp}}}^* = 0.595$. Nevertheless, the flutter reduced frequency is equal to $k_f = \omega_f b / V_f = 0.0193$, which is in agreement with the experimental value. Therefore, as it has been said in chapter 2, the bifurcation frequency can be accurately estimated by a linear analysis, meanwhile its velocity is overestimated.

6.2.1 Training phase

The training signal considered in this case is again an hybrid signal, adopted also in the previous case. Of course the amplitudes and the frequency ranges are adapted to the case under consideration. In figure 6.46 is shown the input signal, meanwhile in figure 6.47 is shown its frequency distribution.

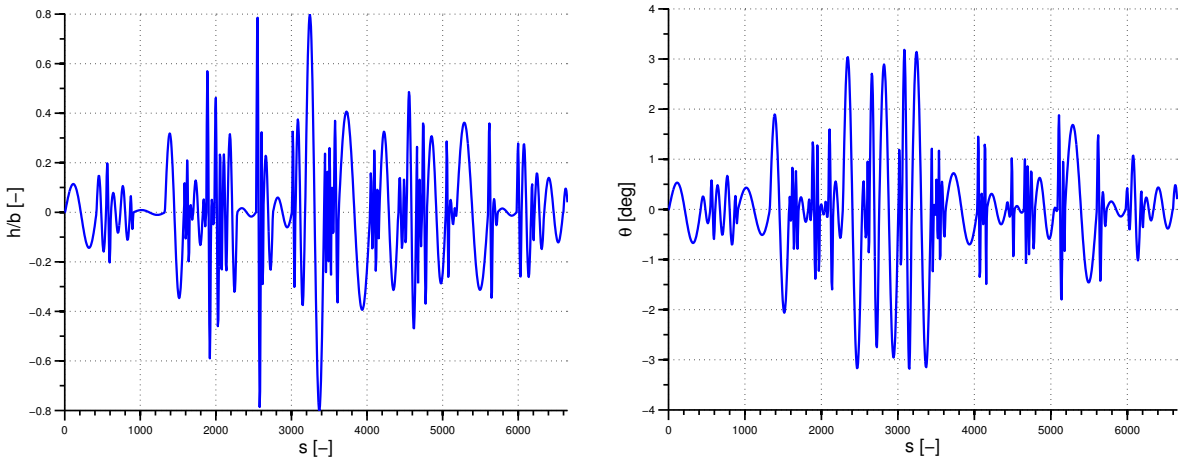


Figure 6.46: Hybrid input for the BACT wing

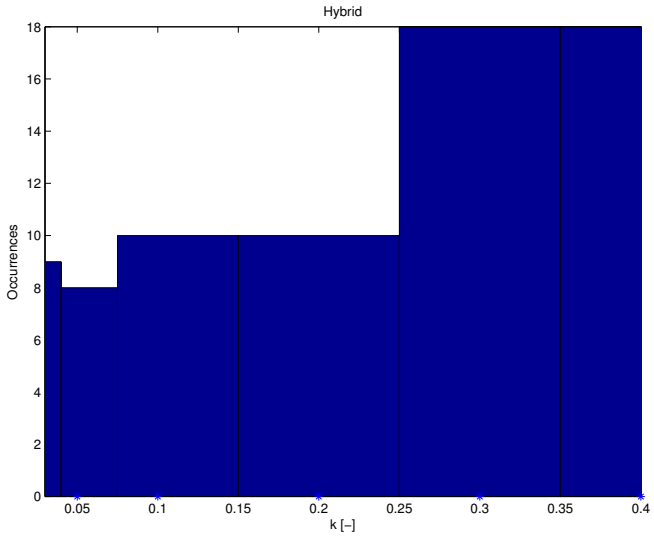


Figure 6.47: Frequency characteristics of the input signal

In this case only the C^{2}TRNN is considered, since the good results obtained previously. The settings of the training algorithm are unaltered from the previous analyses. The CFD simulation is performed with a physical time step of $\Delta t = 2.5 \cdot 10^{-3}\,[s]$, therefore the number of time steps is equal to 2000. The RK5 is employed as time-marching scheme, with the CFL number fixed to 3. The computational time required for generating the training data by AeroFoam is equal to $26\,[h]\,13\,[min]$. The dynamic order of the present network is equal to $n_x = 3$ and $n_h = 5$. The results obtained are shown in figure 6.48.

