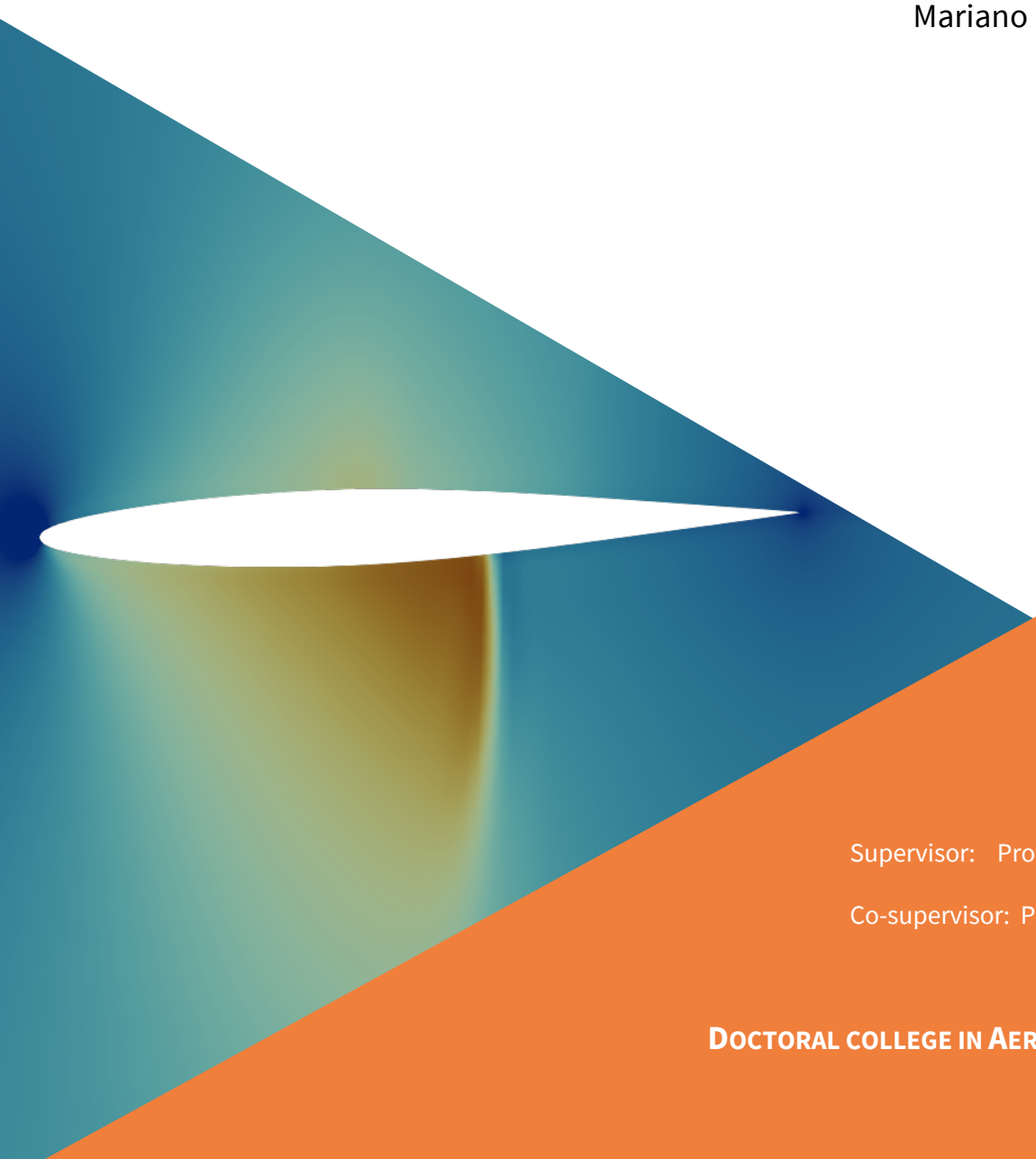
LIÈGE université
Sciences Appliquées

Efficient optimisation of nonlinear time-periodic aeroelastic problems

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy (PhD) in Engineering Science

by

Mariano SÁNCHEZ MARTÍNEZ



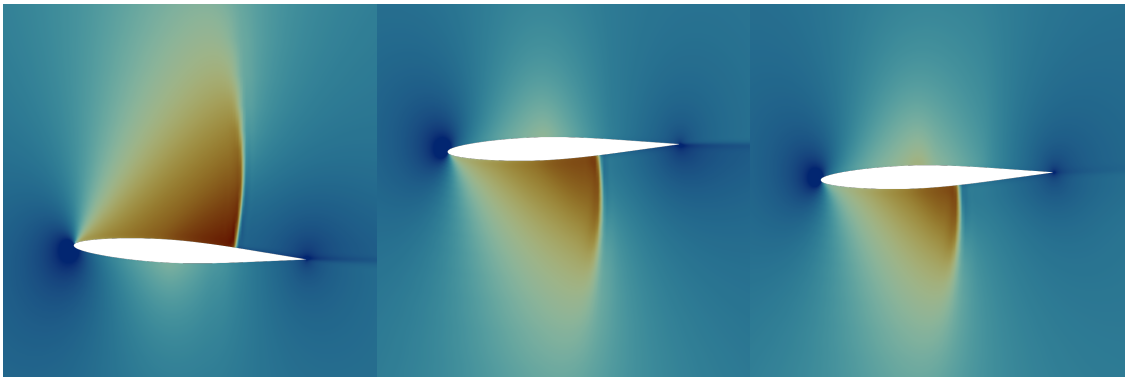
Supervisor: Prof. Grigorios DIMITRIADIS

Co-supervisor: Prof. Vincent E. TERRAPON

DOCTORAL COLLEGE IN AEROSPACE AND MECHANICS

OCTOBER 2024

Efficient optimisation of nonlinear time-periodic aeroelastic problems



Université de Liège - Faculté des Sciences Appliquées

Members of the jury:
Prof. Grigorios Dimitriadis, Université de Liège (supervisor)
Prof. Vincent Terrapon, Université de Liège (co-supervisor)
Prof. Pierre Duysinx, Université de Liège
Dr. Romain Boman, Université de Liège
Dr. Ing. Markus Ritter, Deutsches Zentrum für Luft- und Raumfahrt
Prof. Rafael Palacios, Imperial College London

Thèse réalisée en vue de l'obtention du titre de Docteur en Sciences Appliquées par

Mariano Sánchez Martínez

Année académique 2024-2025

A la memòria dels meus iaios i iaies

Abstract

Optimisation of unsteady fluid-structure interaction problems is of increasing importance in aeronautical design and other fields of engineering. In order to apply gradient-based optimisation methods, the gradients of objective functions with respect to design parameters have to be evaluated. Depending on the required level of modelling fidelity, such calculations can become very computationally expensive or even impractical. Therefore, lower-fidelity aerodynamic modelling methods are usually used, especially in earlier phases of the design process.

This thesis introduces a new adjoint harmonic balance approach for the optimisation of nonlinear, time-periodic fluid-structure interaction problems. The harmonic balance technique reduces the computational cost of the simulations, allowing the use of higher-fidelity methods. The adjoint equations for this approach are then derived to efficiently obtain the gradients of few objective functions with respect to many design parameters.

The harmonic balance method, including a novel frequency iteration technique, is applied to a test case of a pitch-plunge aerofoil undergoing limit cycle oscillations in transonic flight conditions. For this test case, the harmonic balance is 11 times faster compared to a time-marching approach.

The accuracy of the adjoint harmonic balance equations is verified by comparing the gradients of 2 objective functions with respect to 31 design variables in the limit-cycle oscillation test case. The calculation of gradients using the adjoint approach is approximately 30 times faster than finite differences.

Finally, the adjoint harmonic balance technique is used to optimise four aeroelastic test cases. Two cases are unconstrained, one uses a steady constraint and another an unsteady constraint. At the end of all four optimisation processes the objective function significantly improves, which shows that the proposed method can be used to reduce the computational cost of high-fidelity aeroelastic optimisation, paving the way for its practical use in early design phases.

Acknowledgements

This thesis would have been impossible without the support and encouragement from many people. First, I would like to thank my supervisor, Greg. His guidance and suggestions have been crucial for the success of this work. Furthermore, I would also like to express my gratitude to my co-supervisor, Vincent. His feedback has been invaluable on many, many occasions and learning how to explain my work in different contexts was a very important exercise. I acknowledge the members of the jury who have so kindly accepted to form part of it.

I also need to thank my colleagues in the AEA and MTFC groups. The discussions we had were a critical component of the evolution of this thesis. Furthermore, I would like to thank Adrien for his help.

This project builds on work by David Thomas and Rubén Sánchez. I sincerely thank them for it and their help during the earlier stages of the project.

I am also grateful to my parents and family, for their encouragement and for listening to my thoughts and complaints. Also, to all the people who have put up with me in the past few years. Without their support the most difficult times would have been much worse.

Some computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region. Part of the work was carried out under a Fonds pour la Recherche en Industrie et Agriculture (FRIA) grant provided by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS).

Contents

List of Figures

List of Tables

Chapter 1

Introduction and motivation

The present thesis deals with the optimisation of unsteady fluid-structure interaction problems. The current chapter starts by explaining what they are, why they are important and how they are studied. It also introduces optimisation, justifies the need for its application to fluid-structure interaction problems and discusses some of the difficulties of this application. Then, the objective of the work is presented with some key questions that should be answered. Finally, the main contributions of the present work to the state of the art and the outline of the document are briefly discussed.

1.1 Fluid-structure interaction in engineering

Fluid-structure interaction (FSI) refers to a class of problems in which a solid structure, such as an aircraft wing, is immersed in a flow, such as moving air. The presence of the structure leads to changes in the flow characteristics. The fluid then induces loads in the structure. These loads deform the solid, the movement of which in turn modifies the flow. Among FSI problems, there are both steady and unsteady cases.

In the 1940s, Collar proposed a way to classify phenomena of interest to aircraft designers depending on the nature of the forces involved [1]. This came to be known as Collar's aeroelastic triangle, shown in Fig. 1.1. It includes four classifications: flight mechanics, structural dynamics, static aeroelasticity and dynamic aeroelasticity.

Classical flight mechanics considers aerodynamic loads and inertial forces only, making the assumption that the aircraft is rigid or that the frequencies of its flexible motion are much higher than those of its rigid motion. Understanding the behaviour and stability of aircraft is crucial for a good design. For example, instability in one

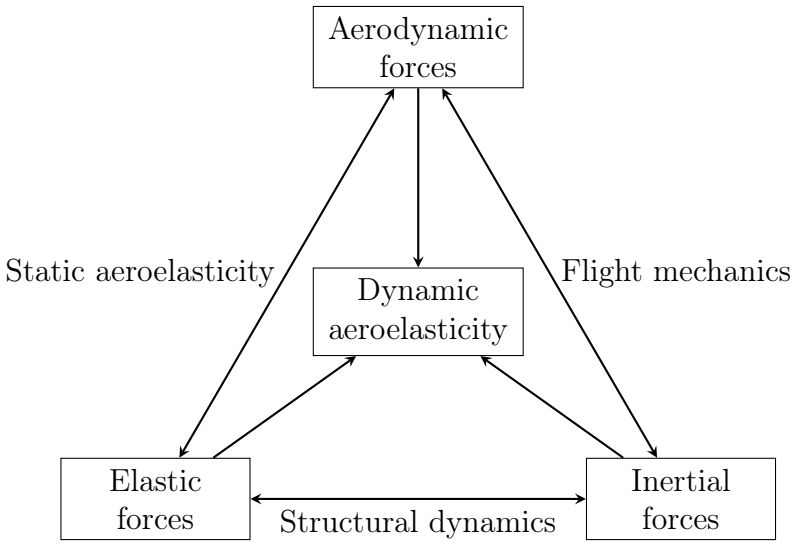


Figure 1.1: Collar's triangle [1]

of the rigid modes of an aircraft, such as the phugoid, would be very detrimental to the aircraft.

If the structure is submitted to inertial and elastic forces, it is in the domain of structural dynamics. One important phenomenon is the effect of impacts on the structure. For example, in the case of aircraft, landings will introduce a sudden change in the load distribution. This change will induce some movement, which should damp out. The structure must be designed to withstand many impacts with the ground without breaking.

Static aeroelasticity studies the interaction between aerodynamic and elastic forces. One example of engineering importance is that of static divergence. Under some flow conditions, the destabilising aerodynamic loads can be higher than the structural restoring loads, leading to structural failure.

Another, less destructive, example of the application of static aeroelasticity is the flight shape of a wing. While the wing has a specific shape when built, the loading required to keep the aircraft in the air leads to a deformation. This change in shape also modifies the aerodynamic characteristics of the wing. It is the aerodynamic properties of the deformed shape during cruise and manoeuvre conditions that are the most important when calculating the fuel burn. Therefore, it is necessary to predict this deformed shape in order to evaluate performance properly. Figure 1.2 shows a comparison between an undeformed wing and its corresponding flight shape in cruise conditions.

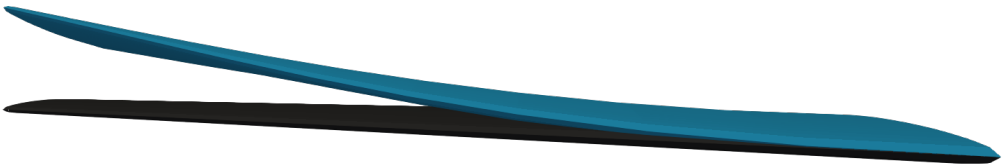


Figure 1.2: The flight shape of a wing, in blue, compared to its undeformed shape, in black. The root is to the right and the tip to the left. Both bending and torsion can be observed, especially close to the tip.

Dynamic aeroelasticity adds inertial forces to static aeroelasticity. Turbulent gusts in the atmosphere are one phenomenon where inertia is relevant. They are inherently unsteady and so is the resulting response of an aircraft. Furthermore, they are aperiodic.

Another example of the application of dynamic aeroelasticity is flutter. Take an aerofoil, or other slender body, submerged in a flow. If appropriately designed, the total damping of the coupled aero-structural system is positive at low speeds, thus reducing the amplitude of any oscillations that may occur. When increasing the airspeed, the damping may increase but it may also decrease, until it reaches a point where it is exactly zero. This is the *flutter* point. Considering only small amplitudes, such that the response is linear in nature, any disturbance will result in undamped periodic oscillations. At the flutter point, these oscillations will continue indefinitely at a constant amplitude. Beyond it, they will show exponential growth in amplitude due to negative damping.

However, aeroelastic systems may feature both structural and fluid nonlinear behaviours. In the 1950s, Woolston et al. used an analogue computer in order to study the influence of structural nonlinearities on flutter. They found that cubic hardening of the structure led to a movement with limited amplitude in post-flutter flight conditions [2]. The resulting oscillations are known as *limit-cycle oscillations* or LCOs. If the LCO amplitude is too large, it leads to very high structural loads and material fatigue. Therefore, predicting the nonlinear behaviour of the system can help improve designs and avoid structural failure.

Furthermore, there has been recent interest in using structures undergoing LCOs for energy harvesting in order to generate electricity. For example, by immersing a kite made of a piezoelectric material in high winds, its aeroelastic deformation generates a current. This current can be transmitted to the ground and used to power machinery in isolated areas.

Experiments can be used to study LCOs in a controlled environment. Another

option is numerical modelling. One possible numerical approach is to use time-marching to study the evolution of the differential equations in time. Another option is to use frequency-domain methods. After the results of Woolston et al., Shen [3] applied a technique developed by Krylov and Bogoliubov [4] for nonlinear mechanics. This method assumes a periodic oscillation, expanding the solution into a series of harmonics. Each harmonic is represented as an equation with a source term introduced by the nonlinear behaviour. These equations have to be balanced to find the correct solution of the problem. Therefore, the method was called the *harmonic balance* [3].

With increasing computational power, there has been a corresponding increase in the use of higher-fidelity models for FSI phenomena. This has allowed the prediction of limit-cycle oscillations in the transonic regime. However, time-marching simulations are still computationally expensive. In order to reduce the computational cost of the simulation of LCOs, the harmonic balance has been implemented in FSI codes.

There are two classes of FSI codes depending on the number of solvers used: monolithic and partitioned. Monolithic approaches consider the flow and the structure to be part of the same problem. Meanwhile, in a partitioned code the fluid and solid parts of the problem are treated as different fields and calculated by different solvers. Partitioned codes are more flexible, allowing the user to change which codes are used according to the desired level of fidelity. For higher-fidelity FSI simulations, a partitioned approach is usually used.

1.2 Design process in fluid-structure interaction

There are several choices to be made in the design process. One of them is the level of fidelity used to model the aeroelastic problem. There is a trade-off between fidelity and computational cost. On the one hand, using lower-fidelity fluid models, such as linear potential solvers, results in fast calculations of flow solutions. On the other hand, higher-fidelity approaches, such as the Euler or Reynolds-averaged Navier-Stokes equations, model important fluid phenomena that can affect the aerostructural solution.

Traditionally, aeroelastic design uses lower-fidelity fluid models, especially in earlier phases of the process. Since they are faster, it is possible to evaluate many different designs using many design parameters quickly. By accelerating higher-fidelity aeroelastic calculations, their use in earlier phases of the design process could be enabled, which could save time in later stages.

A design can be improved through an optimisation process, during which an

objective function is minimised or maximised by changing the values of some design variables subject to some constraints. For aerostructural design, some examples of objective functions are the lift-to-drag ratio of a wing, its mass or the fuel burn of an aircraft. The optimal designs can be obtained by modifying the shape of the wing, by changing its structural properties or by a combination of aerodynamic and structural design modifications. The final designs have to meet, among others, structural failure and flutter constraints.

Another choice is the kind of optimisation algorithm selected to improve the design. Optimisation algorithms can be classified depending on whether they are gradient-based or gradient-free. Gradient-free methods, such as evolutionary algorithms, let the designer study the whole design space in order to find a better solution but they cannot converge to an optimum. They are also useful when dealing with non-smooth objective functions. Gradient-based approaches, the most basic of which is the steepest-descent approach, converge to a local optimum. They require obtaining the gradient of the objective function and constraints with respect to the design variables.

The two kinds of techniques do not have to be used in isolation. A designer could, for example, use a gradient-free method to obtain a good initial design that is then improved by a gradient-based approach.

Many methods have been devised in order to obtain the derivative of the objective functions and constraints with respect to design variables. The most basic approach is finite differences. This technique consists in modifying the value of the design variables by a small amount and solving the problem again. The difference in value of the objective function between the original and modified designs is then used in order to estimate the gradient. Therefore, it requires that the objective function is evaluated at least as many times as there are design variables. This is very computationally expensive when the number of design variables is large, as is the case in the early design stages.

For the *adjoint method*, the gradient calculation is rewritten such that it is computed only once per objective function or constraint. The technique can be applied to either the continuous or the discrete equations that model the problem. In the former case, the adjoint equations are then discretised. Independently of the specific approach, the adjoint method is computationally advantageous in problems with many design variables and few objective functions.

As previously explained, higher-fidelity FSI simulations are often computed using a partitioned approach. Using two different solvers does not make calculating gradients using finite differences significantly more complex. However, it complicates the calculation of the adjoint solution. In order to take into account the coupling of the two solvers, some data must be transferred from the solid to the fluid and vice-versa.

In the adjoint case, it is the gradients with respect to boundary variables, which have to be interpolated.

Once the coupled adjoint system is defined, it can be used to solve optimisation problems. One important avenue of research is aerodynamic shape optimisation. It consists in modifying the shape of an aerodynamic body in order to maximise or minimise an objective function. The surface deformation can be expressed as a function of shape design variables. By using the adjoint method, many design variables can be introduced without increasing the computational cost of the gradients. Therefore, the design space available can be extended.

Shape design variables can be classified into two main families: constructive and deformative [5]. The former fully define the surface while the latter modify an already-existing shape. For both kinds, the gradients of the objective functions with respect to the change they induce on the boundary must be calculated. If using a partitioned code, the effect on the structure and the fluid has to be combined in order to obtain the gradient.

1.3 Objective of the thesis and key questions

The objective of the present thesis is to accelerate high-fidelity, unsteady, nonlinear FSI optimisation in order to make it available at earlier stages of the design process. In order to do so, the harmonic balance and adjoint methods are combined. The harmonic balance technique results in faster unsteady simulations for time-periodic problems, while the adjoint method reduces the computational cost of gradient calculation. During the definition of the steps to be taken, some key questions arise.

First, the harmonic balance method has to be implemented using a partitioned FSI approach. What is needed to apply the harmonic balance to problems with *a priori* unknown frequency in partitioned solvers? Since the main reason behind using the harmonic balance approach is computational efficiency, how much faster is it compared to time marching simulations? How accurate is the harmonic balance solution?

Second, the adjoint method needs to be introduced for partitioned solvers. For simplicity, it can be applied to steady problems first. Its implementation has to be verified. How does the partitioned FSI adjoint approach work? How do the gradients obtained using the adjoint method compare to those obtained using finite differences? Is there a difference between fluid and solid design parameters?

The adjoint method can be applied to the harmonic balance FSI technique developed earlier. Then, shape design variables can be introduced. How does the partitioned HB FSI adjoint approach work? Are the gradients of structural, fluid

and shape parameters comparable to those obtained by finite differences? Is the adjoint approach faster or slower than the direct harmonic balance method? Are there any other advantages to the harmonic balance approach for optimisation?

Once verified, the adjoint harmonic balance approach can be applied to FSI optimisation problems. Are the adjoint-based gradients obtained suitable to optimise limit-cycle oscillations and other nonlinear aeroelastic phenomena? What kinds of objective functions can be maximised or minimised using the present approach? How does the shape of an aerofoil change in order to avoid limit-cycle oscillations? Can the method be combined with other objective functions or constraints?

1.4 Main contributions

In the present work, a new harmonic balance adjoint method for FSI problems is introduced. Three main parts can be identified: development of the technique, implementation in code and verification of results.

For the first part, a partitioned harmonic balance approach with a novel frequency iteration method has been developed. Partitioned FSI harmonic balance solvers use the L^2 residual norm method [6], which introduces one extra equation, to iterate on the frequency. Here, a phase-fixing technique is proposed instead. This approach removes one degree of freedom from the structural problem, keeping the same number of equations. The adjoint method has been applied to the harmonic balance approach developed, including defining the coupling between the fluid and structural solvers. While, recently, there was an adjoint method developed for a harmonic balance aerostructural solver, it did not include deformation of the fluid mesh [7]. The introduction of a deformable mesh is required for the calculation of more complex cases with flexible structural modes. Furthermore, Prasad et al. used a different frequency iteration technique and a different approach to the calculation of limit-cycle oscillations [7].

Regarding the implementation of the method, there are three main code components of the partitioned FSI approach: the fluid and mesh solver, the solid solver and the coupler. All three have been modified for this project to accommodate the harmonic balance adjoint method.

The fluid and mesh code used is the SU2 CFD suite [8]. It includes a time-domain harmonic balance fluid solver [9]. For this work, SU2 has been extended to allow the coupler to programmatically modify the base frequency of the fluid solver. Furthermore, it has been adapted to introduce mesh deformation to the pre-existing harmonic balance solver. An adjoint harmonic balance solver has also been implemented.

A frequency-domain pitch-plunge solver with the new frequency iteration method has been developed by modifying a time-domain solver. Then, the corresponding adjoint equations have also been added. Two structural objective functions have been introduced in the code: the pitching amplitude squared and power dissipation on the plunge damper. In order to enable shape optimisation of the problem, a family of deformative shape variables, Hicks-Henne bumps, has been implemented in the structural solver. The gradients with respect to structural and shape design parameters have also been coded.

The coupler has been extended to work with the time-domain harmonic balance method and frequency iteration scheme. The interpolation and coupling have been modified to take into account the presence of several time instances and to transfer the frequency from the solid solver to the fluid and mesh solver. Furthermore, several interfaces have been written and extended. These interfaces allow transferring data from the individual solvers to the coupler and vice-versa. In particular, adjoint interfaces for SU2 (steady and harmonic balance), the SU2 solid solver, a beam code and the pitch-plunge solver have been implemented.

The aforementioned methods and their implementation have been verified using a series of test cases. The direct HB approach has been compared to time-marching results for a 2D limit-cycle oscillation test case under transonic flow conditions. The gradients obtained using the adjoint methods, both steady and unsteady, have been verified by comparing them to finite differences. For the steady adjoint a 2D beam in crossflow was used, while for the unsteady approach one LCO setup with high pitching amplitude was selected.

Finally, the adjoint harmonic balance method developed has been applied to four limit-cycle oscillation optimisation cases. The cases take advantage of the setup used for the verification of the direct and adjoint approaches.

1.5 Thesis overview

The present document is divided in six chapters. Fluid-structure interaction optimisation is introduced in the first chapter.

In the second chapter, the application of the harmonic balance approach to coupled, partitioned FSI calculations is detailed. This technique is extended to problems with *a priori* unknown base frequency. Then, the FSI calculations are verified for a known-frequency case and an unknown-frequency setup.

The steady adjoint method for partitioned FSI calculations is described in the third chapter. This approach is then applied to a test case of a beam in low-Reynolds crossflow in order to verify the implementation of the coupling.

In the fourth chapter, the adjoint and harmonic balance methods are combined in order to efficiently obtain gradients of FSI time-periodic calculations. The unknown-frequency test case in the second chapter is used in order to verify the gradients. Two different objective functions are used: the motion amplitude and the mean drag coefficient.

The combined adjoint harmonic balance technique presented in Chapter 4 is used to optimise a series of cases based on the unknown-frequency case in the fifth chapter. These include amplitude minimisation and energy dissipation maximisation.

Finally, in the sixth chapter, the main conclusions are presented. Furthermore, some extensions and possible uses of the proposed approach are discussed.

Chapter 2

Frequency-domain techniques for fluid-structure interaction

As described in the previous chapter, there are many interesting phenomena that involve fluid-structure interaction (FSI). While some of them are steady, many others are unsteady. In order to study unsteady, but ultimately time-periodic problems, either time-domain or frequency-domain techniques can be used. While some of these time-periodic problems may have a known fundamental frequency, for many others the frequency is not known.

The present chapter starts by introducing FSI computational techniques, both steady and unsteady. Afterwards, the linear frequency-domain approach and then the harmonic balance methods are presented. Once the frequency- and time-domain harmonic balance are presented, frequency iteration techniques for the coupled problem are shown. Finally, the results of two aeroelastic test cases are described. They are used to verify the coupled FSI harmonic balance algorithm.

2.1 Numerical study of fluid-structure interaction

Often, fluid-structure interaction phenomena are simulated numerically in order to further understand their behaviour. This modelling is crucial to prevent structural failure from occurring, either in testing or in operation, and to ensure an effective and efficient design.

In order to carry out fluid-structure interaction simulations, an engineer is presented with various choices. Two such choices are discussed in the present section: the level of fidelity of the fluid modelling and the coupling between the fluid and the structure.

2.1.1 Different levels of fidelity

Fidelity and computational cost are tightly coupled. There is a trade-off: higher-fidelity methods require higher computational costs. Usually, fluid modelling is more computationally expensive than structural modelling.

Early in the design process, lower-fidelity approaches are used. They provide faster results, which is crucial in this stage, during which thousands of calculations are required [10]. One example of a lower-fidelity method in aerodynamics is the incompressible linear potential equation,

$$\nabla\phi = \mathbf{v} \tag{2.1}$$

$$\nabla^2\phi = 0, \tag{2.2}$$

where $\mathbf{v}$ is the flow velocity vector and the corresponding velocity potential is ϕ. However, this results in an irrotational flow and a wake, which represents a thin discontinuity, is usually added to enable the prediction of lift. The linear potential equation can be solved either by lifting-line, panel or field methods.

One of the most widely used linear potential techniques for aeroelastic prediction is the doublet lattice method (DLM) [11]. It is a linear potential solver that discretises the wing's planform into flat panels that feature elementary solutions of the unsteady subsonic potential equation and imposes the impermeability boundary condition on the panel nodes [12, 13]. This is called the zero-normal velocity boundary condition. Other linear potential approaches include the vortex lattice method (VLM) [14, Ch. 13] and the source-and-doublet panel method (SDPM) [15, 16].

The VLM is similar to the DLM but makes use of different elementary solutions to the potential equation. Unlike the DLM, the VLM requires wake modelling; it can be applied with a fixed or a free wake. Free wake modelling is of great importance in cases where the wake interacts with aerodynamic surfaces, such as in wind turbines [17]. The VLM has been used to predict aircraft aeroelastic responses [18], bird wing flapping [19] or energy-harvesting kites [20].

Unlike the DLM or the VLM, the SDPM can represent the thickness of the wing. It can use both a fixed and a free wake. Some changes to the SDPM, a nonlinear calculation of the pressure coefficient and a zero-normal mass flow boundary condition, lead to a better match with experimental data in the sub-critical subsonic regime [21, 22].

These linear methods work well in subsonic conditions with attached flows. Applying the Prandtl-Glauert geometric compressibility correction extends the range of validity of steady solvers from the incompressible regime ($\mathrm{M}_\infty = 0$) to the compressible regime. However, most commercial aircraft fly at high subsonic freestream

conditions, which leads to the presence of regions with a locally-supersonic flow. This is the transonic regime. In this regime, the flow becomes highly nonlinear, which leads to a loss of prediction power for linear methods.

Furthermore, the aeroelastic behaviour of wings changes. In general, flutter margins are significantly reduced in the transonic regime. This is called the transonic dip. Early on, it was found that supercritical wings with similar structural characteristics, same planform and same maximum thickness as conventional wings had stricter flutter limits in the transonic regime [23].

There are two main ways of extending the linear potential equation to handle transonic flows: correcting the steady aerodynamic loads predicted by linear methods using higher-fidelity data and using the full potential equation. The two main approaches for correcting loads are correcting the generalised aerodynamic loads matrix and correcting the aerodynamic influence coefficient matrix.

The full potential equation is

$$\dot{\rho} + \nabla \cdot (\rho \nabla \phi) = 0, \tag{2.3}$$

where ρ is the local density. As in the linear potential case, the velocity is given by

$$\nabla \phi = \mathbf{v}. \tag{2.4}$$

The linear potential equation shown in Eq. (2.2) can be trivially obtained from Eq. (2.3) by assuming that ρ is constant. Otherwise, a relation between the potential and the density has to be found in order to solve the full potential equation. Assuming that the flow is isentropic, that the gas is perfect and that heat conduction is negligible, the density can be written as a function of the potential by means of a thermodynamic relation [24]. The ideal gas relations for an adiabatic flow are

$$\rho = \frac{p}{RT} \tag{2.5}$$

$$\frac{p}{p_\infty} = \left(\frac{\rho}{\rho_\infty}\right)^\gamma, \tag{2.6}$$

where p is the local pressure, T the local temperature, p_∞ and ρ_∞ are the freestream pressure and density, $R = C_p - C_v$ is the specific gas constant and $\gamma = \frac{C_p}{C_v}$ is the specific heat capacity ratio. Then, the local speed of sound, a, is given by

$$a = \sqrt{\gamma RT} = \sqrt{\gamma \frac{p}{\rho}}. \tag{2.7}$$

The inviscid momentum equation is

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \nabla h = 0, \tag{2.8}$$

where $h = C_p \cdot T$ is the enthalpy. Since the flow is irrotational, the second term in Eq. (2.8) becomes

$$\mathbf{v} \cdot \nabla \mathbf{v} = \nabla \left(\frac{v^2}{2} \right), \tag{2.9}$$

where v is the magnitude of the velocity. Substituting Eq. (2.9) and Eq. (2.4) into Eq. (2.8) and operating leads to

$$\nabla \left(\frac{\partial \phi}{\partial t} + h + \frac{v^2}{2} \right) = 0. \tag{2.10}$$

For an ideal gas, the enthalpy can be expressed as a function of the pressure and density

$$h = C_p \cdot T = C_p \cdot \frac{p}{R \cdot \rho} = \frac{\gamma}{\gamma - 1} \cdot \frac{p}{\rho}. \tag{2.11}$$

Substituting the enthalpy in Eq. (2.10) by this expression and integrating over an arbitrary line, the unsteady Bernoulli equation can be written as

$$\frac{\partial \phi}{\partial t} + \frac{\gamma}{\gamma - 1} \cdot \frac{p}{\rho} + \frac{v^2}{2} = C(t), \tag{2.12}$$

where $C(t)$ is a function of time [25].

Equating the value of Eq. (2.12) at freestream and local conditions and operating results in

$$\frac{p}{\rho} \cdot \frac{\rho_\infty}{p_\infty} = 1 + (\gamma - 1) \frac{p_\infty}{\gamma \cdot \rho_\infty} \cdot \left(-\frac{\partial \phi}{\partial t} - \frac{v^2}{2} + \frac{v_\infty^2}{2} \right). \tag{2.13}$$

Substituting Eq. (2.6) on the left-hand side and Eq. (2.7) on the right-hand side, the expression of the ratio of local to freestream density is

$$\frac{\rho}{\rho_\infty} = \left[1 + \frac{\gamma - 1}{2} \left[\mathrm{M}_\infty^2 - \frac{1}{a_\infty^2} \left(2\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y} + \frac{\partial \phi}{\partial z} \right) \right] \right]^{\frac{1}{\gamma - 1}}. \tag{2.14}$$

This relation results in a dependence between the gradients of the potential, ϕ, and the local density, ρ, that can be used to close Eq. (2.3) [25].

The isentropic assumption is violated in strong shocks, that result in a marked increase in entropy. For transonic solvers, the shock is considered weak if the upstream Mach number is $\mathrm{M} < 1.3$ [25]. Like in the linear potential case, the full potential equation assumes that the flow is irrotational, leading to the necessity of adding special wake treatment. Full potential solvers, such as TRANAIR [26], have been used in order to study aerodynamic behaviour. The full potential method has also been used recently in order to solve steady aeroelastic problems [27] and their optimisation [28].

The next level of fidelity are the compressible Euler equations. These still model an inviscid flow. They are

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}^c = 0 \tag{2.15}$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho \mathbf{v} \\ \rho E \end{bmatrix}, \qquad \mathbf{F}^c = \begin{bmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \otimes \mathbf{v} + p\mathbf{I} \\ \rho E \mathbf{v} + p\mathbf{v} \end{bmatrix}, \tag{2.16}$$

where $\mathbf{U}$ is the vector of conservative variables, $\mathbf{F}^c$ are the convective fluxes, ρ is the density, $\mathbf{v}$ is the velocity vector, E is the total specific energy, p is the pressure and $\mathbf{I}$ is the identity matrix. Unlike the potential equations, whether linear or full, the Euler equations do not require the wake to be treated in a particular way because they are not irrotational. Furthermore, they can model strong shocks in the transonic regime because they are not isentropic.

Viscous effects are important for drag prediction. Furthermore, in transonic flows there is an interaction between the shock and the boundary layer. Usually, the shock is softened and appears upstream of the inviscid prediction. This can affect the lift and the aeroelastic behaviour of the body being studied. Similarly to the difference between the subsonic and the transonic regime, these effects can be taken into account in two ways: the inviscid solution can be corrected by means of a boundary layer model or the Navier-Stokes equations can be solved.

The Navier-Stokes equations include directly the effect of viscosity. They add to Eq. (2.15) a diffusive flux vector $\mathbf{F}^d$,

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}^c - \nabla \cdot \mathbf{F}^d = 0 \tag{2.17}$$

$$\mathbf{F}^d = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \\ \boldsymbol{\tau}\mathbf{v} \end{bmatrix}, \qquad \boldsymbol{\tau} = \mu \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T - \frac{2}{3}\mathbf{I} \nabla \cdot \mathbf{v} \right), \tag{2.18}$$

where $\boldsymbol{\tau}$ is the stress tensor and μ is the dynamic viscosity. The Navier-Stokes equations are difficult to solve at high Reynolds numbers, such as the ones found in aircraft. At those Reynolds numbers, the flow is intrinsically unsteady, with a wide range of frequencies and scales of turbulence. In order to fully resolve the turbulent behaviour of the flow, very fine meshes and, thus, very small time-steps are required. This is called direct numerical simulation (DNS). Assuming that the number of operations is proportional to the number of nodes of the mesh and the number of time steps required, the computational cost scales with Re^3.

One way to model high-Reynolds flows is to separate the turbulent oscillations from the mean flow by using a Reynolds average or, for compressible flows, a Favre

average. This process results in what is usually called the Reynolds-averaged Navier-Stokes equations (RANS). The equations are not closed, so a turbulence model has to be introduced to calculate the Reynolds' stresses. There are many turbulence models of varying complexity and applicability, which introduces uncertainty on the solutions obtained. The RANS equations still require a fine grid, especially in the boundary layer, but coarser than the equivalent DNS grid.

Since in many aeroelastic problems viscous effects are concentrated in the boundary layer, some methods that model the behaviour of this region separately have been proposed. These consist in solving the Navier-Stokes equations inside the boundary layer, with some simplifications. Then, the boundary layer solution can be coupled with an inviscid solver, that uses Euler [29, 30] or potential flow equations [31] to provide more accurate results at a reduced computational cost.