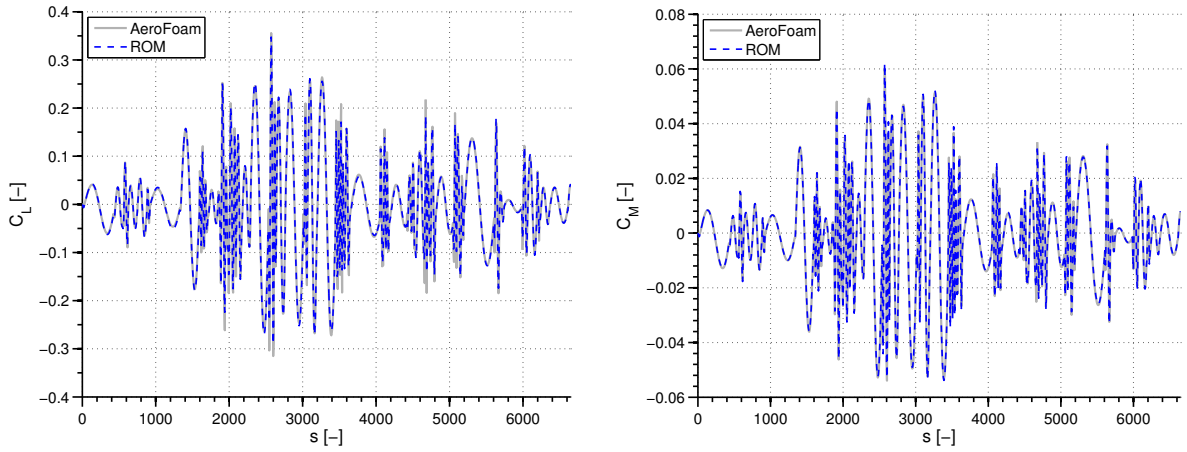


Figure 6.48: Training results for the BACT wing

A very good fit has been obtained with a very small error on both the coefficients, especially for what regard the C_M. The comparison between the different kind of training is shown in table 6.12.

Jacobian matrix computational method	Computational time	Converged F
Finite difference	$12\,[h]\,21\,[min]$	$\mathcal{O}\left(10^{1}\right)$
Analytic	$6\,[h]\,34\,[min]$	$\mathcal{O}\left(10^{-1}\right)$
AD	$6\,[h]\,02\,[min]$	$\mathcal{O}\left(10^{-1}\right)$

Table 6.12: Convergence properties, BACT wing case

Again the finite difference approach has shown its limitation during the network training, producing a very high output error. Instead, the other two approaches are more accurate, presenting a big time saving.

6.2.2 Aeroelastic simulation

The C^{2}TRNN is finally coupled with the mechanical model of the BACT wing, resulting in a 2 DOFs aeroelastic model very similar to the typical section previously treated, with the only difference of a more complicated aerodynamic behaviour, since in the flow are present strong three-dimensional effects. Therefore the dynamic equations which represent the system are the same of 6.16. Also in this case the LCO behaviour is computed through both the time-marching and periodic collocation methods explained in chapter 2. The LCO response obtained at $V^* = 0.65$ is shown in figures 6.49, 6.51 and 6.51 in terms of motion and loads, meanwhile in figures 6.52 and 6.53 is reported the LCO envelope computed. All the tests performed by the time marching method assume the initial condition $h_0/b = 0.2$, $\theta_0 = 0\,[deg]$ and $\dot{\mathbf{u}} = \mathbf{0}$.

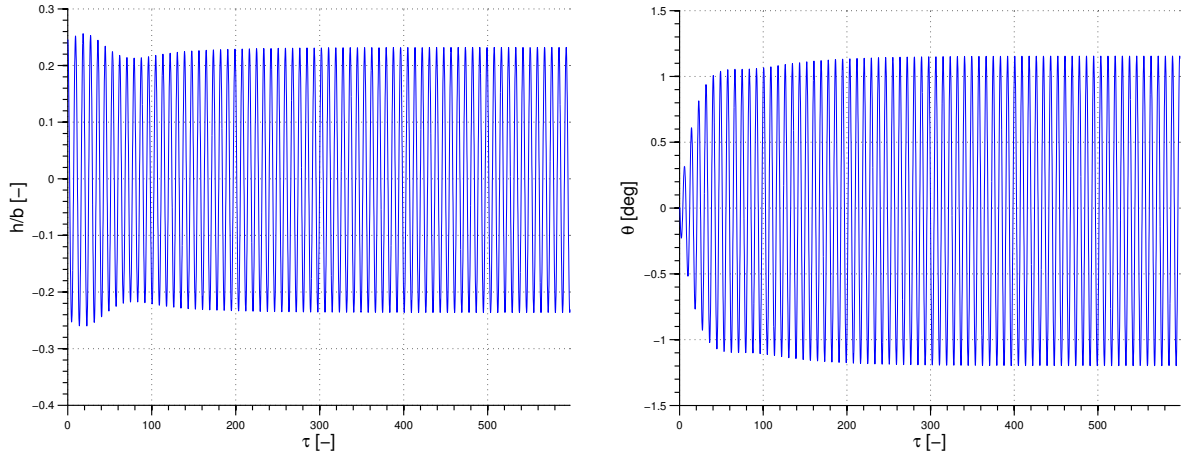


Figure 6.49: LCO response of the BACT wing at $V^* = 0.65$

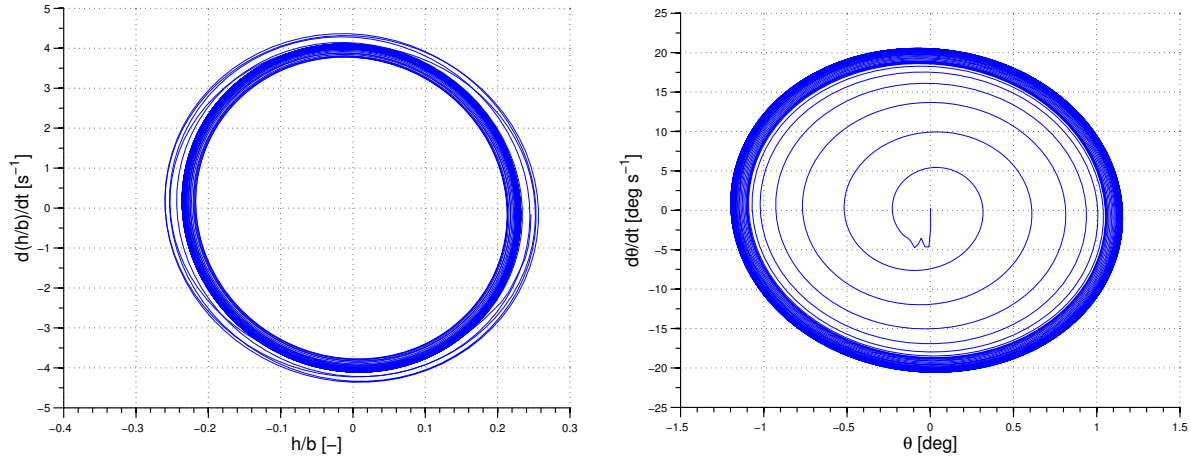


Figure 6.50: LCO response of the BACT wing, shown in the phase space at $V^* = 0.65$

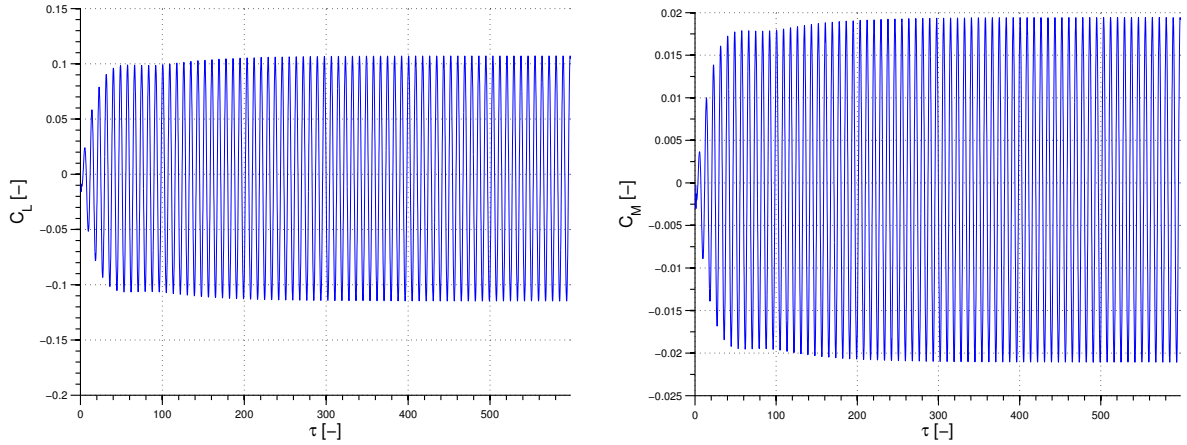


Figure 6.51: Loads acting on the BACT wing at $V^* = 0.65$

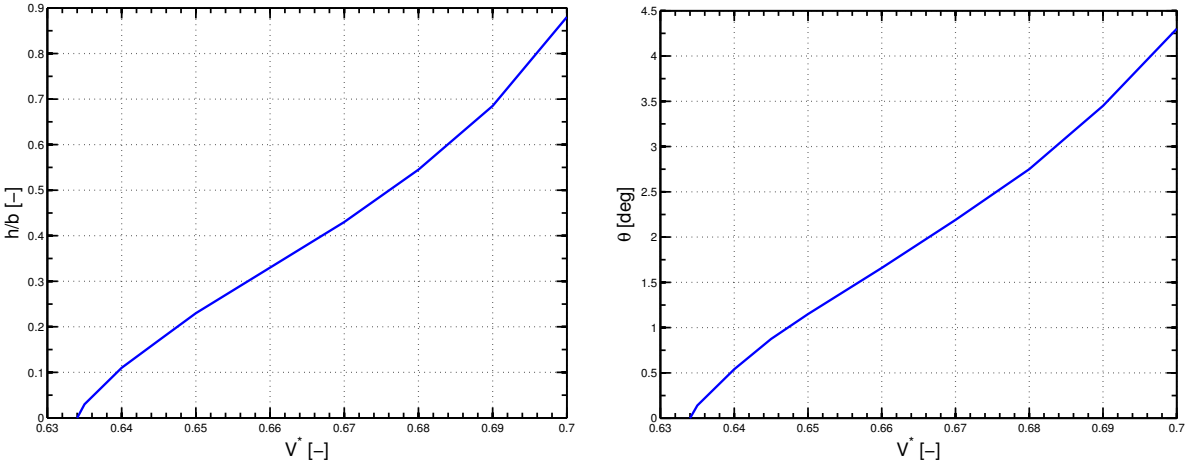


Figure 6.52: LCO envelope of amplitudes versus reduced velocity for the BACT wing