Currently, there is some interest in using models that resolve only the larger scales of the turbulence, such as Large Eddy Simulation (LES) [32] or mixed RANS-LES models [33]. These simulations are computationally expensive and, like DNS, unfeasible for complete aircraft geometries.

2.1.2 Steady coupling of fluid and solid domains

Independently of the fluid and solid models used, FSI calculations can also be classified depending on the number of solvers used. A monolithic solver is one that treats both the fluid and solid physics as part of the same problem. A partitioned approach uses different solvers for the fluid and solid problems. Generally, this second approach is more flexible, since it allows switching either solver by a different one at any given point.

In the case of a partitioned code, two fields can be considered: the structure ($\mathcal{S}$) and the fluid ($\mathcal{F}$). However, since the structural displacements modify the boundary of the fluid problem, the mesh also needs to be considered. Since creating a new mesh is computationally expensive, often the mesh is deformed to adapt to the new boundary. Then, the mesh can be added as a third field ($\mathcal{M}$) to be solved depending on the boundary displacements [34, 35]. Adding the respective dependence between the variables [36, 37]

$$\begin{cases} \mathcal{S}\left(\mathbf{u}, \mathbf{w}, \mathbf{z}\right) & = 0 \\ \mathcal{F}\left(\mathbf{w}, \mathbf{z}\right) & = 0 \\ \mathcal{M}\left(\mathbf{u}, \mathbf{z}\right) & = 0 \end{cases}, \tag{2.19}$$

where $\mathcal{S}$ represents the solid solver, $\mathcal{F}$ represents the fluid solver and $\mathcal{M}$ the mesh solver; and $\mathbf{u}$ are the structural displacements, $\mathbf{w}$ are the fluid conservative variables and $\mathbf{z}$ the mesh node displacements. Note that this formulation is independent of the equations being solved.

Equation (2.19) can be linearised with respect to each of the solver's variables obtaining the expression

$$\begin{bmatrix} \mathcal{S} \\ \mathcal{F} \\ \mathcal{M} \end{bmatrix} + \begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \mathbf{u}} & \frac{\partial \mathcal{S}}{\partial \mathbf{w}} & \frac{\partial \mathcal{S}}{\partial \mathbf{z}} \\ 0 & \frac{\partial \mathcal{F}}{\partial \mathbf{w}} & \frac{\partial \mathcal{F}}{\partial \mathbf{z}} \\ \frac{\partial \mathcal{M}}{\partial \mathbf{u}} & 0 & \frac{\partial \mathcal{M}}{\partial \mathbf{z}} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \mathbf{w} \\ \Delta \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{2.20}$$

The matrix containing the partial derivatives has a set of three diagonal blocks: $\partial \mathcal{S}/\partial \mathbf{u}$, $\partial \mathcal{F}/\partial \mathbf{w}$, $\partial \mathcal{M}/\partial \mathbf{z}$. These terms depend on each solver's own variables. Furthermore, there are some cross-terms, outside of the diagonal. These cross-terms represent the coupling between the solvers.

While, in theory, the full system could be solved at every iteration using a Newton-Raphson procedure, in practice the system is usually too large and complex. It would require calculating and accessing the derivatives of each of the codes and then solving a larger system than any of the individual solvers. In practice, instead of solving the full problem, the coupling can be applied by means of boundary conditions. So, the solid problem receives the loads from the fluid solver as a Neumann boundary condition. The solid solver calculates the structural displacements and their value at the boundary is applied as a Dirichlet boundary condition of the fluid and mesh solvers. This results in a staggered problem, with the solid and fluid solvers alternating. This is a Block-Gauss-Seidel (BGS) scheme.

While many problems converge quickly applying a BGS scheme, others do not. If the effect of the structural solver on the problem is too large, the solution may diverge. One of the reasons why this can occur is added-mass effects. They are especially important when the ratio between the fluid density and the solid density is high enough.

One way to prevent divergence of the FSI solver is to apply underrelaxation. In this technique, a parameter $0 < \Omega \leq 1$ is defined so that the new solution is only partially updated. For example, if the underrelaxation is applied to the structural displacements at FSI iteration n, the expression is

$$\mathbf{u}_{\mathcal{F},n} = \mathbf{u}_{\mathcal{F},n-1} + \Omega \cdot (\mathbf{u}_{\mathcal{S},n-1} - \mathbf{u}_{\mathcal{F},n-1}), \tag{2.21}$$

where $\mathbf{u}_{\mathcal{F},i}$ are the fluid boundary displacements at the ith FSI iteration and $\mathbf{u}_{\mathcal{S},i}$ are the displacements calculated by the solid solver at the end of the same iteration. In this case for every iteration the solid equations are solved after the fluid equations. While the simplest case is one in which Ω is constant, some schemes have

been developed to update its value in order to improve convergence. Usually, the relaxation parameter is updated based on the value of the residuals.

One such method is based on Aitken's Δ^2 method. It updates the relaxation parameter for FSI iteration $n+1$, Ω^{n+1}, on the basis of the previous value, Ω^n, and the residuals [38, 39].

Another way of avoiding divergence due to added-mass effects is by using other coupling methods, such as quasi-Newton approaches. In these, the full Jacobian is not stored, so the memory requirements are lower compared to the full Newton-Raphson iteration. An example is the interface quasi-Newton inverse least squares (IQN-ILS) approach. In this method the inverse of the interface Jacobian is estimated from the changes in residuals at the boundary [40].

2.1.3 Interpolation of loads and displacements

The previous section has presented some methods for coupling fluid and structural solvers. If the position of the boundary nodes matches, the transfer of forces and displacements from one solver to the other is simple. However, sometimes the boundary nodes differ. Often, the structural mesh is simpler than the fluid mesh. Hence, some approaches have been developed in order to interpolate loads and displacements from one mesh to another.

There are two main kinds of interpolation techniques: conservative and consistent. Conservative interpolation methods conserve the total loads, the energy transferred at the boundary and rigid-body translations of the boundary. Applying the principle of equivalence of virtual work leads to [41]

$$\delta W = \delta \mathbf{u}_f^T \mathbf{f}_f = \delta \mathbf{u}_s^T \mathbf{f}_s, \tag{2.22}$$

where δW is the virtual work, $\delta \mathbf{u}$ is the vector of virtual displacements at the nodes, $\mathbf{f}$ is the vector of forces at the nodes and the subindices f and s represent the fluid and the solid boundaries, respectively. Generally, the interpolation is linear. Then, the solid-to-fluid interpolation matrix $\mathbf{H}$ transforms the solid boundary displacements into fluid boundary displacements

$$\mathbf{u}_f = \mathbf{H}\mathbf{u}_s \tag{2.23}$$

and the fluid-to-solid interpolation matrix $\mathbf{G}$ transforms the fluid boundary loads into solid boundary loads

$$\mathbf{f}_s = \mathbf{G}\mathbf{f}_f. \tag{2.24}$$

The virtual displacements are then

$$\delta \mathbf{u}_f = \mathbf{H}\delta \mathbf{u}_s \tag{2.25}$$

and in order to conserve the virtual work at the interface, per Eq. (2.22)

$$\mathbf{f}_s = \mathbf{H}^T \mathbf{f}_f. \tag{2.26}$$

Therefore, comparing Eqs. (2.24) and (2.26), for a conservative interpolation method $\mathbf{G} = \mathbf{H}^T$.

Instead of conserving the energy transferred, consistent methods conserve a constant pressure distribution. This can be important for some highly-flexible problems in which a conservative method may lead to unphysical oscillations [42]. In order to do so, the row-sum of the pressure-transfer matrix and matrix $\mathbf{H}$ must be equal to 1 [42]. For other multiphysics problems, such as combined heat transfer, a consistent approach that conserves the heat flux is desirable [37].

Determination of interpolation matrices

There are multiple methods to obtain the interpolation matrices $\mathbf{H}$ and $\mathbf{G}$. The simplest approach is a nearest-neighbour interpolation. In this method, each node at a fluid boundary receives the displacement from the nearest solid boundary node. Rigid body translations are conserved. However, if the displacement at the solid boundary is not constant, its interpolation at the fluid boundary is not continuous. This leads to a staircase shape of the fluid boundary.

Another method is the use of radial-basis functions. The interpolating function is given by

$$w(\mathbf{x}) = p(\mathbf{x}) + \sum_{k=1}^{n_s} \alpha_k \cdot \phi\left(\|\mathbf{x} - \mathbf{x}_k\|\right), \tag{2.27}$$

where $\mathbf{x}$ is the coordinate of the node, p is a low-order polynomial, α_k are the coefficients multiplying the basis function ϕ and n_s is the number of nodes at the solid interface. The interpolating function has to recover the original values of the displacements at each solid boundary node

$$w(\mathbf{x}_k) = u_k, \qquad 1 \leq k \leq n_s. \tag{2.28}$$

In many cases, the order of the polynomial p is chosen to be 1

$$p(x, y, z) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot y + \beta_3 \cdot z. \tag{2.29}$$

Furthermore, the coefficients α_k in Eq. (2.27) have to follow

$$\sum_{k=1}^{n_s} \alpha_k \cdot q(\mathbf{x}) = 0 \tag{2.30}$$

for all polynomials q such that $\deg(q) \leq \deg(p)$. Equations (2.29) and (2.30) together imply that if the values of u_k come from a linear polynomial, the interpolating function is this exact linear polynomial. Therefore, rigid-body translations and rotations are recovered exactly. Furthermore, if function ϕ is positive-definite, the interpolant is unique [41, 43].

Two interpolation matrices can be introduced: one that recovers the displacement of the solid boundary nodes from the interpolation coefficients, $\mathbf{C}_{ss}$ of size $(n_s + 4) \times (n_s + 4)$:

$$\mathbf{C}_{ss} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_s} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{u}_s \end{bmatrix} \tag{2.31}$$

and one that interpolates the displacements to the fluid boundary nodes, $\mathbf{A}_{fs}$ of size $(n_f + 4) \times (n_s + 4)$:

$$\mathbf{A}_{fs} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_s} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{u}_f \end{bmatrix}. \tag{2.32}$$

Assume that $\mathbf{C}_{ss}$ is invertible. Premultiply both sides of Eq. (2.31) by $\mathbf{C}_{ss}^{-1}$. The coefficients of the interpolant in Eq. (2.27) are equal to the inverse of matrix $\mathbf{C}_{ss}$ times the displacement vector. These coefficients can be substituted on the left-hand side of Eq. (2.32). The expression for the interpolation matrix $\mathbf{H}$ is then

$$\mathbf{H} = \mathbf{A}_{fs}\mathbf{C}_{ss}^{-1}. \tag{2.33}$$

The elements of $\mathbf{C}_{ss}$ are

$$\mathbf{C}_{ss} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & 0 & x_{s_1} & x_{s_2} & \cdots & x_{s_{n_s}} \\ 0 & 0 & 0 & 0 & y_{s_1} & y_{s_2} & \cdots & y_{s_{n_s}} \\ 0 & 0 & 0 & 0 & z_{s_1} & z_{s_2} & \cdots & z_{s_{n_s}} \\ 1 & x_{s_1} & y_{s_1} & z_{s_1} & \phi_{s_1,s_1} & \phi_{s_1,s_2} & \cdots & \phi_{s_1,s_{n_s}} \\ 1 & x_{s_2} & y_{s_2} & z_{s_2} & \phi_{s_2,s_1} & \phi_{s_2,s_2} & \cdots & \phi_{s_2,s_{n_s}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{s_{n_s}} & y_{s_{n_s}} & z_{s_{n_s}} & \phi_{s_{n_s},s_1} & \phi_{s_{n_s},s_2} & \cdots & \phi_{s_{n_s},s_{n_s}} \end{bmatrix}, \tag{2.34}$$

where x_{s_i}, y_{s_i}, z_{s_i} are the coordinates of the ith node on the solid boundary and $\phi_{s_i,s_j} = \phi\left(\left\|\mathbf{x}_{s_i} - \mathbf{x}_{s_j}\right\|\right)$. Since $\phi_{s_i,s_j} = \phi_{s_j,s_i}$ matrix $\mathbf{C}_{ss}$ is symmetric. From its structure, it is possible to see that if all the points in the solid boundary are coplanar, the matrix is not invertible, independently of the basis function used. In particular, the first four rows and columns are linear combinations of each other if and only if the points are coplanar. The same issue appears for a two-dimensional case with all the solid boundary points lying on a straight line. If matrix $\mathbf{C}_{ss}$ is not invertible, per Eq. (2.33) the interpolation matrix $\mathbf{H}$ cannot be calculated. The fluid interpolation matrix is given by

$$\mathbf{A}_{fs} = \begin{bmatrix} 1 & x_{f_1} & y_{f_1} & z_{f_1} & \phi_{f_1,s_1} & \phi_{f_1,s_2} & \cdots & \phi_{f_1,s_{n_s}} \\ 1 & x_{f_2} & y_{f_2} & z_{f_2} & \phi_{f_2,s_1} & \phi_{f_2,s_2} & \cdots & \phi_{f_1,s_{n_s}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{f_{n_f}} & y_{f_{n_f}} & z_{f_{n_f}} & \phi_{f_{n_f},s_1} & \phi_{f_{n_f},s_2} & \cdots & \phi_{f_{n_f},s_{n_s}} \end{bmatrix}, \tag{2.35}$$

where x_{f_i}, y_{f_i}, z_{f_i} are the coordinates of the ith node on the fluid boundary, n_f is the number of nodes on the fluid boundary and $\phi_{f_i,s_j} = \phi\left(\left\|\mathbf{x}_{f_i} - \mathbf{x}_{s_j}\right\|\right)$. For a conservative interpolation approach, the fluid-to-solid interpolation is performed by transposing matrix $\mathbf{H}$. In the case of a consistent interpolation technique, this procedure is repeated for the fluid-to-solid transfer.

Function ϕ can have either global or local support. The value of globally-supported interpolating functions depends on the distance from each node to every node. One example of a globally-supported function is the thin-plate spline method [44], which is given by

$$\phi\left(\|\mathbf{x} - \mathbf{x}_k\|\right) = \|\mathbf{x} - \mathbf{x}_k\|^2 \cdot \ln\left(\|\mathbf{x} - \mathbf{x}_k\|\right). \tag{2.36}$$

Because of the global support, matrix $\mathbf{C}_{ss}$ in Eq. (2.33) is full. This means that an inversion of a full $(n_s + 4) \times (n_s + 4)$ matrix is required in order to obtain the coefficients of the interpolating function [42, 45].

Local support eliminates the impact of nodes that are farther than some radius r, known as the support radius. This elimination introduces zero-elements in matrix $\mathbf{C}_{ss}$. If enough elements are eliminated, matrix $\mathbf{C}_{ss}$ is then sparse, which helps with performance in very large problems. One example of local support is the family of smooth, compactly-supported functions introduced by Wendland. The simplest $\mathcal{C}^2$ function in $\mathbb{R}^3$ is given by [43]

$$\phi\left(\left\|\mathbf{x}-\mathbf{x}_k\right\|,\rho\right)=\left(1-\frac{\left\|\mathbf{x}-\mathbf{x}_k\right\|}{\rho}\right)_+^4\cdot\left(4\cdot\frac{\left\|\mathbf{x}-\mathbf{x}_k\right\|}{\rho}+1\right), \tag{2.37}$$

where $\rho = r$ is the support radius. The first term, a truncated power function, is strictly positive if $\|\mathbf{x}-\mathbf{x}_k\| < \rho$ and 0 otherwise [41, 43].

2.2 Unsteady fluid-structure interaction

Section 2.1.2 has presented some methods to simulate steady fluid-structure interaction. However, many problems of engineering interest are unsteady. A generic, time-marching technique for unsteady, partitioned FSI is shown in the present section.

The equation solved is of the form

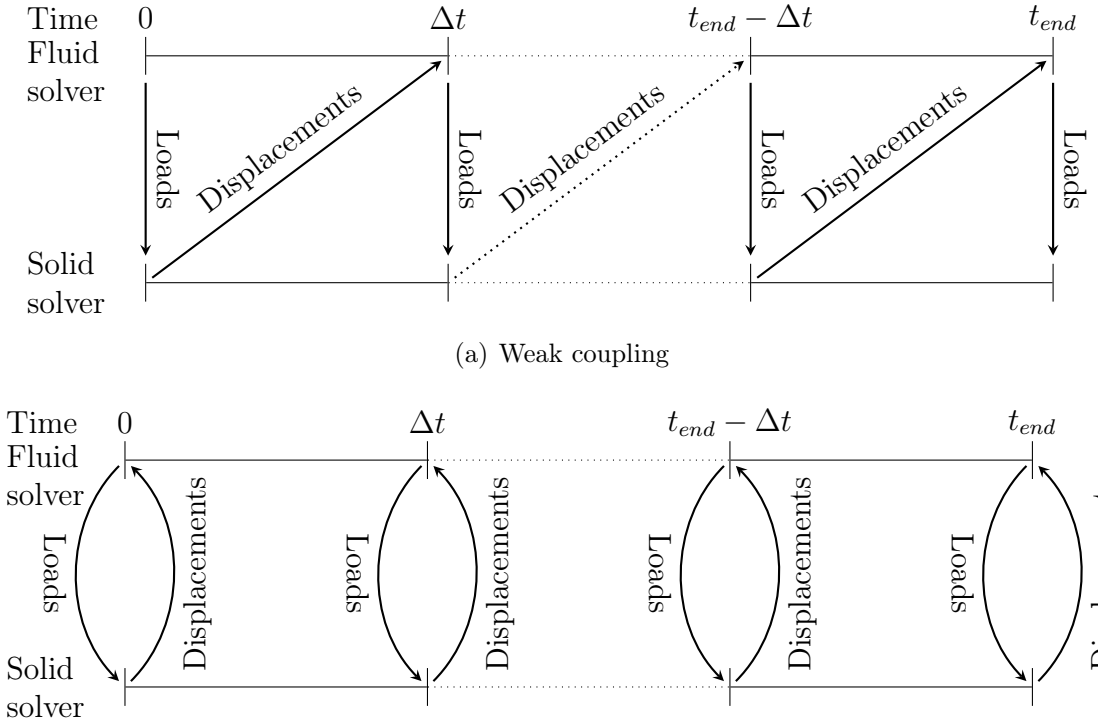
$$\frac{\partial}{\partial t}\mathbf{U}(t)-\mathbf{R}\left(\mathbf{U}(t),t\right)=0, \tag{2.38}$$

where $\mathbf{U}(t)$ is the vector of conservative variables at time t and $\mathbf{R}\left(\mathbf{U}(t),t\right)$ is the corresponding residual. For simplicity, the time step used in the fluid and structural solvers is assumed to be the same. However, some methods use larger time steps for the structural solver.