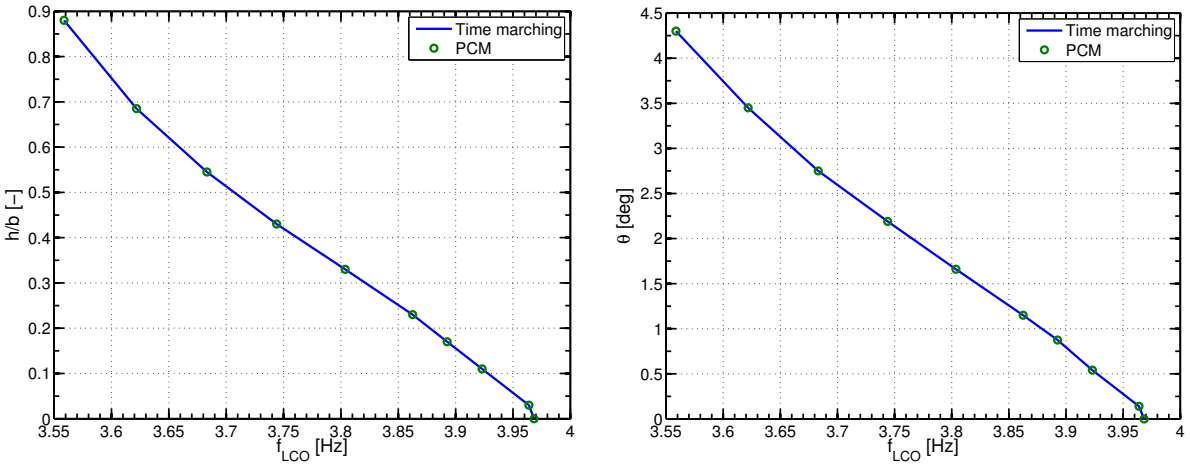


Figure 6.53: LCO envelope of amplitudes versus LCO frequency for the BACT wing

Unfortunately, at this moment there are no useful results from the CFD simulation by AeroFoam, since no LCO behaviour has been obtained testing at various reduced velocity. It can be due to a problem encountered in the aeroelastic interface, since the `Rigid` one has been used for the generation of the training signal, but only the `Modal` interface permits to compute aeroelastic responses, and this latter may present some inaccuracies when the wing motion is not small [13]. However, different test are now under consideration for identifying the problem. So the results shown here are only those obtained through the training of the neural network, without proof of their reliability, since the CFD counterpart results are absent for the time being. However, the bifurcation frequency has been only slightly underestimated by the C^2TRNN , so there are good possibilities that such results present physical meaning. Differently from figure 6.32, in figure 6.53 on the horizontal axis is shown the effective frequency of the LCO, since it gives a direct physical interpretation of the phenomenon. Finally, in table 6.13 is reported the computational time required by the time-marching and periodic collocation methods for computing the LCO envelope. For what regard the PCM, the results computed through the time-marching method are adopted as initial guess, and by a continuation method on the reduced velocity the LCO envelope is computed. The number of intervals employed starts from 100 up to 200 during the finest LCO estimation.

	Time marching	Periodic collocation method
Generation of the training input-output data pair by AeroFoam	26 [h] 23 [min]	
Training time	6 [h] 02 [min]	
Envelope simulation time (10 points)	10 [*min*]	14 [*min*]
Total time required	32 [h] 35 [min]	32 [h] 39 [min]

Table 6.13: ROM-based simulation for the BACT wing, computational time

6.3 Final considerations

The continuous time neural network based ROM has proved to be an efficient tool in the prediction of different LCOs from one only high order simulation of the aerodynamic system. However, not all the parametrizations detailed in chapter 3 have not worked succesfully. The C^2TRNN has shown the best performances in terms of regularity of the results and computational time. Also ECTRNN has worked fine, but in the construction of the LCO envelope it has been discovered that the results are not as regular as the ones shown by the C^2TRNN . This can be due to a not perfect convergence of the training, with the bias terms that influence negatively the response. Instead, the CTRNN.LI has shown the worst performances of the three networks proposed. Its failure in constructing a correct LCO envelope induces to discard this kind of parametrization. This bad behaviour of the network, even if after a good training and generalization phases, can be due to the partial nonlinearity of the network, with the structural motion which appears linearly in the network dynamics. The LCO behaviour has proved to be difficult to capture, in all the cases, with a training signal designed with great care. Perhaps, the too simple structure of the CTRNN.LI has been the main cause of its failure.

Despite the partial failure of the network parametrizations proposed, expecially for what regard the CTRNN.LI , the results obtained are encouraging, since it has been proved that a nonlinear system represented by only (4 structural + 3 aerodynamic) states is able to capture the same nonlinear behaviour of a system represented, in the case of the typical section, by (4 structural + 48800 aerodynamic) states, reducing the computational time required in the computation of the LCO envelope by 1 orders of magnitude, with respect to the CFD method.