Unsteady FSI calculations can be classified depending on whether the coupling is strong or weak. In weak coupling, at each time iteration the fluid solver runs first. Then, the loads are transferred to the solid solver. This solver then calculates the corresponding displacements, which are given as boundary conditions to the fluid solver at the following time iteration. Figure 2.1(a) shows schematically this kind of coupling.

While it is simple to implement, weak coupling generally requires smaller time steps. This leads to a larger number of time steps for the same total time and, thus, a higher computational cost.

In strong coupling, however, there is a back and forth exchange of data between the fluid and the solid. This process is repeated at each time step until a certain



(a) Weak coupling

(b) Strong coupling

Figure 2.1: Coupling of fluid and structural solvers with equal time step

level of FSI convergence is reached. Thus, the boundary displacements are obtained at the same time iteration as the aerodynamic loads, instead of at the previous time iteration. The strong coupling process is shown in Fig. 2.1(b).

2.3 Linear frequency-domain method

Time-marching simulations can represent the behaviour of coupled, unsteady fluid-structure systems. However, they are computationally expensive. They have to simulate the transient part of the system's response, which may require many cycles. While in some cases a lower-fidelity method can be used, in others higher-fidelity approaches are needed to provide more accurate results. In order to reduce the computational cost of these solutions while keeping higher fidelity, some methods that work on the frequency domain have been developed. One such technique is the linear frequency-domain method (LFD). It consists in linearising the solution in time with respect to frequency [46]. This procedure was used by Clark and Hall

for the study of stall flutter in turbomachinery [47, 48] and has also been used to predict loads around aerofoils, wings and aircraft [49–51].

The flow is decomposed into a mean flow and a small, time-dependent perturbation that captures the unsteady behaviour of the system:

$$\mathbf{U}(\mathbf{x},t) \simeq \hat{\mathbf{U}}_0 + \tilde{\mathbf{U}}(t) \qquad \left|\tilde{\mathbf{U}}\right| \ll \left|\hat{\mathbf{U}}_0\right| \tag{2.39}$$

$$\mathbf{x}(t) \simeq \hat{\mathbf{x}}_0 + \tilde{\mathbf{x}}(t) \qquad |\tilde{\mathbf{x}}| \ll |\hat{\mathbf{x}}_0|\,, \tag{2.40}$$

where $\hat{\mathbf{U}}_0$ is the mean flow, $\tilde{\mathbf{U}}$ is the unsteady component of the flow, $\hat{\mathbf{x}}_0$ is the mean displacement and $\tilde{\mathbf{x}}$ the unsteady displacement. Introduce the dependence on the displacements of Eq. (2.38)

$$\frac{\partial}{\partial t}\mathbf{U}(t) - \mathbf{R}\left(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{U}(t), t\right) = 0. \tag{2.41}$$

Assume that the mean flow follows the steady equations

$$\mathbf{R}\left(\hat{\mathbf{x}}_0, \mathbf{0}, \hat{\mathbf{U}}_0\right) = 0. \tag{2.42}$$

Linearise Eq. (2.41) around the steady solution. Then, substitute Eqs. (2.39) and (2.40) into the linearised equation

$$\frac{\partial \tilde{\mathbf{U}}}{\partial t} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\tilde{\mathbf{U}} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\tilde{\mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{x}}}\dot{\tilde{\mathbf{x}}}. \tag{2.43}$$

The unsteady perturbation and its time derivatives can be written as a one-term Fourier series:

$$\tilde{\mathbf{U}}(t) \simeq \hat{\mathbf{U}}_1 e^{\mathrm{i}\omega t} \tag{2.44}$$

$$\frac{\partial \tilde{\mathbf{U}}(t)}{\partial t} \simeq \mathrm{i}\omega \hat{\mathbf{U}}_1 e^{\mathrm{i}\omega t} \tag{2.45}$$

$$\tilde{\mathbf{x}}(t) \simeq \hat{\mathbf{x}}_1 e^{\mathrm{i}\omega t} \tag{2.46}$$

$$\dot{\tilde{\mathbf{x}}}(t) \simeq \mathrm{i}\omega \hat{\mathbf{x}}_1 e^{\mathrm{i}\omega t}, \tag{2.47}$$

with $\hat{\mathbf{U}}_1$ and $\hat{\mathbf{x}}_1$ being the complex amplitudes of the flow variables and motion and ω the frequency being studied. Substitute these series into Eq. (2.43), and group the resulting terms

$$\left(\mathrm{i}\omega\mathbf{I} - \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)\hat{\mathbf{U}}_1 = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \mathrm{i}\omega\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{x}}}\right)\hat{\mathbf{x}}_1, \tag{2.48}$$

where $\mathbf{I}$ is the identity matrix.

The mean flow solution is considered independent of the harmonic motion and its frequency. Per Eq. (2.42) the mean flow can be obtained from one steady simulation. Therefore, only the oscillatory solution needs to be calculated at different frequencies.

While the linear frequency-domain method was first applied to non-deforming meshes, Clark and Hall applied a change of coordinate system that moves with the solid body. They found that this approach was more accurate than the LFD using a non-deforming mesh [47, 48].

The LFD is adequate for linear aeroelastic problems, such as flutter onset. However, the actual mean flow is not equal to the corresponding steady flow, especially at large amplitudes. Since the linear frequency-domain method assumes a constant mean flow, it cannot capture the impact of the harmonic movement on the mean flow. Furthermore, in the presence of highly nonlinear phenomena, such as shocks, the method's accuracy is reduced [49].

2.4 Harmonic balance

Harmonic balance refers to a family of frequency-domain methods. Unlike the LFD, they include a coupling between the different harmonics studied. Therefore, the mean flow, or 0th harmonic, depends on each of the N_h harmonics used. However, this makes studying the effect of the frequency more computationally expensive compared to the linear frequency-domain technique. Dufour et al. compared the harmonic balance and linear frequency-domain methods around an aerofoil with an oscillating flap. They found that in the transonic cases the harmonic balance performed better, even when using only one harmonic [49].

Harmonic balance approaches have been used extensively in fluid applications, both for internal flows in turbomachinery [9, 52, 53] and external flows [20, 49, 54, 55]. They are quite flexible, having been applied to fluid solvers of varying fidelity: from Theodorsen theory [56, Ch. 7] to Reynolds-averaged Navier-Stokes models. Furthermore, they allow the study of several fundamental frequencies, which is relevant for turbomachinery, where the rotors and stators can have different numbers of blades. For the present section, one fundamental frequency ω will be used. The corresponding period is $T = 2\pi/\omega$.

Within the harmonic balance methodology there are two main approaches: the frequency-domain harmonic balance (FDHB) and the time-domain harmonic balance (TDHB). The FDHB transforms the time-domain equations into the frequency domain. The time derivatives are only a function of the corresponding harmonic's amplitude and its frequency. The TDHB, on the other hand, selects at least $2 \cdot N_h + 1$

Figure 2.2: Diagram of time instances in a harmonic balance solution

pseudo-steady time instances and inserts the time derivative as a source term on them. Compared to the FDHB, the main advantage of the TDHB is that the existing machinery of a steady solver can be re-used in order to solve an unsteady problem.

2.4.1 Frequency-domain harmonic balance

Starting from the generic approach in Eq. (2.38), the time-domain solution $\mathbf{U}(t)$ is assumed periodic with fundamental frequency ω. One period of the time-domain solution is discretised into $2 \cdot N_h + 1$ equispaced time instances where $\mathbf{U}(t)$ is evaluated. The time difference between each consecutive instance is

$$\Delta t = \frac{2\pi}{\omega \cdot (2 \cdot N_h + 1)} = \frac{T}{2 \cdot N_h + 1}. \tag{2.49}$$

Figure 2.2 shows a diagram of how the time instances are distributed within a period.

Evaluating Eq. (2.38) in these instances, there is an uncoupled system of equations

$$\begin{cases} \dfrac{\partial}{\partial t}\mathbf{U}(t)\Big|_{t=0} - \mathbf{R}\left(\mathbf{U}(0)\right) & = 0 \\ \quad \vdots & \\ \dfrac{\partial}{\partial t}\mathbf{U}(t)\Big|_{t=T-\Delta t} - \mathbf{R}\left(\mathbf{U}(T-\Delta t)\right) & = 0 \end{cases}. \tag{2.50}$$

However, the value of the time derivative term is unknown.

Similarly to the linear frequency-domain method, the function can be approximated by using a Fourier series expansion. Using N_h harmonics, this approximation is

$$\mathbf{U}(t) \simeq \hat{\mathbf{U}}_0 + \sum_{n=1}^{N_h} \left[\hat{\mathbf{U}}_{2n-1} \cos\left(n\omega t\right) + \hat{\mathbf{U}}_{2n} \sin\left(n\omega t\right)\right], \tag{2.51}$$

where $\hat{\mathbf{U}}_n$ is the nth harmonic solution. There are $2 \cdot N_h + 1$ real harmonic coefficients. While Eq. (2.51) is defined using real-valued coefficients, an equivalent expansion

can be performed with complex-valued coefficients and the complex exponential. In that case, there are N_h complex coefficients and one real coefficient, $\hat{\mathbf{U}}_0$.

Define the full solution vectors of size $m \cdot (2 \cdot N_h + 1)$ for m variables in $\mathbf{U}$

$$\mathbf{U}^* = \begin{bmatrix} \mathbf{U}(0) \\ \vdots \\ \mathbf{U}(T - \Delta t) \end{bmatrix} \qquad \hat{\mathbf{U}}^* = \begin{bmatrix} \hat{\mathbf{U}}_0 \\ \vdots \\ \hat{\mathbf{U}}_{2N_h} \end{bmatrix}, \tag{2.52}$$

where $\mathbf{U}^*$ is the time solution vector and $\hat{\mathbf{U}}^*$ is the frequency solution vector. The corresponding discrete Fourier transform matrix $\mathbf{E}$, such that $\hat{\mathbf{U}}^* = \mathbf{E}\mathbf{U}^*$, is defined as

$$\mathbf{E} = \frac{2}{2 \cdot N_h + 1} \begin{bmatrix} \mathbf{I}\frac{1}{2} & \cdots & \mathbf{I}\frac{1}{2} \\ \mathbf{I}\cos\frac{2\pi \cdot 0}{2 \cdot N_h + 1} & \cdots & \mathbf{I}\cos\frac{2\pi \cdot 2 \cdot N_h}{2 \cdot N_h + 1} \\ \mathbf{I}\sin\frac{2\pi \cdot 0}{2 \cdot N_h + 1} & \cdots & \mathbf{I}\sin\frac{2\pi \cdot 2 \cdot N_h}{2 \cdot N_h + 1} \\ \vdots & \ddots & \vdots \\ \mathbf{I}\cos\frac{2\pi \cdot N_h \cdot 0}{2 \cdot N_h + 1} & \cdots & \mathbf{I}\cos\frac{2\pi \cdot N_h \cdot 2 \cdot N_h}{2 \cdot N_h + 1} \\ \mathbf{I}\sin\frac{2\pi \cdot N_h \cdot 0}{2 \cdot N_h + 1} & \cdots & \mathbf{I}\sin\frac{2\pi \cdot N_h \cdot 2 \cdot N_h}{2 \cdot N_h + 1} \end{bmatrix}, \tag{2.53}$$

where $\mathbf{I}$ is an identity matrix of size $m \times m$ [57]. Matrix $\mathbf{E}$ is a square $m \cdot (2 \cdot N_h + 1) \times m \cdot (2 \cdot N_h + 1)$ matrix.

Obtaining the time derivative in the frequency domain is straightforward. Differentiate Eq. (2.51) with respect to t,

$$\frac{\partial}{\partial t}\mathbf{U}(t) \simeq \sum_{n=1}^{N_h} \left[-n\omega \hat{\mathbf{U}}_{2n-1} \sin(n\omega t) + n\omega \hat{\mathbf{U}}_{2n} \cos(n\omega t) \right]. \tag{2.54}$$

Then, comparing Eq. (2.51) and Eq. (2.54), the frequency-domain derivative matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \omega \cdot \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\mathbf{I} & \cdots & 0 & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -N_h \cdot \mathbf{I} \\ 0 & 0 & 0 & \cdots & N_h \cdot \mathbf{I} & 0 \end{bmatrix}. \tag{2.55}$$

Introducing the expression of the time derivative from Eq. (2.55) and expressing the result in matrix form, the frequency-domain harmonic balance equation is

$$\mathbf{A}\mathbf{E}\mathbf{U}^* - \mathbf{E}\mathbf{R}^*(\mathbf{U}^*) = \mathbf{0}. \tag{2.56}$$

This expression is a system of size $m \cdot (2 \cdot N_h + 1)$, with every m equations corresponding to each harmonic that needs to be balanced.

If $\mathbf{R}(\mathbf{U})$ is linear, then the harmonics are uncoupled. However, if it is nonlinear, there is a coupling between the lower- and higher- order harmonics. One can choose, as in the present derivation, to either compute the discrete Fourier transform of the residual, or to transform the calculation of the residual to the frequency domain. Another alternative is to use a fast Fourier transform, which benefits from choosing a number of time instances that is a power of 2 [58]. The two methods are comparable in their results.

2.4.2 Time-domain harmonic balance

The frequency-domain harmonic balance allows solving complex problems in the frequency domain. However, it changes the equations solved, requiring either a rewrite of the problem or transforming the solution back and forth from the frequency domain.

Define the inverse of the matrix shown in Eq. (2.53), so that $\mathbf{U}^* = \mathbf{E}^{-1}\hat{\mathbf{U}}^*$

$$\mathbf{E}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{I}\cos\frac{2\pi\cdot 0}{2\cdot N_h+1} & \mathbf{I}\sin\frac{2\pi\cdot 0}{2\cdot N_h+1} & \cdots & \mathbf{I}\cos\frac{2\pi\cdot N_h\cdot 0}{2\cdot N_h+1} & \mathbf{I}\sin\frac{2\pi\cdot N_h\cdot 0}{2\cdot N_h+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{I} & \mathbf{I}\cos\frac{2\pi\cdot 2\cdot N_h}{2\cdot N_h+1} & \mathbf{I}\sin\frac{2\pi\cdot 2\cdot N_h}{2\cdot N_h+1} & \cdots & \mathbf{I}\cos\frac{2\pi\cdot N_h\cdot 2\cdot N_h}{2\cdot N_h+1} & \mathbf{I}\sin\frac{2\pi\cdot N_h\cdot 2\cdot N_h}{2\cdot N_h+1} \end{bmatrix}, \quad (2.57)$$

where $\mathbf{I}$ is an identity matrix of size $m \times m$. Like matrix $\mathbf{E}$, $\mathbf{E}^{-1}$ is a square $m \cdot (2 \cdot N_h + 1) \times m \cdot (2 \cdot N_h + 1)$ matrix. The time-derivative matrix in the time domain is defined as

$$\mathbf{D} = \mathbf{E}^{-1}\mathbf{A}\mathbf{E}, \quad (2.58)$$

where $\mathbf{E}^{-1}$ is the inverse discrete Fourier transform matrix, $\mathbf{A}$ is the frequency-domain derivative matrix shown in Eq. (2.55) and $\mathbf{E}$ is the discrete Fourier transform matrix.

For $m = 1$, each term $d_{i,j}$ in matrix $\mathbf{D}$ is defined by

$$\begin{aligned} d_{i,j} = \frac{2\cdot\omega}{2\cdot N_h+1}\sum_{n=1}^{N_h} & -n\cdot\sin\left(\frac{2\pi\cdot n\cdot i}{2\cdot N_h+1}\right)\cdot\cos\left(\frac{2\pi\cdot n\cdot j}{2\cdot N_h+1}\right) \\ & + n\cdot\cos\left(\frac{2\pi\cdot n\cdot i}{2\cdot N_h+1}\right)\cdot\sin\left(\frac{2\pi\cdot n\cdot j}{2\cdot N_h+1}\right). \end{aligned} \quad (2.59)$$

Apply the trigonometric identity $\sin\alpha\cos\beta - \cos\alpha\sin\beta = \sin(\alpha-\beta)$:

$$d_{i,j} = \frac{2\cdot\omega}{2\cdot N_h+1}\sum_{n=1}^{N_h} n\cdot\sin\left[\frac{2\pi\cdot n}{2\cdot N_h+1}\cdot(j-i)\right]. \quad (2.60)$$

From this expression, it can be seen that the diagonal of the $\mathbf{D}$ matrix is 0. Furthermore, since the sine is an odd function, $d_{i,j} = -d_{j,i}$. This means that the time-derivative matrix is antisymmetric. Another important property is that each member of a diagonal has the same value. This is desirable, since it means that the time derivative is not affected by a change in phase of $2\pi/(2 \cdot N_h + 1)$ of the problem.

Transform Eq. (2.56) from the frequency domain back to the time domain,

$$\begin{aligned} \mathbf{E}^{-1}\mathbf{A}\mathbf{E}\mathbf{U}^* - \mathbf{E}^{-1}\mathbf{E}\mathbf{R}^*\left(\mathbf{U}^*\right) = \mathbf{0}; \\ \mathbf{D}\mathbf{U}^* - \mathbf{R}^*\left(\mathbf{U}^*\right) = \mathbf{0}. \end{aligned} \tag{2.61}$$

A standard steady solver can be re-used to implement this method. In that case, the time derivative source term $\mathbf{D}\mathbf{U}^*$ is added to the steady equations and the solution is iterated until convergence. For low-dimensional cases the TDHB problem can also be solved as a coupled system of $m \cdot (2 \cdot N_h + 1)$ equations.

2.4.3 Application within a deforming-mesh framework

While the application of the time-domain harmonic balance is straightforward, there are some important considerations that depend on the problem being studied. The present work deals with coupled aerostructural systems in which the boundary deforms. The fluid mesh is then deformed depending on these boundary values, as described in Sec. 2.1.2.

If the cell volumes remain constant, as is the case of rigid mesh movements, the volume can be taken out of the source term in Eq. (2.61). Otherwise, the source term has to be modified because the cell volume appears in the vector of variables at each time instance. For the Euler equations introduced in Eq. (2.16), the variables used are

$$\mathbf{U} = \begin{bmatrix} \mathcal{V} \cdot \rho \\ \mathcal{V} \cdot \rho\mathbf{v} \\ \mathcal{V} \cdot \rho E \end{bmatrix}, \tag{2.62}$$

where $\mathcal{V}$ is the time-changing volume of the cell.

If using an arbitrary Lagrangian-Eulerian method, the grid velocities appear as a source term in the solution. They can be calculated by multiplying the displacements of each grid point by the time derivative matrix $\mathbf{D}$, defined in Eq. (2.58). The expression for the grid velocities is

$$\dot{\mathbf{z}}^* = \mathbf{D}\mathbf{z}^*. \tag{2.63}$$

They have to be updated only when either the frequency or the mesh displacements are modified, at the start of every FSI iteration.

2.5 Frequency iteration techniques

The harmonic balance methods presented in Sec. 2.4 assume a known, given frequency. This is the case in, for example, forced resonance in turbomachinery. The blades of a stage are excited by a fluid field that changes with a frequency that is proportional to the rotational speed of the turbine. However, many phenomena have an *a priori* unknown frequency. One example of these are limit-cycle oscillations (LCOs), in which the frequency is a crucial parameter.

Thus, the numerical study of LCOs using the harmonic balance method requires calculating the frequency. Ekici and Hall found that once movement amplitude is converged, iterations at a constant but wrong frequency lead to a shifting of the phase in every FSI iteration [6]. In order to find the correct frequency of the problem, some techniques have been developed in order to iterate around a given initial frequency. When used with partitioned solvers, most of them rely on using the structural solver.

The present section deals with the procedure to find the fundamental frequency of the coupled problem and fixing the phase of the solution. First, L^2 residual norm techniques are presented. They attempt to obtain this frequency by minimising a structural residual norm, which adds an extra equation to be solved. This technique and its variations are the current standard for partitioned harmonic balance FSI.

Then, phase-fixing, which consists in setting a constant phase to one of the harmonics, is presented. Finally, a combined phase-fixing frequency-iteration technique is introduced. It guarantees a constant phase while obtaining the correct fundamental frequency of the problem. The combined technique solves the same number of equations as the known-frequency harmonic balance method. Similar techniques have been widely used for monolithic solvers, using lower-fidelity fluid models, but not for partitioned codes.

2.5.1 L^2 residual norm techniques

According to Ekici and Hall, if the frequency used in a harmonic balance calculation for an LCO is close but not equal to the correct frequency, the phase will drift from iteration to iteration while the amplitude will be close to the correct one. They introduced a new equation taking advantage of this, where they attempt to minimise the squared norm of the structural residual from one FSI iteration to the following one [6]. This L^2 residual norm can be expressed in the form

$$Z(\omega) = \left\| \mathbf{D}(\omega)\mathbf{x}^* + \mathbf{R}\mathbf{x}^* - \mathbf{F}^* \right\|^2 , \tag{2.64}$$

where $\mathbf{x}^*$ is the vector of solutions at every time instance, matrix $\mathbf{D}$ is a linear function of ω as per Eq. (2.60), $\mathbf{R}$ is the Jacobian of the structural problem and $\mathbf{F}^*$ is the current iteration's fluid load vector [59].

Assuming that the dependence of $\mathbf{F}^*$ with respect to the frequency is negligible, as argued by Ekici and Hall, the residual function $Z(\omega)$ is quadratic [6, 59]. Then, the minimum of this quadratic function in FSI iteration $n-1$ is expressed by

$$\omega_n = -\omega_{n-1} \cdot \frac{\left(\mathbf{R}\mathbf{x}^* - \mathbf{F}^*\right)^T \mathbf{D}\mathbf{x}^*}{\mathbf{x}^{*T}\mathbf{D}^T\mathbf{D}\mathbf{x}^*}, \tag{2.65}$$

where ω_n is the new value of the fundamental frequency.

However, Yao and Marques found that adding the gradient of the fluid loads with respect to the frequency improved convergence in certain problems [60]. They updated Eq. (2.65) such that,

$$\omega_n = -\omega_{n-1} \cdot \frac{\left(\mathbf{R}\mathbf{x}^* - \mathbf{F}^*\right)^T \left(\mathbf{D}\mathbf{x}^* + \frac{\partial \mathbf{F}^*}{\partial \omega}\right)}{\mathbf{x}^{*T}\mathbf{D}^T \left(\mathbf{D}\mathbf{x}^* + \frac{\partial \mathbf{F}^*}{\partial \omega}\right)}. \tag{2.66}$$

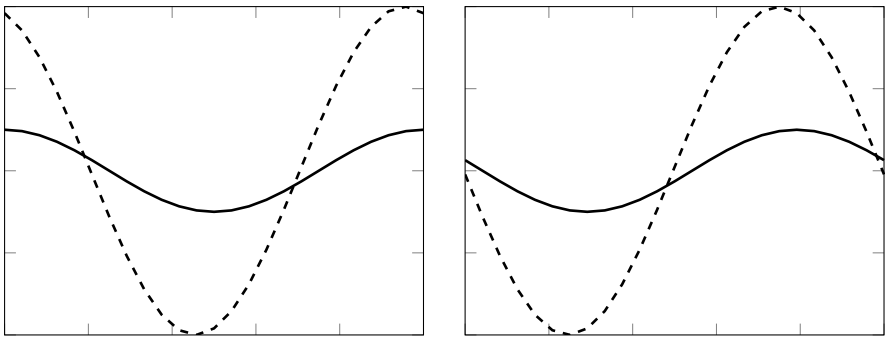
While this method results in a more accurate estimate of the LCO frequency, it requires calculating the $\partial\mathbf{F}^*/\partial\omega$ term. Yao and Marques used finite differences in order to do so, changing slightly the value of ω in order to evaluate the gradient. They updated the estimate of $\partial\mathbf{F}^*/\partial\omega$ every $10-15$ iterations [60, 61].

2.5.2 Phase-fixing

There is an infinite number of equivalent solutions to a periodic problem. Each of these solutions has a different phase, shifting the results. For example, Figure 2.3 shows two different possible solutions to the same LCO problem with two degrees of freedom. The solutions in Fig. 2.3(a) and Fig. 2.3(b) are equivalent: they have the same amplitude and the same relative shift between the two degrees of freedom. While it is possible to obtain a converged result without fixing the phase [6], it is generally better to try to converge towards one solution. Fixing the phase of the problem also eliminates the dependence of the solution on the values used to initialise the calculation.

In cases where the response frequency is given by some external excitation, maintaining the phase of this excitation can be enough. This would be the case of an aerofoil with a forced pitching oscillation and free plunging motion.

In other cases in which there is no external excitation, a given degree of freedom's phase can be kept constant. One of the harmonic components of said degree of freedom can be set to 0, thus fixing the phase. For example, in a pitching and



(a) Original solution (b) Phase-shifted solution

Figure 2.3: Two LCO solutions with two degrees of freedom (continuous and dashed lines) with different phase for one period. The phase difference between the two degrees of freedom and the amplitudes are equal for both solutions

plunging aerofoil the first harmonic sine component of the pitching degree of freedom can be 0.

One further advantage of phase-fixing is that it helps with convergence of the fluid sub-problem [62]. If the phase of the structural solution is kept constant, the flow field will differ less between FSI iterations. Therefore, the previous FSI iteration's fluid solution will be a better initial guess.

2.5.3 Combined technique

In many problems, the frequency is an unknown in the system that needs to be calculated. This leads to more variables than equations. The L^2 norm technique solves this issue by adding another equation to be solved. Phase-fixing eliminates one of the structural unknowns in order to keep a constant phase. Instead of changing the phase of the newly-obtained solution at each iteration, it is also possible to combine phase-fixing and frequency iteration procedures in the structural solver. A similar method is commonly used in simple, monolithic codes [56, Ch. 7] where the frequency derivative of the whole system is obtained. While for partitioned solvers the L^2 norm technique is standard, in this work the approach combining phase-fixing and frequency iteration for monolithic solvers is proposed instead.

In order to implement the combined approach for a partitioned code, the equations of the structural solver have to be rewritten. For example, if the sine component of the first harmonic of the pitch response is set to 0, at FSI iteration $n + 1$

the new solution is obtained from

$$\begin{bmatrix} \frac{\partial}{\partial \alpha_0} & \cancel{\frac{\partial}{\partial \alpha_{1,s}}} & \frac{\partial}{\partial \alpha_{1,c}} & \frac{\partial}{\partial \alpha_{2,s}} & \frac{\partial}{\partial \alpha_{2,c}} & \cdots & \frac{\partial}{\partial \omega} \end{bmatrix} \mathcal{S} \cdot \begin{bmatrix} \delta\alpha_0 \\ \cancel{\delta\alpha_{1,s}}^{0} \\ \delta\alpha_{1,c} \\ \delta\alpha_{2,s} \\ \delta\alpha_{2,c} \\ \vdots \\ \delta\omega \end{bmatrix} = R, \tag{2.67}$$

where α_0 is the mean value, and $\alpha_{i,c}$ and $\alpha_{i,s}$ are the cosine and sine components of the ith harmonic of the pitch response. The exponent $\bullet^n$ represents that it is the value $\bullet$ at FSI iteration n. The new values are

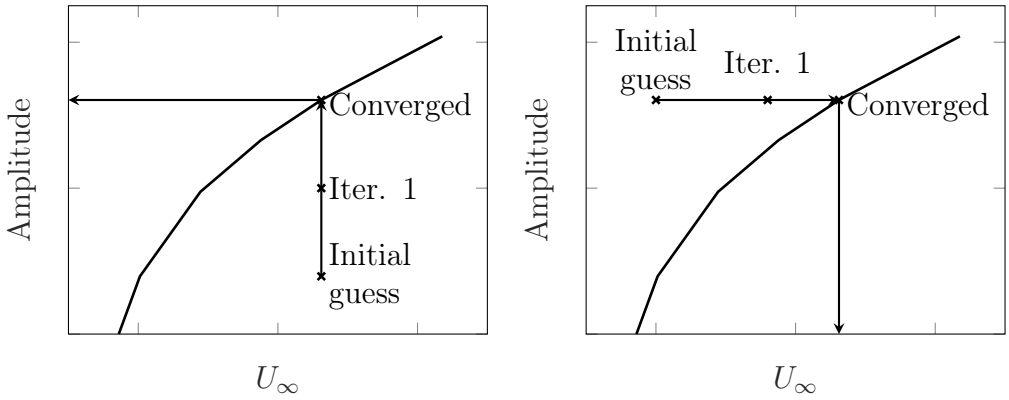
$$\begin{bmatrix} \alpha_0^{n+1} \\ \alpha_{1,c}^{n+1} \\ \alpha_{2,s}^{n+1} \\ \alpha_{2,c}^{n+1} \\ \vdots \\ \omega^{n+1} \end{bmatrix} = \begin{bmatrix} \alpha_0^{n} \\ \alpha_{1,c}^{n} \\ \alpha_{2,s}^{n} \\ \alpha_{2,c}^{n} \\ \vdots \\ \omega^{n} \end{bmatrix} + \begin{bmatrix} \delta\alpha_0 \\ \delta\alpha_{1,c} \\ \delta\alpha_{2,s} \\ \delta\alpha_{2,c} \\ \vdots \\ \delta\omega \end{bmatrix}. \tag{2.68}$$

This approach maintains the chosen component at 0 without any extra processing. However, it requires choosing an appropriate degree of freedom to fix, which is complex in cases with a large number of degrees of freedom.

2.6 Harmonic balance fluid-structure interaction coupling

In order to obtain aeroelastic solutions there needs to be some sort of coupling. In the specific case of LCOs, a given set of flow conditions results in a given amplitude, or amplitudes. Thus, there are two main ways of predicting an LCO: imposing the freestream flow conditions and calculating the LCO amplitude and frequency, and imposing the amplitude and calculating the corresponding frequency and freestream airspeed. The amplitude for the second method can be defined in several ways. Two examples are imposing the amplitude of one of the degrees of freedom [62, 63] and the mean energy of the movement [7]. This method requires solving another equation for the freestream velocity. One possible approach is introducing the minimiser of the L^2 structural residual norm as a function of the velocity [7].

Imposing the freestream flow conditions, on the other hand, does not require the extra equation. However, it requires for the LCO to have a non-zero amplitude at



(a) Constant freestream conditions (b) Constant movement amplitude

Figure 2.4: Comparison of harmonic balance FSI definition of cases

the given freestream conditions. A comparison of these two methods is shown in Fig. 2.4. These two approaches were compared by Li and Ekici and found to give very similar results [59].

The system of equations in Eq. (2.20) can be extended to include the base frequency as a parameter, obtaining

$$\begin{bmatrix} \mathcal{S} \\ \mathcal{F} \\ \mathcal{M} \end{bmatrix} + \begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \mathbf{u}} & \frac{\partial \mathcal{S}}{\partial \omega} & \frac{\partial \mathcal{S}}{\partial \mathbf{w}} & \frac{\partial \mathcal{S}}{\partial \mathbf{z}} \\ 0 & \frac{\partial \mathcal{F}}{\partial \omega} & \frac{\partial \mathcal{F}}{\partial \mathbf{w}} & \frac{\partial \mathcal{F}}{\partial \mathbf{z}} \\ \frac{\partial \mathcal{M}}{\partial \mathbf{u}} & \frac{\partial \mathcal{M}}{\partial \omega} & 0 & \frac{\partial \mathcal{M}}{\partial \mathbf{z}} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \omega \\ \Delta \mathbf{w} \\ \Delta \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{2.69}$$

The most important change with respect to the steady method is the addition of the dependence of each of the fields on the fundamental frequency, ω.

Like in the steady case, the system of equations in Eq. (2.69) is larger than each individual problem. Obtaining the derivative terms and solving the system would be impractical in cases with enough degrees of freedom. Similarly, instead of using a Newton-Raphson method the problem can be solved using a BGS method. The loads from the fluid solver appear as a Neumann boundary condition in the solid solver and the displacements obtained by the solid solver appear as a Dirichlet boundary condition in the fluid and mesh solvers. The frequency can be obtained by the structural solver using the methods described in Sec. 2.5.1 or Sec. 2.5.3, for example. It can then be imposed as a parameter of the fluid and mesh solvers. The grid velocity can then be obtained by the mesh solver from Eq. (2.63).

2.6.1 Prescribed freestream conditions harmonic balance coupling algorithm

Figure 2.5 shows a flowchart describing the harmonic balance fluid-structure interaction direct algorithm developed for the present work. It shows in green the steps performed by the coupler, in blue those by the fluid solver and in red those by the structural solver. In order to start the solution process, an initial guess for the amplitudes of each degree of freedom and LCO frequency needs to be provided. Results calculated at other freestream conditions can provide guesses for the amplitudes and frequency. If those are unavailable, the flutter frequency can give a good initial guess for the LCO frequency close to the Hopf point. In that region the eigenvector corresponding to the fluttering mode, multiplied by a small amplitude, can be used as a guess for the amplitude.

The initial displacements of the boundary are obtained from the initial guess. They are transmitted to the fluid solver, which applies them as a Dirichlet boundary condition. The fluid mesh is deformed. The grid velocity at each node, which depends on the initial guess for the frequency, is obtained by means of Eq. (2.63). Then, the fluid code solves the fluid harmonic balance equations. It provides as an output the loads at the boundary, which are then interpolated to the solid boundary. The solid boundary loads are transmitted to the solid solver, which solves the solid harmonic balance equations.

If the structural model is nonlinear, the solid solver is iterated until either the maximum number of iterations or convergence is reached. One example of a convergence criterion is the value of the change in nodal displacement between solid iterations. Provided that the solid solution is converged at the end, it does not matter whether the prescribed level of convergence is reached during the intermediate FSI iterations. From the solid solver, the new frequency and solid boundary displacements are obtained. If the absolute value of the change in frequency and norm of the boundary displacements vector are lower than a given tolerance, the method has converged. Otherwise, if the number of FSI iterations is lower than the maximum number set, the displacements are interpolated and applied to the fluid boundary. The new frequency guess is also imposed as a parameter of the fluid and mesh solvers. These values may be relaxed according to the methods described in Sec. 2.1.2. This procedure repeats until the tolerances are met or the maximum number of FSI iterations is reached.

If after the maximum number of FSI iterations the required tolerances are not met, the solution is generally not accepted. However, for some specific cases, such as flutter calculations, the objective of the FSI calculation is to obtain the aerodynamic response to some structural input at some given conditions. In that case, the fluid

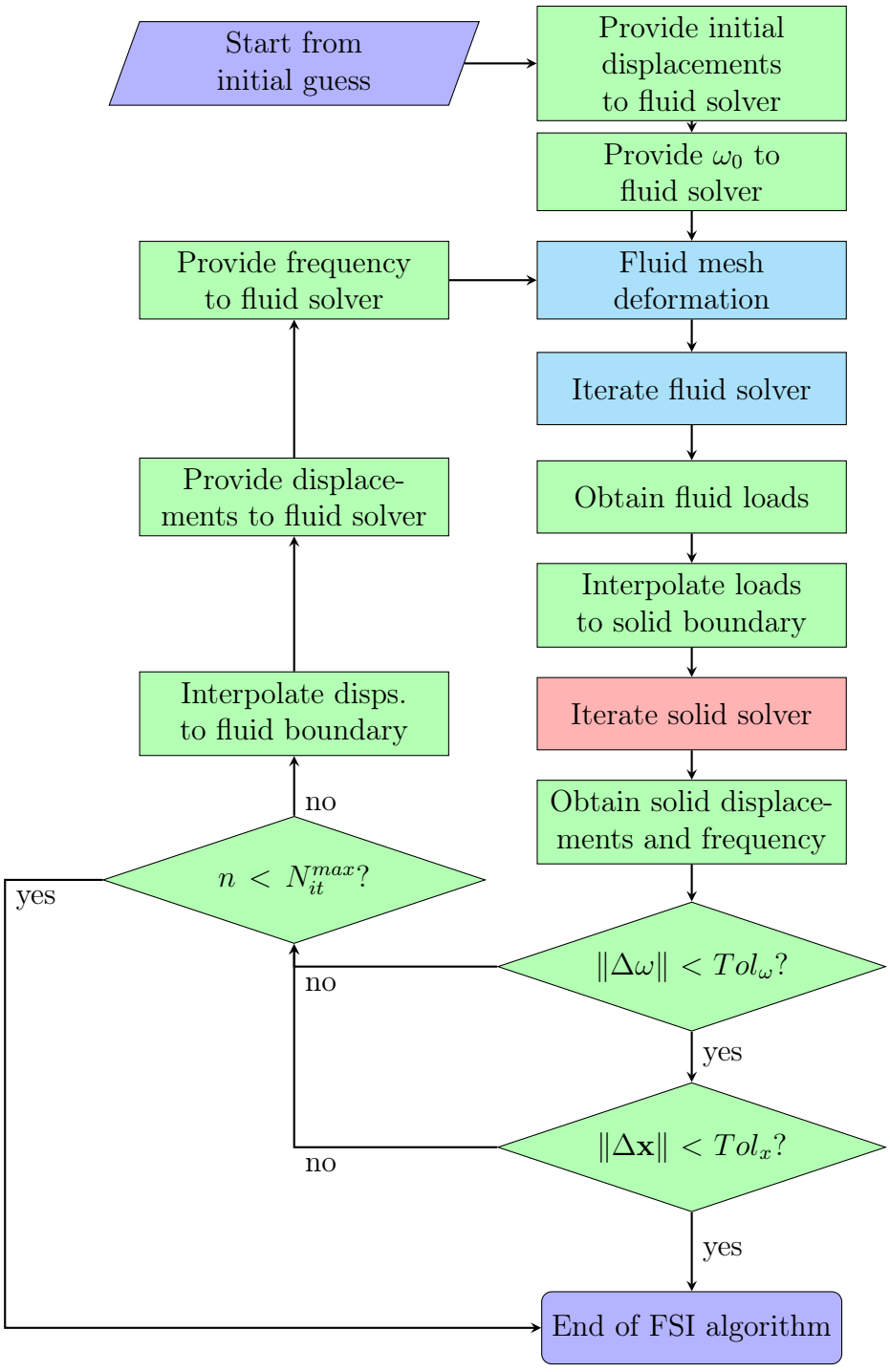


Figure 2.5: Flowchart describing the harmonic balance fluid-structure interaction algorithm. In light blue, the actions carried out by the fluid solver; in red, those by the structural solver; in green, those by the coupler.

solution will be accepted after the first FSI iteration independently of the level of convergence.