The CFD computational cost could be further reduced by optimizing the time step employed and the different parameters of the DTS method. Also the computatational time of the ROM can be efficiently reduced, exploiting the parallelization properties of the GA, which tipically requires the 60% of the total training time as said previously. The refinement part of the algorithm, played by the LM algorithms, has shown to be very efficient, with a very fast convergence, thus a further optimization doesn't seem to be required.

The adoption of AD algorithms during the network training has not introduced a great advantage in terms of computational time. However a slight reduction has been proved, and a further optimization of the algorithm can enhance the efficiency of the training. All the networks have proved to work correctly even with an integration time step much more greater than the one used during the training.

The PCM has shown to be more than an alternative to the time-marching method in the computation of LCO behaviours since it has been able to capture such a phenomenon on a relatively large range of the initial guess of the LCO period.

Chapter 7

Conclusions and possible future developments

7.1 Conclusions

In this work it has been detailed the development of a nonlinear Reduced Order Model (ROM) of an aerodynamic system, starting from a training data pair generated by a CFD code and ending with a complete aeroelastic simulation of a Limit Cycle Oscillation (LCO). The proposed ROM is based on Continuous Time Recurrent Neural Network (CTRNN), and different parametrizations have been considered, since no standard framework has been estabilished until nowadays for the representation of nonlinear dynamic systems.

Some strength and weaknesses with respect to the others ROM techniques nowadays present in the literature have been noticed: a neural network approach reduces the user choice to the model order and to the training signal generation, without choosing any *snapshots* (characteristics of the POD technique) of the full order solution. Thus in this sense the complexity of the ROM design has been reduced. With respect of an HB approach there is the greater generality of a neural network: not only time periodic solution can be computed, but also responses to generic inputs, also non-periodic. CTRNN and Volterra series are actually very similar, but the neural network formulation seems to be more robust to model uncertainties, since the latter one is composed by a series of neurons, and the fault of one neuron does not necessarily degrade the output, meanwhile the Volterra series has shown to be quite sensitive to such modeling. Of course the main drawbacks of the neural network approach is the presence of the training phase, absent in the other approaches, which increases the computational time required for constructing the ROM. The generation of a training signal is a problem common for all the ROM techniques (except HB) and it is still an open issue [1].

Regarding the CTRNN performances, it has been found out that the level of accuracy depends on the type of nonlinearities considered in the model, on the order of the network, and in the case of the LCO simulation on the quality of the training signal. Moreover, in the latter case it has been discovered that only training signal with a very wide spectrum of frequencies are able to excite all the nonlinearities required by the aerodynamics to reproduce the LCO of the aeroelastic system. Instead, a limited influence is associated with the amplitude distribution of the signal, and also the network dynamic order seems to be not critical in its design. For what regard the particular application of this work, good results have been obtained with a dynamic model of the III order, but also models with four states have been tested, with no appreciable improvement of the results. Models with order smaller than three has not been able to capture the LCO behaviour. A greater influence has been noticed on the number of hidden neurons, represented in this work by the parameter n_h: increasing the number of hidden neurons, while mantaining constant the number of dynamic states, has been proved to increase the accuracy of the CTRNN, as shown in chapter 5.

The CTRNNs have shown to be quite robust to variations of the reduced velocity, therefore a

CTRNN constructed on the results of a pure aerodynamic simulation can be employed in an aeroelastic system at various reduced velocities, mantaining the Mach number fixed. Moreover it has proved to be quite insensitive to various perturbations in the initial condition of the time marching problem. The evolution toward the LCO conditions has been shown to be quite different from the ones obtained by the CFD simulations. This can be due to the different nonlinear eigenstructure of the two systems: the CTRNN can be able to capture the main nonlinearities of the high order system, but is can present a sligthly different behaviour with respect of the CFD model near the perturbation point.

Even if the results obtained are slightly different from the ones present in the literature, the good fit with respect of the CFD results available is promising. One of the results of this work is the demonstration that a small order dynamic system is able to capture the main nonlinearities of a phenomenon which has been obtained with a (very) large dynamic system. Moreover, it has been proved again that a continuous time formulation of a neural network permits to work with a wide range of time step amplitudes employed in the time marching scheme, where the discrete time version is forced to work with the same time step employed in the training phase, as shown in [31].

It is interesting to note that the LCO solution has been obtained also through a periodic collocation method. Such a result is relatevely new in the field of nonlinear aeroelasticity with a nonlinear aerodynamic model, since until now such technique has been employed for concentrated structural nonlinearities. The results obtained shown that such a technique can be efficiently used in the evaluation of the LCO envelope, with a computational time comparable to the one required for the direct time marching simulation. It is worth to note that the considered approach can be adopted only for continuous time models, therefore enhancing an other feature of the CTRNNs presented here against the DTRNNs found in the literature.