In the present work, the fluid solver implements the time-domain harmonic balance to take advantage of the already-existing machinery. The solid solver is written in the frequency domain to more easily implement the frequency iteration algorithm described in Section 2.5.3. Since the problem is linear, the structural system of equations is only solved once. The interpolation of loads and displacements is local to each time-domain instance. Therefore, the solid solver transforms the loads to the frequency domain and the displacements it calculates to the time domain. A more detailed explanation of the two-degree-of-freedom rigid body motion integrator is presented in Appendix C.

2.6.2 Fluid-structure interaction framework

For the present work CUPyDO [37, 64], an open-source FSI coupling code developed at the University of Liège[1], is used and extended. CUPyDO uses a strong, partitioned approach in order to communicate between solid and fluid solvers. It implements both the BGS with relaxation [37, 64] and the IQN-ILS [65, 66] approaches. The BGS algorithm was extended to work with multiple time instances. CUPyDO includes a C++ kernel which is wrapped using SWIG [67] to Python in order to combine the former's high efficiency with the latter's greater flexibility. The extensions coded in order to implement the harmonic balance method within this framework are presented in Sec. A.2 of Appendix A.

Parallelisation

CUPyDO can be compiled either in serial or in parallel. When compiled in parallel, communication is handled via OpenMPI. In this case, the numerics are solved using PETSc, the Portable, Extensible Toolkit for Scientific Computation [68][2]. CUPyDO uses the mpi4py [69] and petsc4py [70] libraries to provide the necessary Python wrappers.

2.6.3 Structural solver

The structural solver used is a pitch-plunge rigid body motion integrator. The original code was previously developed at the University of Liège [64][3] and has subsequently been extended for this work in order to implement the frequency-domain approach, the combined frequency iteration technique of Sec. 2.5.3 and

[1] `https://github.com/ulgltas/CUPyDO`

[2] `https://petsc.org`

[3] `https://github.com/ulgltas/NativeSolid`

adjoint methods. A derivation of the frequency-domain equations for this solver can be found in Appendix C and of the adjoint-based gradients in Appendix D.

2.6.4 Fluid solver

The fluid code used is SU2 [8], an open-source computational fluid dynamics suite[4]. SU2 has adjoint, time-domain harmonic balance [9] as well as steady fluid-structure adjoint capabilities [36, 71]. It has also been used for coupled aeroacoustic optimisation using a time-marching method [72]. In the pre-existing code, the frequencies and period used by the time-domain harmonic balance approach were taken from each case's configuration file. Therefore, a way to modify these parameters programmatically was implemented for this project in order to enable the use of the combined frequency iteration-phase fixing technique. These changes are explained in Sec. A.1 of Appendix A.

Mesh deformation in SU2

Within SU2, mesh deformation capabilities are based on a linear elastic solution. The displacements are imposed on a given surface, as boundary conditions of the problem. The stiffness of each element can be chosen in two main ways: inverse distance to the deformed surface and inverse element volume. The objective in both cases is to have higher rigidity where it is more needed, so that those elements are less deformed than they otherwise would. With constant stiffness ill-shaped elements are much more likely. For the present work, the unsteady mesh capabilities of SU2 were extended to allow mesh deformation and to calculate the grid velocities at each point in harmonic balance calculations.

2.7 Forced-pitch, free-plunge aerofoil

In the present section, the results of applying the harmonic balance fluid-structure interaction framework to a simple 2D case using a rigid body motion integrator, CUPyDO and SU2 are shown. The test case consists in a symmetric 2D NACA 64A010 aerofoil subjected to a forced pitch motion and free to move in the plunge direction. The main objective of the test case is to verify the harmonic balance-based coupling between the solvers. For this test case, CUPyDO was extended to work with multiple time instances using the BGS algorithm for the coupling.

A secondary objective of the case is to verify the implementation of the frequency-domain rigid body motion integrator described in Appendix C. A similar case was already used by Blanc et al. in order to verify their coupled approach [73].

[4] `https://github.com/su2code/SU2`

A diagram of the structure is shown on Fig. 2.6: the forced pitch motion is $\alpha(t) = \alpha_0 + \Delta\alpha \cdot \sin(\omega \cdot t + \varphi)$ and is positive in the clockwise direction. The plunge motion, h, has a counteracting spring-mass-damper system and is positive downwards.

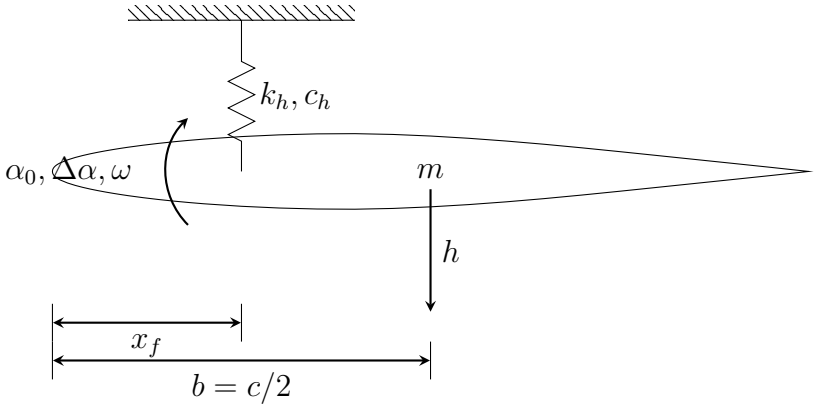


Figure 2.6: Forced-pitch free-plunge aerofoil

The main parameters of the system are shown in Table 2.1. Some of them, such as the imposed frequency of the movement $\omega = 3.89 \cdot \sqrt{k_h/m}$, match those provided by Blanc et al. [73]. However, the parameters they included do not define the test case fully. In particular, neither the reduced frequency nor a freestream velocity index are provided. Using the original structural parameters and the SU2-determined freestream parameters obtained from the Reynolds and Mach numbers led to very high plunging amplitudes, of the order of $20 \cdot b$. In order to reduce this amplitude, the mass and stiffness were increased by two orders of magnitude. Furthermore, the original case did not include any damping. Since part of the objective of this case was to verify the harmonic balance structural solver, some damping was included. The damping ratio of this test case is $\zeta = c_h / \left(2 \cdot \sqrt{k_h \cdot m}\right) = 0.1$. The reduced frequency of the forced motion is $k = \omega \cdot b / U_\infty = 0.0227$, which is low. The pitching motion is applied around the flexural axis, at x_f.

2.7.1 Simulation setup

Since the aim of the present test case is the verification of the implementation, grid convergence is not studied. The Reynolds-averaged Navier-Stokes equations are solved with Menter's $k - \omega$ SST turbulence model [74, 75]. In order to verify the present harmonic balance coupled FSI approach, a time-marching solution was obtained with the same solvers and mesh.

Parameter	Value
M_∞ [−]	0.796
U_∞ [$\mathrm{m\,s^{-1}}$]	270.876
Re [−]	13×10^6
k_h [$\mathrm{N\,m^{-1}}$]	3000
c_h [$\mathrm{kg\,s^{-1}}$]	189.737
m [kg]	300
x_f/c [−]	0.25
α_0 [°]	0
$\Delta\alpha$ [°]	1.01
ω [$\mathrm{rad\,s^{-1}}$]	12.301
φ [rad]	0

Table 2.1: Parameters of transonic forced pitch free plunge test case

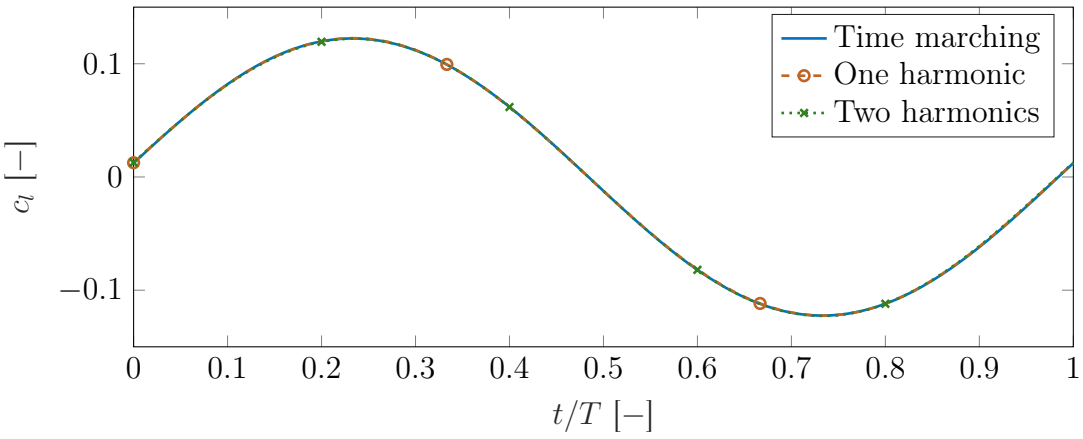
The time-marching simulation uses a second-order dual time stepping method. The Courant-Friedrichs-Lewy (CFL) number of the internal pseudo-time iterations was initially set to 10 and allowed to adapt. The time step was $\Delta t = \frac{T}{100} \simeq 0.005\,108\,\mathrm{s}$ and the simulation ran for ten time periods.

Two cases were run with the harmonic balance method: one with one harmonic and another with two harmonics. Both of them used a constant CFL number of 5.

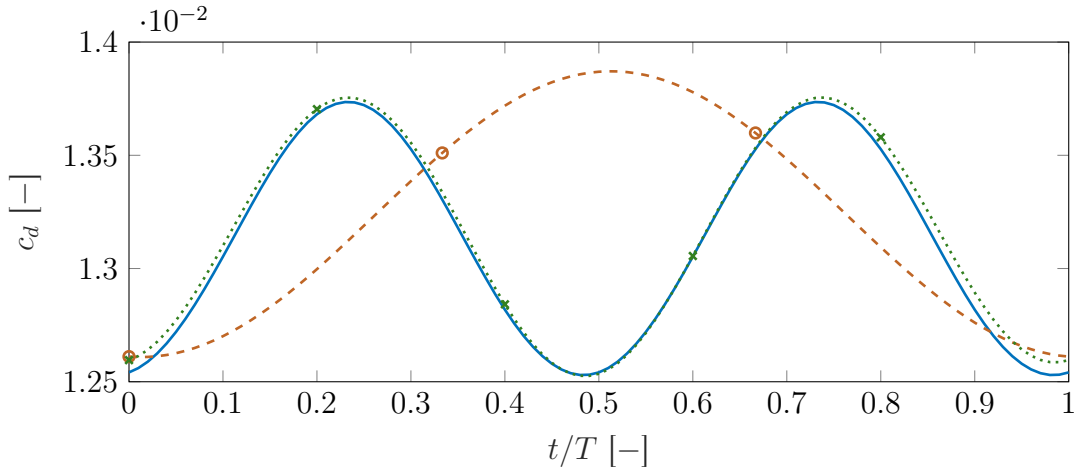
2.7.2 Comparison of unsteady results

The load coefficient time responses are shown in Fig. 2.7 as a function of t/T for the last time period. At this point, time convergence had been attained. Figure 2.7(a) shows the evolution of the lift coefficient during one time period, while Fig. 2.7(b) shows the drag coefficient. They compare the time-marching results, in blue, with the harmonic balance predictions. In orange is the one-harmonic case, with the reconstructed signal appearing as a dashed orange line. Each time instance is a circle. While the match for the lift coefficient is quite close, for the drag coefficient there are significant differences in the reconstruction. These differences appear because of the second-order effects on the drag. The two-harmonics case is plotted in green, with its reconstruction appearing as a dotted line. The time instances are represented as crosses. While the lift coefficient changes very little, the drag coefficient shows a marked improvement compared to the one-harmonic case.

The time-marching simulation predicts a mean drag coefficient of $\bar{c}_d = 0.013124$. Using one harmonic, the predicted mean drag is $\bar{c}_d = 0.013240$, while using two harmonics it is $\bar{c}_d = 0.013155$. These differences are small, despite the fact that in



(a) Lift coefficient



(b) Drag coefficient

Figure 2.7: Comparison of load coefficients for one cycle between time-marching and harmonic balance methods as a function of t/T

order to properly represent the drag coefficient's unsteady behaviour at least two harmonics are required. This can be seen in the time-marching curve in Fig. 2.7(b), which has two peaks. However, the second harmonic aliases to the first harmonic in the one-harmonic case. Therefore, it does not affect the value of the mean drag, which explains the lack of significant change as the number of harmonics increases.

Besides the load coefficients, the structural response is also important. The non-dimensional plunge time responses are compared in Fig. 2.8, scaled by the half-chord, b. The time-marching computation appears as a solid blue line. For the harmonic balance results, the time instances and the harmonic reconstructions are plotted. Each time instance is represented as an orange circle for the one-harmonic solution and a green cross for the two-harmonics solution. The reconstructions are an orange dashed line and a green dotted line, respectively. The plunge motion results show a very close match between the harmonic balance and time-marching solutions. This is expected from the lift in Fig. 2.7(a).

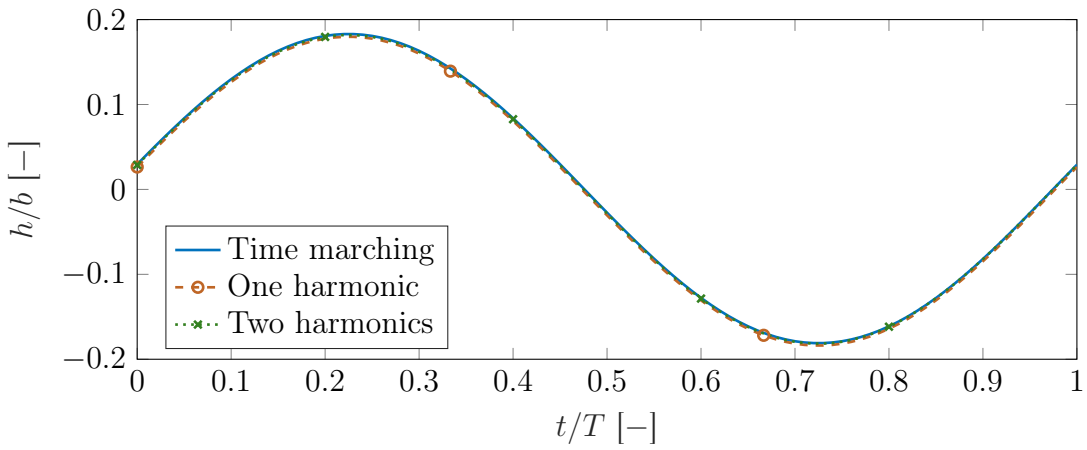


Figure 2.8: Comparison of plunge response for one cycle between time-marching and harmonic balance methods as a function of t/T

To summarise, a one-degree-of-freedom case has been solved using the harmonic balance method, with its predictions showing good agreement with those obtained from the time-marching method. As was expected, the reconstruction of the drag coefficient with only one harmonic was poor, but the lift coefficient and plunge responses matched well. Finally, adding one harmonic significantly improved the match of the drag coefficient.

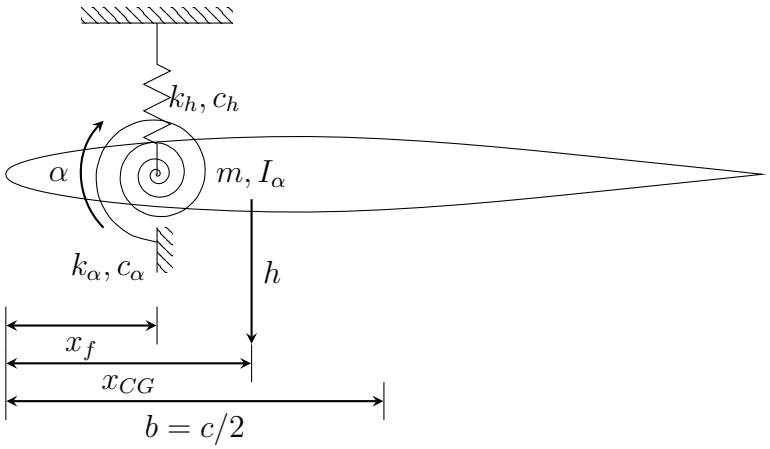


Figure 2.9: Free pitch, free plunge 2D aerofoil

2.8 Two-degree-of-freedom aerofoil

A known-frequency test case using the harmonic balance method was verified in the previous section. In this second test case, the unknown-frequency coupling described in Fig. 2.5 is applied in order to obtain the amplitude of a limit-cycle oscillation. It consists in a NACA 64A010 aerofoil in transonic conditions that pitches and plunges freely with linear springs. The test case has been used to verify different solvers, including reduced order models [7, 59, 60, 62, 63, 76].