The training procedure based on Automatic Differentiation (AD) algorithms has shown to be quite effective from the point of view of the computational time and in some case also in the final accuracy of the results. However, a great improvement of the performances with respect of the ones obtained with the analytical expression of the Jacobian matrix has not been obtained: since the straightforward procedure detailed in this work for the computation of this latter, the training can be accomplished even without the AD tools. However a code optimization can be considered in order to improve the results of the AD training. The main conclusion that can be drawn from the convergence analysis of the training algorithm is that the finite difference approach should be avoided if it is possible, since the great loss in accuracy accounted in chapter 6. The computation of the analytical Jacobian matrix has been shown to be quite easy, therefore this fact should be exploited. The hybrid training algorithm considered from chapter 3 has shown to be quite effective, with a relative small computational time. However such performance can be improved as said previously, considering a parallelization of the Genetic Algorithm employed in the initialization phase, since it occupies about the 60% of the training time. No particular modification may be considered about the Levenberg-Marquardt algorithm, since it has proved to be quite fast, even if the solution given as output of the initialization phase was not very close to the optimum solution of the training.

The efficiency of the presented ROM is very attracting, since with a limited computational effort, much smaller than a full order analysis, it makes possible the inclusion of nonlinear dynamic behaviours even in the first stages of an aircraft design, mantaining the same level of accuracy of the solution computed by a CFD approach to the representation of the aerodynamics, which use is limited by its run-time requirements. The computational efficiency of the proposed ROM technique have improved of one order of magnitude the one obtained with the CFD computation. However, as pointed out in different sources in the literature, for example [74], along with the limitation related to the run-time of the simulation, the application of Computational Aeroelasticity (CA) techniques is still hampered by the CFD tools credibility for unsteady numerical simulations. But the research in this field is very active and fruitful, expecially nowadays,

therefore great improvements are expected, with positive consequences to the employment of ROMs.

7.2 Future developments

Different possibilities can be considered as future developments of the ROM proposed. First of all, it should be compared to a in-house discrete time neural network, since it seems the main stream today followed in the literature [31, 32, 33]. The formulation of the problem in a discrete framework, expecially with the adoption of RBF neurons has proved to be quite efficient, because of the very fast training phase, associated to a linear least square problem related with such a formulation. However, also a continuous time neural network has proved its efficacy in this work. Moreover such a representation allows to consider the periodic collocation method in the search of LCO behaviours. Different consideration may be done at this level. First of all, there is the possibility to generalize the fomulation of the CTRNN with the following model:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{W}^x \mathbf{\Phi} \left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u} \right) \\ \mathbf{y} = \mathbf{W}^c \mathbf{x} + \mathbf{W}^d \mathbf{u} \end{cases}$$

therefore including a term of direct transmission from the input to the output. Moreover, instead of fixing the structure of $\mathbf{W}^c$, its terms may be considered as unknowns. In this manner there is the possibility that a good fit of the training data is obtained also with a smaller number of states and hidden neurons. A similar variant considers the series of a linear and nonlinear representation of the system: first the best linear approximation of the system is carried out, for example with a subspace projection techinque, considering only small perturbations to the input in the training phase. Then the obtained results is taken as initial guess for the following nonlinear representation:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{W}^x \mathbf{\Phi} \left(\mathbf{W}^a \mathbf{x} + \mathbf{W}^b \mathbf{u} \right) \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{cases}$$

starting with a null initial guess for the nonlinear terms. With such an approach, the resulting nonlinear system behaves at least as good as the linear representation, reducing the required time of the training phase, since it has be seen that the major part of this burdensome phase was due to the initialization of the network weights.

A second, and maybe more interesting consideration is the possibility to train a discrete time model, characterized by a fast training phase, and with a Tustin-like transformation obtaining a continuous time model. However the adoption of a RBF network leads to consider a NARX representation of the system, losing the state space formulation that has characterized the work here presented.

As concluding remark, the prediction of the LCO behaviour of a typical section is just the beginning of this work. As shown previously, the simulation of LCO responses for the Benchmark Active Control Technologies (BACT) wing is currently under consideration, but the hope is to obtain a ROM of a flexible wing capable of predicting LCO behaviours due to both structural and aerodynamic nonlinearities, including also the effect of a control system (maybe nonlinear as well).

Bibliography

[1] D. J. Lucia, P. S. Beran, and W. A. Silva. Reduced-order modeling: new approaches for computational physics. *Progress in aerospace sciences*, 40:51–117, 2004.

[2] P. Mantegazza. *"Bigino" di dinamica e controllo delle strutture aerospaziali*. `www.aero.polimi.it`, 2012.

[3] E. H Dowell, R. Clark, D. Cox, H. C. Curtiss Jr., J. W. Edwards, K. C. Hall, D. A. Peters, R. Scanlan, E. Simiu, F. Sisto, and T. W. Strganac. *A Modern Course In Aeroelasticity*. Kluver Academic Publishers, 2004.

[4] J. Katz and A. Plotkin. *Low-speed aerodynamics: from wing theory to panel methods*. McGraw-Hill, 1991.

[5] R. L. Bisplinghoff, H. Ashley, and R. L. Halfman. *Aeroelasticity*. Dover, 1955.

[6] J. G. Leishman. *Principles of Helicopter Aerodynamics*. 2004.

[7] K. L. Roger. Airplane math modeling methods for active control design. Technical report, AGARD, 1977.

[8] M. Karpel. Design for active flutter suppression and gust alleviation using state-space aeroelastic modeling. *Journal of Aircraft*, 3:221–227, 1982.

[9] G. Pasinetti and P. Mantegazza. Single finite states modeling of aerodynamic forces related to structural motions and gusts. *AIAA Journal*, 37:604 – 612, 1999.

[10] J.G. Leishman and G. Crouse. A state-space model of unsteady aerodynamics in a compressible flow for flutter analysis. *AIAA Paper*, 89-0022, 1989.

[11] W. Silva. *Discrete-time linear and nonlinear aerodynamic impulse responses for efficient CFD analyses*. PhD thesis, College of William & Mary, 1997.

[12] D. E. Raveh. Reduced-order models for nonlinear unsteady aerodynamics. *AIAA Journal*, 39(8):1417–1429, 2001.

[13] G. Romanelli and E. Serioli. Un approccio libero alla moderna aeroelasticità computazionale. Master's thesis, Politecnico di Milano, 2008.

[14] L. Sirovich. Turbulence and the dynamic of coherent structures, part I: Coherent structures. *Quarterly of applied mathematics*, 45:561–571, 1987.

[15] K. Willcox and J. Peraire. Balanced model reduction via proper orthogonal decomposition. *AIAA Journal*, 40:2323–2330, 2002.

[16] W. H. Schilders, H. A. van der Vorst, and J. Rommes. *Model Order Reduction: Theory, Research Aspects and Applications*. Springer, 2008.

[17] J. P. Thomas, E. H. Dowell, and K. C. Hall. Using automatic differentiation to create a nonlinear reduced-order-model aerodynamic solver. *AIAA Journal*, 48:19–24, 2010.

[18] A. C. Antoulas. *Approximation of large-scale dynamical systems*. Springer, 2004.

[19] J. P. Thomas, E. H. Dowell, and K. C. Hall. Nonlinear inviscid aerodynamic effects on transonic divergence, flutter, and limit-cycle oscillations. *AIAA Journal*, 40:638–646, 2002.

[20] J. Blazek. *Computational Fluid Dynamics: Principles and Applications*. Elsevier, 2001.

[21] B. Epureanu and E. Dowell. Compact methodology for computing limit-cycle oscillations in aeroelasticity. *Journal of Aircraft*, 5:955–963, 2003.

[22] A. Gelb and W. E. Vander Velde. *Multiple-input describing functions and nonlinear system design*. McGraw-Hill Book Company, 1968.

[23] M. Manetti, G. Quaranta, and P. Mantegazza. Numerical evaluation of limit cycles of aeroelastic systems. *Journal of Aircraft*, 46:1759 – 1769, 2009.

[24] K. C. Hall, J. P. Thomas, and W. S. Clark. Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40:879–886, 2002.

[25] S. Haykin. *Neural networks and learning machines*. Pearson International Edition, 2009.

[26] O. Nelles. *Nonlinear system identification*. Springer, 2001.

[27] Rihab Khalid Al Seyab. *Nonlinear Model Predictive Control Using Automatic Differentiation*. PhD thesis, Cranfield University, 2006.

[28] C. Kambhampati, F. Garces, and K. Warwick. Approximation of non-autonomous dynamic systems by continuous time recurrent neural networks. *IEEE transactions on neural networks*, pages 64–69, 2000.

[29] B. A. Pearlmutter. Learning state-space trajectories in recurrent neural networks. *Neural computation*, 1:263–269, 1989.

[30] J. A. K Suykens and J. Vanderwalle. Learning a simple recurrent neural state space model to behave like chua's double scroll. *IEEE transactions on circuits and systems - I*, 42:499–502, 1995.

[31] W. Zhang, B. Wang, and Z. Ye. High efficient numerical method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order-model. In *51st AIAA, ASME, ASCE, AHS, ASC Structures, Structural Dynamics and Material Conference*, 2010.

[32] G. Chen, Y. Zuo, J. Sun, and Y. Li. Support-Vector- Machine based reduced-order-model for limit cycle oscillation prediction of nonlinear aeroelastic system. *Mathematical problems in engineering*, 2012, 2012.

[33] W. Yao and M. Liou. Reduced-order modeling for flutter/LCO using recurrent artificial neural network. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM*, 2012.

[34] Y. Nakamura and M. Nakagawa. Approximation capability of continuous time recurrent neural networks for non-autonomous dynamical systems. In *Artificial Neural Networks - ICANN*, 2009.

[35] M. Stephanson. *An Adaptive, Black-Box Model Order Reduction Algorithm Using Radial Basis Function*. PhD thesis, Ohio State University, 2012.

[36] S. Timme, S. Marques, and Badcock K. J. Transonic aeroelastic stability analysis using a Kriging-based Schur complement formulation. *AIAA Journal*, 49:1202 – 1213, 2011.

[37] S. Timme and J. K. Badcock. Transonic aeroelastic instability searches using sampling and aerodynamic model hierarchy. *AIAA Journal*, 49:1191 – 1201, 2011.

[38] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of global optimization*, 13:455–492, 1998.