The structural model is shown in Fig. 2.9. The aerofoil has a mass m and a moment of inertia, I_α. The pitching motion, α, is defined around the flexural axis, $x_f = 0.4 \cdot b$, and is positive in the clockwise direction. The plunging motion, h, is positive downwards. In the plunge direction, the structural restoring force is provided by a linear spring, k_h, but there is no damping, $c_h = 0$. The pitch restoring force is provided by a torsional spring, k_α with zero damping, $c_\alpha = 0$. The dimensional values of the structural parameters are in Table 2.2.

Table 2.3 gives the values of the non-dimensional parameters that define the problem: M_∞ is the freestream Mach number, $\mu = \frac{m}{\pi \cdot \rho_\infty \cdot b^2}$ is the mass ratio, ω_h/ω_α is the ratio between the natural frequencies of the pitch and plunge degrees of freedom, $x_\alpha = \frac{x_{CG} - x_f}{b}$ is the static imbalance and $r_\alpha^2 = \frac{I_\alpha}{m \cdot b^2}$ is the radius of gyration squared. The pitch and plunge natural frequencies are wind-off and uncoupled, so $\omega_h = \sqrt{k_h/m}$ and $\omega_\alpha = \sqrt{k_\alpha/I_\alpha}$. The reduced velocity, $\tilde{U} = \frac{U_\infty}{\omega_\alpha \cdot 2 \cdot b}$, varies and is approximately 3 at the flutter point.

The freestream Mach number, $\mathrm{M}_\infty = 0.8$, results in transonic flow over the aerofoil. At $\alpha = 0$ there are supersonic regions on both sides of the aerofoil that end

Parameter	Value
b [m]	0.5
m [kg]	50.33
I_α [kg m^2]	9.437
c_h [kg s]	0
c_α [kg m^2 s]	0
k_h [N m^{-1}]	107 559.2
k_α [N m rad^{-1}]	80 669.4

Table 2.2: Structural parameters used for transonic NACA 64A010 test cases

Parameter	Value
M_∞	0.8
μ	75
ω_h/ω_α	0.5
x_α	0.25
r_α^2	0.75

Table 2.3: Non-dimensional parameters of transonic NACA 64A010 test case

in a shockwave. The presence of strong shockwaves greatly affects the behaviour of this aeroelastic system, leading to a change in the linear flutter speed and the appearance of a limit-cycle oscillation.

2.8.1 Simulation setup

In order to solve the problem SU2 is used for the fluid part of the model. The Euler equations with the time-domain harmonic balance method are used to solve the flow.

The solid part is solved using a pitch-plunge rigid body motion integrator [64][5], in which a frequency-domain approach and the frequency iteration technique described in Section 2.5.3 have been implemented during the course of this project. While the behaviour of the structure is linear, the aerofoil is pitched and plunged geometrically. That is, the positions of the boundary nodes are modified depending on the values of pitch and plunge, which leads to a dependence of the pitching moment on the value of the pitch degree of freedom. An alternative approach, widely used in panel codes, is to impose the velocity as a boundary condition.

[5] `https://github.com/ulgltas/NativeSolid`

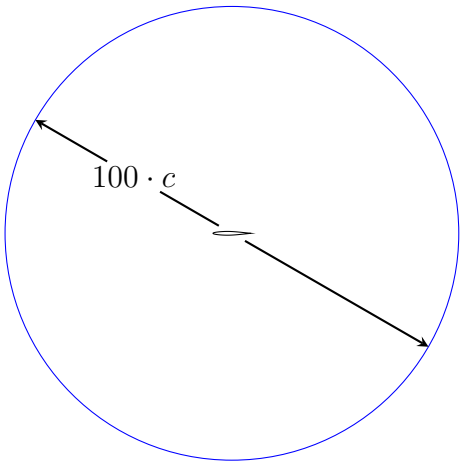


Figure 2.10: Domain studied

The CFL number was modified depending on the frequency. The higher the frequency, the lower the maximum CFL that led to a converged solution. The CFL number required for convergence did not show a strong dependence on amplitude. The setup for the flutter study used a small amplitude to limit the nonlinear behaviour of the flow and required the same CFL number at the same frequency as the setups used to study limit-cycle oscillations with significantly higher amplitudes.

Both solvers use the same mesh in the boundary. Therefore, the matching mesh interpolation method is used. This method ties each node on the fluid boundary to a node on the solid boundary and vice-versa. The process used in order to obtain the lift and moment from the boundary forces is detailed in Appendix C. A block Gauss-Seidel algorithm was used for the coupling.

2.8.2 Steady grid convergence study

First, five O-meshes were created with Gmsh [77] in order to obtain the point at which the solution did not depend on the mesh. The computational domain is shown in Fig. 2.10. The meshes were structured, using quadrilateral elements. Their characteristics are shown in Table 2.4, with N_{el} being the total number of elements over the domain and n being the number of nodes on the aerofoil. The external boundary, blue in Fig. 2.10, had farfield boundary conditions imposed. Since an inviscid flow solver was chosen, the aerofoil itself had free-slip boundary conditions, with the velocity calculated as a function of the mesh displacements. All meshes were used for both aerodynamic and aeroelastic grid convergence studies.

Since the aerofoil is symmetric and the angle of attack is $\alpha = 0$, the steady lift

Mesh	N_{el}	n
Mesh 0	1700	50
Mesh 1	3800	100
Mesh 2	8400	200
Mesh 3	18 000	400
Mesh 4	39 200	800

Table 2.4: Characteristics of meshes for transonic NACA 64A010 test case

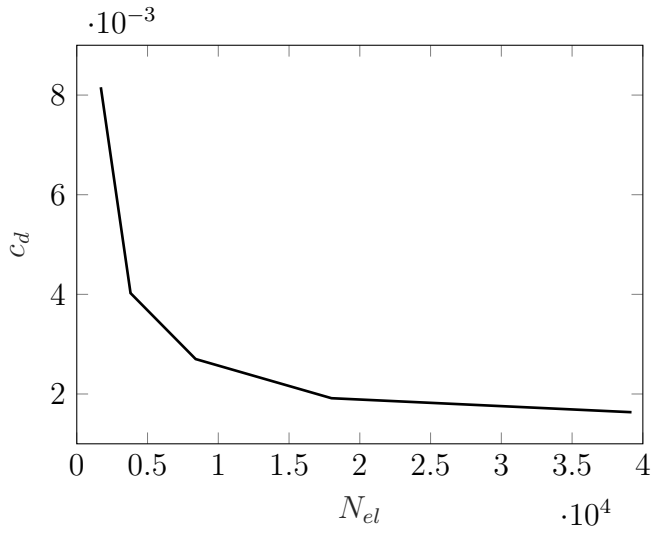


Figure 2.11: Drag coefficient as a function of mesh size

coefficient will be negligible. Therefore, the zero-angle-of-attack drag coefficient is the parameter chosen to study grid convergence. Its evolution with increasing mesh density is shown in Fig. 2.11. The drag coefficient tends to decrease as mesh density increases, with significant changes until Mesh 3.

2.8.3 Flutter point calculation

It is important to study how the mesh affects the coupled aerostructural solution, which may be different from its effect on the steady aerodynamic loads such as the drag. Since the flutter point can be obtained through a linear aeroelastic analysis of the system, it is less computationally expensive than the full limit-cycle oscillation. Furthermore, it provides a starting point for the nonlinear study. A mesh convergence study of the flutter point is described in this section.

In order to obtain the flutter point, the approach devised by Güner based on

interpolation between reduced frequencies and dynamic-mode interpolation (DMI) is used. The open-source Python code aedmi[6] was used for this purpose. However, instead of dynamic mode decomposition, the time-domain harmonic balance is used. The two methods were reported by Güner to be equivalent [78].

For this problem, the $p-k$ method is used to find the flutter point. This technique was applied to a flutter analysis by Irwin and Guyett [79]. Then, Jocelyn Lawrence and Jackson described it in further detail, as the British flutter method. They presented a graphic approach in order to obtain the correct value of the critical flutter speed and frequency [80]. The name $p-k$ method was used by Hassig, who also introduced an iterative solution procedure that replaced the graphic approach [81]. The method assumes that the system undergoes a damped sinusoidal motion

$$\mathbf{q}\left(t\right) = \hat{\mathbf{q}}^{*} \cdot e^{p\frac{t \cdot U_{\infty}}{b}}, \tag{2.70}$$

where $\mathbf{q}$ is the time-dependent vector of modal displacements, $\hat{\mathbf{q}}^{*}$ is the vector of modal amplitudes, U_{∞} is the freestream velocity, b is the half-chord and $p = g + \mathrm{i}k$ is a complex-valued non-dimensional parameter. Its real part, g, corresponds to the damping of the system, while its imaginary part, $k = \omega \cdot b/U_{\infty}$, is the reduced frequency. The equations for the structural problem are

$$\mathbf{M}_{q}\ddot{\mathbf{q}} + \mathbf{C}_{q}\dot{\mathbf{q}} + \mathbf{K}_{q}\mathbf{q} = \mathbf{f}_{q}, \tag{2.71}$$

where $\mathbf{M}_{q}$ is the modal mass matrix, $\mathbf{C}_{q}$ the modal damping matrix, $\mathbf{K}_{q}$ the modal stiffness matrix, $\ddot{\mathbf{q}}$ the vector of modal accelerations, $\dot{\mathbf{q}}$ the vector of modal velocities and $\mathbf{f}_{q}$ is the vector of modal aerodynamic forces.

Assuming that the aerodynamic forces are linear with respect to the amplitude of the structural modes, $\mathbf{f}_{q}$ is given by

$$\mathbf{f}_{q}\left(t\right) = \frac{1}{2}\rho_{\infty}U_{\infty}^{2}\mathbf{Q}\left(p\right)\hat{\mathbf{q}}^{*} \cdot e^{p\frac{t \cdot U_{\infty}}{b}}, \tag{2.72}$$

where ρ_{∞} is the freestream density and $\mathbf{Q}\left(p\right)$ is the modal aerodynamic force matrix. This assumption is valid for low values of the modal amplitudes, $\hat{\mathbf{q}}^{*}$, that do not result in aerodynamic nonlinearities. Substituting Eqs. (2.70) and (2.72) into Eq. (2.71) and then operating leads to

$$\left[\frac{U_{\infty}^{2}}{b^{2}}\mathbf{M}_{q}p^{2} + \frac{U_{\infty}}{b}\mathbf{C}_{q}p + \mathbf{K}_{q} - \frac{1}{2}\rho_{\infty}U_{\infty}^{2}\mathbf{Q}\left(p\right)\right]\hat{\mathbf{q}}^{*} = \mathbf{0}. \tag{2.73}$$

The modal aerodynamic force matrix depends on the parameter p. Assuming that the real part of this parameter, the damping, has a small effect on matrix $\mathbf{Q}$,

[6] `https://gitlab.uliege.be/am-dept/aedmi`

Eq. (2.73) becomes

$$\left[\frac{U_\infty^2}{b^2}\mathbf{M}_q p^2 + \frac{U_\infty}{b}\mathbf{C}_q p + \mathbf{K}_q - \frac{1}{2}\rho_\infty U_\infty^2 \mathbf{Q}\left(\mathrm{i}k\right)\right]\hat{\mathbf{q}}^* = \mathbf{0}. \tag{2.74}$$

This equation has a non-trivial solution for the eigenvalues, p, of the left-hand side. Since $\mathbf{Q}$ depends on the imaginary part of p, the problem has to be solved iteratively. Flutter onset occurs at the airspeed at which the real part of p, g, changes sign from negative to positive.

The modal aerodynamic force matrix at each reduced frequency can be obtained by applying the principle of virtual work to the distribution of forces [56, 82]. A fluid simulation at reduced frequency k can be carried out for each mode shape. The matrix formulation of $\mathbf{Q}\left(\mathrm{i}k\right)$ is then

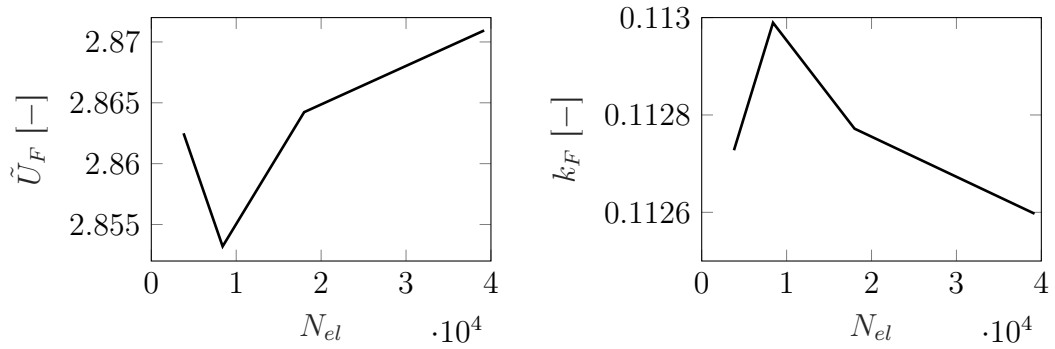
$$\mathbf{Q}\left(\mathrm{i}k\right) = \mathbf{W}^T\mathbf{N}\left(\mathrm{i}k\right), \tag{2.75}$$

where $\mathbf{W}$ is the matrix of wind-off mode shape displacements at each interface cell centre and $\mathbf{N}$ is the matrix of aerodynamic forces at each interface cell. $\mathbf{N}$ can be calculated by integrating the pressure over each cell. Matrix $\mathbf{W}$ and matrix $\mathbf{N}$ have the same size. Since chordwise displacements are negligible at low amplitudes (and 0 at the linear limit) for the pitching mode and 0 for the plunging mode, the two matrices are of size $n \times N_m$, where n is the number of cells on the surface and N_m is the number of modes being taken into consideration for the flutter calculation. Matrix $\mathbf{Q}$ is $N_m \times N_m$.

In the present work, the pressure distribution is obtained from the harmonic balance CFD simulations referenced earlier. The pressure depends on the frequency of motion. In order to avoid the costly recalculation of these values at each studied frequency, $\mathbf{Q}$ is calculated at a reduced number of frequencies. Then, it is interpolated based on the value of k used as a guess. Other flutter methods, such as the g method [83], can also be used to calculate the flutter point [78, 84].

The amplitudes imposed are $\Delta h/b = 0.01$ and $\Delta\alpha = 0.005\,\mathrm{rad}$ for the plunge and pitch degrees of freedom, respectively. It was reported by Güner that for the NACA 64A010 aerofoil and transonic flow at M = 0.8 the nonlinear behaviour of the flow was negligible for pitch amplitudes of $\Delta\alpha \leq 1^\circ \simeq 0.017\,\mathrm{rad}$ at a reduced frequency $k = 0.202$ [78].

First, two frequencies ($k = 0.075, k = 0.15$) corresponding roughly to the wind-off natural frequencies of the two structural modes at the expected flutter speed are studied. In order to allow for the possible effect of nonlinear behaviour with respect to the frequency, a third, middle frequency was added close to the originally predicted flutter frequency ($k = 0.11$). This process is repeated for both degrees of freedom and all the meshes studied in the previous section.



(a) Flutter velocity index

(b) Flutter reduced frequency

Figure 2.12: Flutter point as a function of mesh size

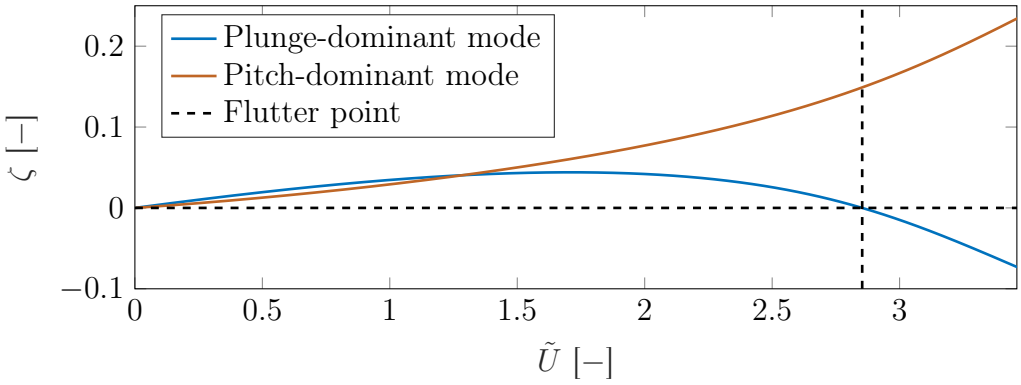
The change in flutter velocity index with respect to mesh density is shown in Fig. 2.12(a), while the corresponding flutter reduced frequencies are plotted in Fig. 2.12(b). The results obtained using the coarsest grid are excluded from the figure because the predicted flutter mechanism is wrong. It predicts that the pitch-dominant mode flutters instead of the plunge-dominant mode. Since there is little difference between the flutter results obtained from Meshes 1 to 4, Mesh 2 from Table 2.4 is chosen. The flutter point obtained using this mesh is sufficiently converged. The predicted reduced flutter speed is 0.6% lower than that of the finest mesh, Mesh 4, while the reduced frequency is 0.3% higher.

The evolution of the damping ratios and natural frequencies of the system with respect to the velocity index for the chosen mesh is shown in Fig. 2.13. As expected, both the pitch- and plunge-dominant modes of the coupled system are stable at low speeds. This appears as positive damping in Fig. 2.13(a). However, around $\tilde{U} = 1.5$ the damping of the plunge-dominant mode reaches a maximum and starts to decrease. Finally, at $\tilde{U}_F = 2.853$ it reaches 0 and the system flutters.

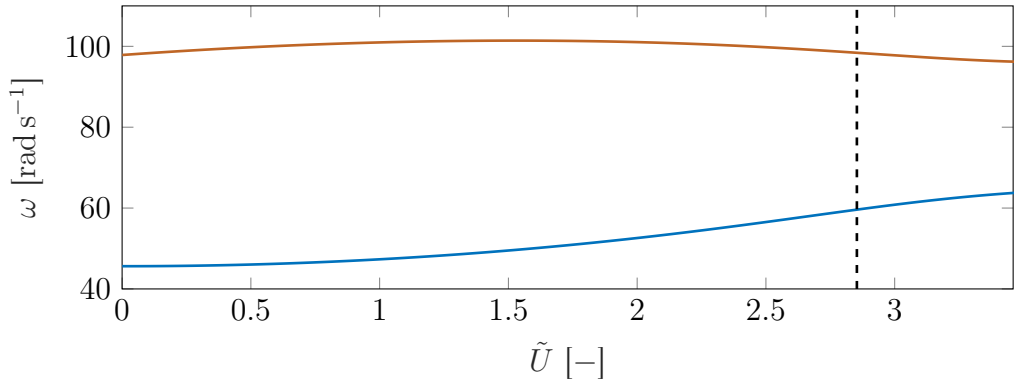
Since the flutter problem is linear, adding more harmonics to the procedure should not significantly impact the flutter point. The same procedure was repeated with two harmonics in order to confirm this behaviour. Table 2.5 compares the flutter point obtained with one and two harmonics for Mesh 2, with the digits that differ in bold. As expected, the differences are small. They are roughly two orders of magnitude smaller than those between Meshes 2 and 3.