[39] S. A. Morton and P. S. Beran. Hopf-bifurcation analysis of airfoil flutter at transonic speed. *Journal of Aircraft*, 36:421–429, 1999.

[40] P. Cvitanovic', R. Artuso, R. Mainieri, G. Tanner, and G'bor Vattay. *Chaos: Classical and Quantum, I: Deterministic Chaos*. ChaosBook.org, 2009.

[41] R. Seydel. *From equilibrium to chaos: practical bifurcation and stability analysis*. Elsevier, 1988.

[42] S. Sastry. *Nonlinear systems: analysis, stability and control*. Springer, 1999.

[43] D. W Jordan and P. Smith. *Nonlinear ordinary differential equations*. Oxford University Press, 2007.

[44] L. Dinjun, W. Xian, Z. Deming, and H. Maoan. *Bifurcation theory and methods of dynamical systems*. World Scientific Publishing Co., 1997.

[45] B. H. K. Lee, S. J. Price, and Y. S. Wong. Nonlinear aeroelastic analysis of airfoils: bifurcation and chaos. *Progress in aerospace sciences*, 35:205 – 334, 1999.

[46] R. W. Bunton and C. M. Denegri. Limit cycle oscillations characteristic of fighter aircraft. *Journal of Aircraft*, 37:916–918, 2000.

[47] Z. Zhang, S. Yang, F. Liu, and D.M. Schuster. Prediction of flutter and LCO by an Euler method on non-moving cartesian grids with boundary-layer corrections. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[48] M. Borri and P. Mantegazza. Nonlinear oscillation and limit cycles: A numerical approach by finite elements in time domain. In *Sixth International Conference on Mathematical Modelling*, 1987.

[49] M. Manetti, G. Quaranta, and P. Mantegazza. Numerical evaluation of limit cycles of aeroelastic systems. In *IFASD*, 2009.

[50] P. A. Thompson. *Compressible-fluid dynamics*. 1988.

[51] F. Sabetta. *Gasdinamica*. 1999.

[52] L. Quartapelle. *Godunov method and related schemes for hyperbolic problems*. `www.aero.polimi.it`, 2011.

[53] G. Romanelli. *Computational Aeroservoelasticity of Free-Flying Deformable Aircraft*. PhD thesis, Politecnico di Milano, 2012.

[54] Various Authors. *OpenFOAM v.1.7.1 User's Guide*, 2010.

[55] A. Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA Journal*, 91-1596, 1991.

[56] A. Brandt. Multi-Level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31:333 – 390, 1977.

[57] A. Quarteroni. *Modellistica numerica per problemi differenziali*. Springer, 2008.

[58] M. Mattaboni, G. Quaranta, and P. Mantegazza. Active flutter suppression for a three-surface transport aircraft by recurrent neural networks. *Journal of guidance, control and dynamics*, 32:1295–1307, 2009.

[59] F. Bernelli-Zazzera, P. Mantegazza, G. Mazzoni, and M. Rendina. Active flutter suppression using recurrent neural networks. *Journal of guidance, control and dynamics*, 23:1030–1037, 2000.

[60] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependecies with gradient descent is difficult. *IEEE transactions on neural networks*, 5:157–166, 1994.

[61] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 2:431–441, 1963.

[62] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, 1989.

[63] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. `http://matrixcookbook.com`, 2008.

[64] A. Griewank and A. Walther. *Evaluating derivatives - Principles and Techniques of Algorithmic Differentiation*. Siam, 2008.

[65] A. Quarteroni, F. Sacco, and F. Saleri. *Matematica numerica*. Springer, 2005.

[66] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press, 1981.

[67] A. Griewank, D. Juedes, and J. Utke. ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM TOMS*, 22:131–167, 1996.

[68] W. Haase, V. Selmin, and B. Winzell. *Progress in computational flow-structure interaction: results of the Project UNSI, supported by the European Union 1998-2000 (Notes on Numerical Fluid Mechanics and Multidisciplinary Design)*. Springer, 2002.

[69] S. R. Bland. AGARD two-dimensional aeroelastic configurations. Technical report, AGARD, 1979.

[70] P. Masarati, M. Lanz, and P. Mantegazza. Multistep integration of ordinary, stiff, and differential-algebraic problems for multibody dynamics applications. In *XVI Congresso nazionale AIDAA, Palermo*, 2001.

[71] M.R. Waszak. Modeling the benchmark active control technology wind-tunnel model for application to flutter suppression. *AIAA Journal*, 96-3437, 1996.

[72] J.A. Rivera, B.E. Dansberry, R.M. Bennet, M.H. Durham, and W.A. Silva. NACA 0012 benchmark model experimental flutter results with unsteady pressure distributions. *AIAA Journal*, 92-2396:1898 – 1908, 1992.

[73] C. H. Stephens. CFD-based aeroservoelastic predictions on a benchmark configuration using the transpiration method. Master's thesis, Oklahoma State University, 1998.

[74] R. M Bennet and J. W. Edwards. An overview of recent developments in computational aeroelasticity. In *Proceedings of the* 29^{th} *AIAA Fluid Dynamics Conference*, 1998.