2.8.4 Limit-cycle oscillation prediction

Once the appropriate mesh for the problem is chosen, the calculation of the amplitude of limit-cycle oscillations can be carried out. The reduced velocity is chosen to



(a) Damping



(b) Frequency

Figure 2.13: Results of the flutter analysis

N_h	$\tilde{U}_F$	k_F
1	2.853**2**	0.11**298**
2	2.853**0**	0.11**301**

Table 2.5: Convergence of flutter point with number of harmonics

be higher than that at the flutter point obtained in the previous section. The flutter point also provides an initial guess for the fundamental frequency of the limit-cycle oscillation.

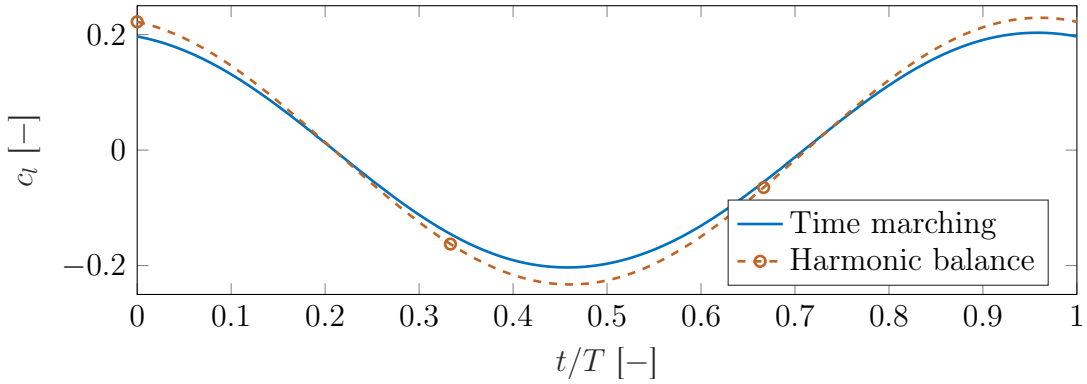
Case 1 uses a reduced velocity $\tilde{U} = 2.944$. Its fluid parameters are shown in Table 2.6 and its structural parameters are those used in order to calculate the flutter point. A comparison of the results obtained using a time-marching and a one-harmonic HB simulation for this particular case is shown in Fig. 2.14(a), while the equivalent for two harmonics is plotted in Fig. 2.14(b). The two figures compare the lift coefficient, c_l, for one full cycle. Each harmonic balance time instance is represented as an orange circle. The HB reconstruction is plotted as a dashed orange line.

The one-harmonic simulation overestimates the amplitude of the lift coefficient. While still overestimating this amplitude, the two-harmonic simulation provides a better match with the time-domain prediction. However, the increase in the number of harmonics results in a higher computational cost.

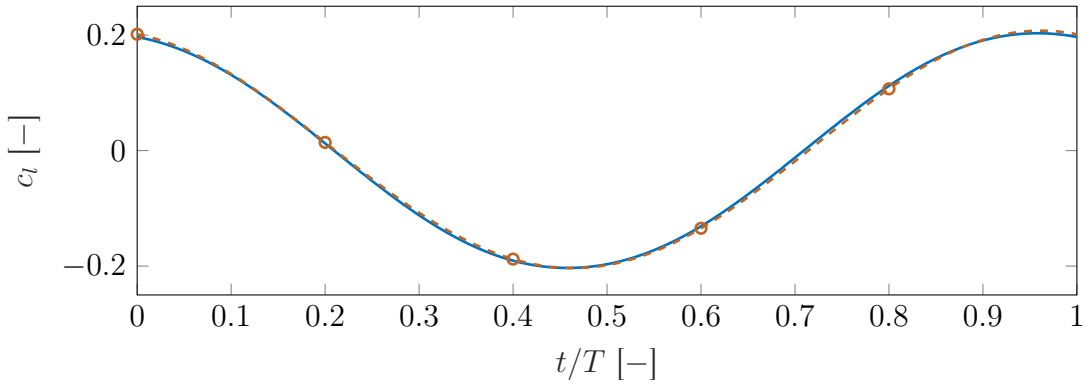
	U_∞ [$\mathrm{m\,s^{-1}}$]	$\tilde{U}$ [$-$]	ρ_∞ [$\mathrm{kg\,m^{-3}}$]
Case 0	268.2	2.9012	0.8544
Case 1	272.2	2.9445	0.8544
Case 2	276.2	2.9878	0.8544
Case 3	280.2	3.0311	0.8544
Case 5	288.3	3.1177	0.8544
Case 6	296.3	3.2043	0.8544
Case 7	304.3	3.2909	0.8544
Case 8	312.3	3.3775	0.8544
Case 9	320.3	3.4641	0.8544
Case 10	340.3	3.6806	0.8544
Case 11	360.3	3.8971	0.8544

Table 2.6: Freestream conditions of various transonic NACA 64A010 test cases

For the time-marching calculation, a second-order dual time stepping method was used. The CFL number was set to 10 and allowed to adapt. Different time steps were tested in order to check the convergence behaviour. The final chosen time step was $\Delta t = 0.001\,\mathrm{s}$. It resulted in approximately 100 time steps per cycle. Halving the value of the time step resulted in a $\sim 0.8\%$ increase in amplitude, while doubling it led to a $\sim 3.5\%$ decrease. Each time step required for the most part 2 FSI iterations. The average number of FSI iterations was 1.97. Each fluid calculation required 30 to 100 pseudo-time iterations to converge. The initial conditions imposed to the



(a) One harmonic



(b) Two harmonics

Figure 2.14: Comparison of lift coefficient for one cycle predicted by time-marching and harmonic balance methods for Case 1

structural solver were the pitch and plunge obtained for the first time instance of the harmonic balance calculation in Case 0. This instance represents the maximum pitch. The freestream conditions for this case are shown in Table 2.6. The simulation was run for 20 s, around 200 cycles.

The harmonic balance calculation used the results from Case 0 in order to start the solution. The CFL number was set to 4 in order to ensure convergence of the fluid solver. The maximum number of fluid iterations was set to 20 000 per FSI iteration.

Similar convergence criteria were used for the time-marching and harmonic balance methods. For the fluid part, the simulation stopped when the density residual was below a given value. The FSI coupler either moved to the following time iteration or ended the calculation once the norm of the displacement residual was

below a set tolerance. The displacement residual tolerance and minimum density residual values were the same for both methods. The harmonic balance solver used a tolerance for the frequency of $\Delta\omega = 0.002\,\mathrm{rad\,s^{-1}}$.

The simulations were run on a computer with a 3.5 GHz Quad-Core Intel Core i7 processor and 8 GB of RAM. Table 2.7 compares the CPU time cost of the one-harmonic HB approach (t_{HB}) and the time-marching approach with the selected time step (t_{TM}). The harmonic balance technique is more than ten times less computationally expensive than the time-marching approach.

The computational cost is separated into several steps. Since the matching mesh interpolation method is used, the mesh mapping is simple. Furthermore, this mapping is performed only once, at the beginning of each simulation. In both cases it represents a negligible portion of the total time. Mesh deformation is performed once per FSI iteration. The time-marching solver uses strong coupling, which results in several FSI iterations per time step and, thus, several mesh deformations per time step. The harmonic balance approach performs $2 \cdot N_h + 1$ mesh deformations per FSI iteration, but it requires only 35 FSI iterations in total to converge. This explains the much lower computational cost. The communication of displacements from the structural solver to the fluid solver and of loads from the fluid solver to the structural solver behaves similarly. Both approaches' computational cost is dominated by the fluid solver. The harmonic balance approach for the fluid is ~ 9.8 times faster than the time marching solution. For both approaches the solid solver is the second least computationally expensive component of the total cost. The harmonic balance solid solver is more than two orders of magnitude faster than the time marching one because the solid harmonic solution is evaluated fewer times than its time-marching counterpart.

Step	t_{HB} [s]	t_{TM} [s]
Mesh mapping	0.0	0.0
Mesh deformation	41.9	19 530.2
Communication	0.3	153.4
Fluid solver	19 909.6	194 308.0
Solid solver	0.1	33.3
Total	19 955.0	216 855.7

Table 2.7: Comparison of computational cost between time-marching and harmonic balance methods for Case 1

A comparison of the CPU time and first harmonic pitching amplitude error for time-marching and harmonic balance methods is shown in Fig. 2.15. For the time-

marching approach, it shows in blue four computations with different time steps ($\Delta t = 0.004\,\mathrm{s}$, $0.002\,\mathrm{s}$, $0.001\,\mathrm{s}$ and $0.0005\,\mathrm{s}$). Then, the results obtained using the harmonic balance technique are plotted in orange for one, two and three harmonics. The error is defined as the difference in amplitude of the first harmonic:

$$\varepsilon_\alpha = \left| 1 - \frac{\alpha_1}{\alpha_{1,ref}} \right|, \tag{2.76}$$

where $\alpha_{1,ref}$ is a reference value obtained from the time-marching simulation with $\Delta t = 0.0005\,\mathrm{s}$. Note that all results obtained using the time-marching technique underestimate the amplitude and all those obtained using the harmonic balance overestimate the amplitude.

The error in the amplitude of the first harmonic has a very strong dependence on the time step. The HB curve is below the time-marching one, with higher accuracy at a lower computational cost. However, it appears that its slope is steeper. Notably, the amplitude predicted using two and three harmonics is quite similar. It would be expected that the simulation using three harmonics results in a better match with the reference time-marching simulation. However, while the two harmonics case has an error of $\varepsilon_\alpha \simeq 0.3\%$, the three harmonics computation's error is slightly higher at $\varepsilon_\alpha \simeq 0.5\%$. These errors are very small. It is possible that the reference value, $\alpha_{1,ref}$, is not converged enough. Then, using a smaller time step in the time-marching simulation would result in a larger amplitude. Another possibility is that due to aliasing, higher-order harmonics reduce the amplitude of the first harmonic in the two-harmonics calculation, which leads to an apparently better match for α_1.

There are several reasons for the computational cost increasing with the number of harmonics. At equal number of fluid iterations, increasing the number of harmonics results in a corresponding increase of the number of required calculations. The number of pseudo-steady computations for the time-domain harmonic balance increases linearly with the number of time instances. The calculation of the source term representing the time derivative, on the other hand, increases quadratically. Furthermore, the CFL number had to be decreased for the higher harmonics to 1.8 for the two-harmonics case and to 1.3 for the three-harmonics computation. This led to an increase in the number of fluid iterations required for convergence.

Finally, the number of FSI iterations was more or less constant with the number of harmonics. Compared to the 35 required by the one-harmonic technique, the two- and three-harmonics simulations needed 34 and 38 iterations for FSI convergence, respectively. Therefore, they did not have a large effect on the computational cost.

In conclusion, the fluid solver dominates the computational cost of this problem. The harmonic balance approach is significantly faster thanks to a reduction of the cost of the fluid solver and that of the mesh deformation. If using one harmonic, the

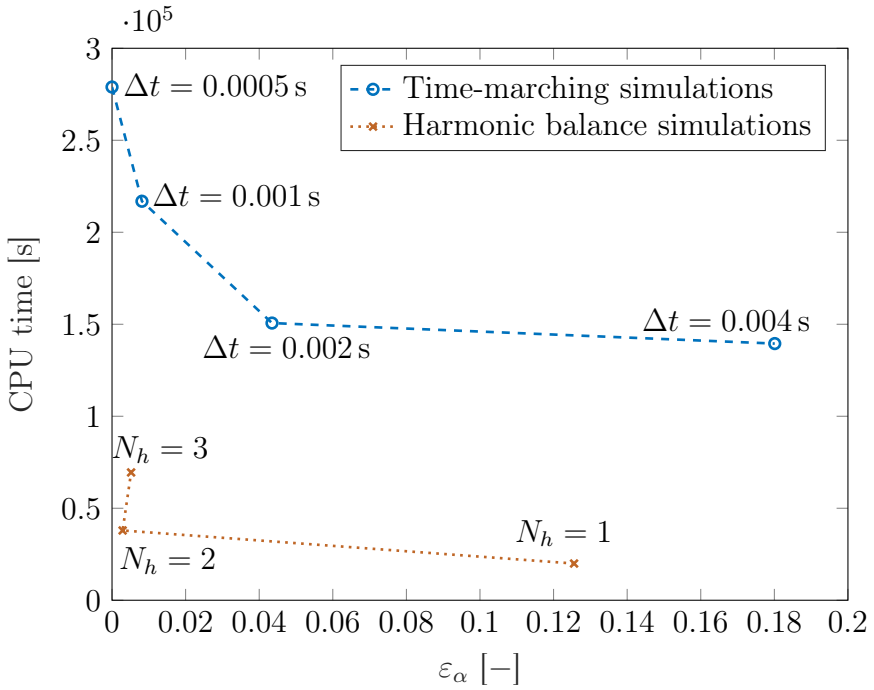


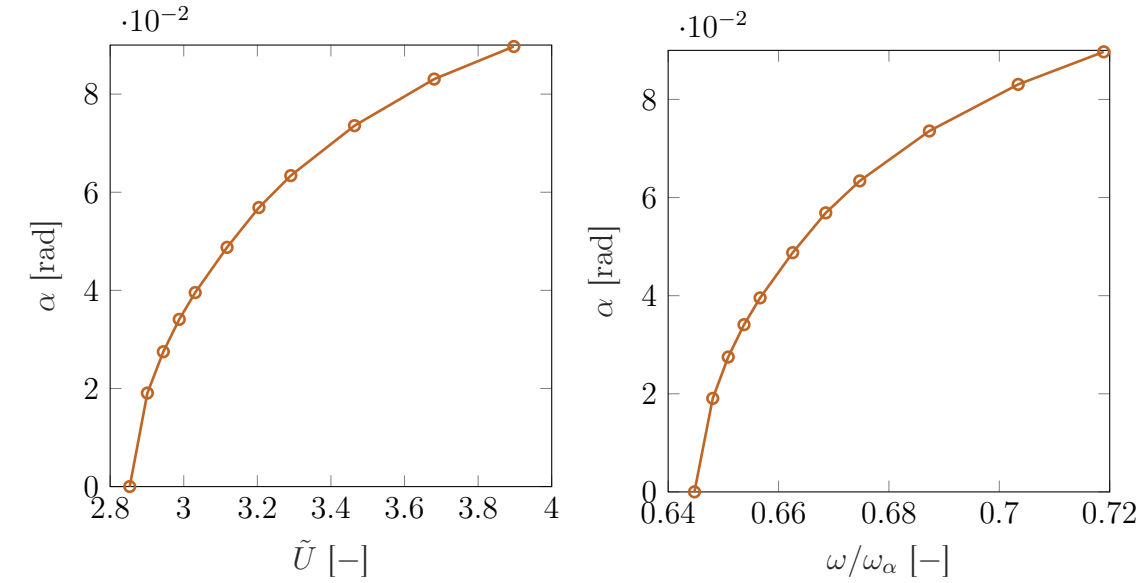
Figure 2.15: Comparison of CPU time and pitching amplitude error

computational cost savings are much more important. The time-marching solution depends on the time step used, which may be because small changes in the pressure distribution lead to large changes in the amplitude close to the flutter point.

2.8.5 Bifurcation diagram

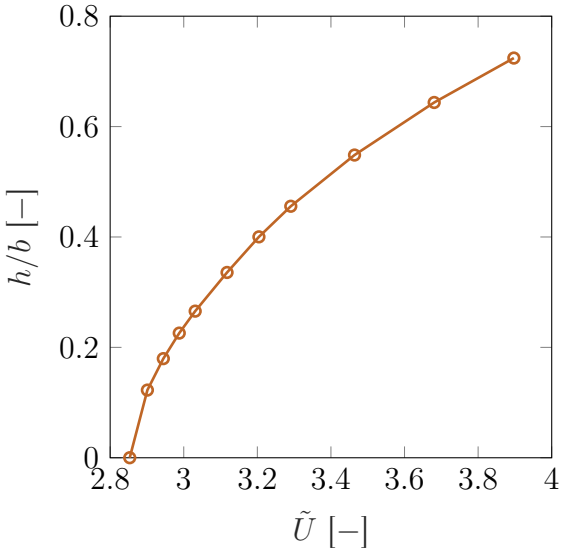
Figure 2.15 shows that using two harmonics would improve the results, but it would approximately double the computational cost. However, the error in amplitude of the single harmonic approach is acceptable, being small in magnitude and more conservative than the two-harmonics or time-marching solutions. Therefore, the other limit-cycle oscillation simulations to plot the bifurcation diagram are carried out using one harmonic. The results of a velocity sweep with constant freestream Mach number $\mathrm{M}_\infty = 0.8$ are shown in Fig. 2.16. The freestream conditions used for the sweep are shown in Table 2.6.

Figure 2.17 compares results obtained using the present method, in orange; those presented by Simiriotis and Palacios [62] using a constant-amplitude harmonic balance approach, in dashed blue; and those shown by Li and Ekici [59] in dotted purple. As in the present study, Simiriotis and Palacios used one harmonic while Li

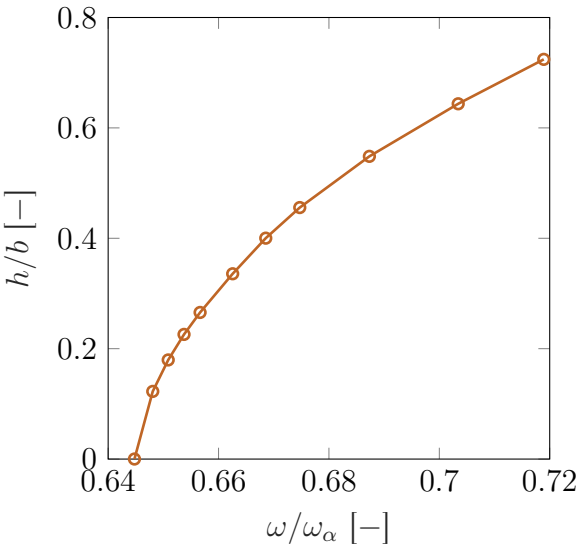


(a) Pitch versus reduced velocity

(b) Pitch versus frequency ratio



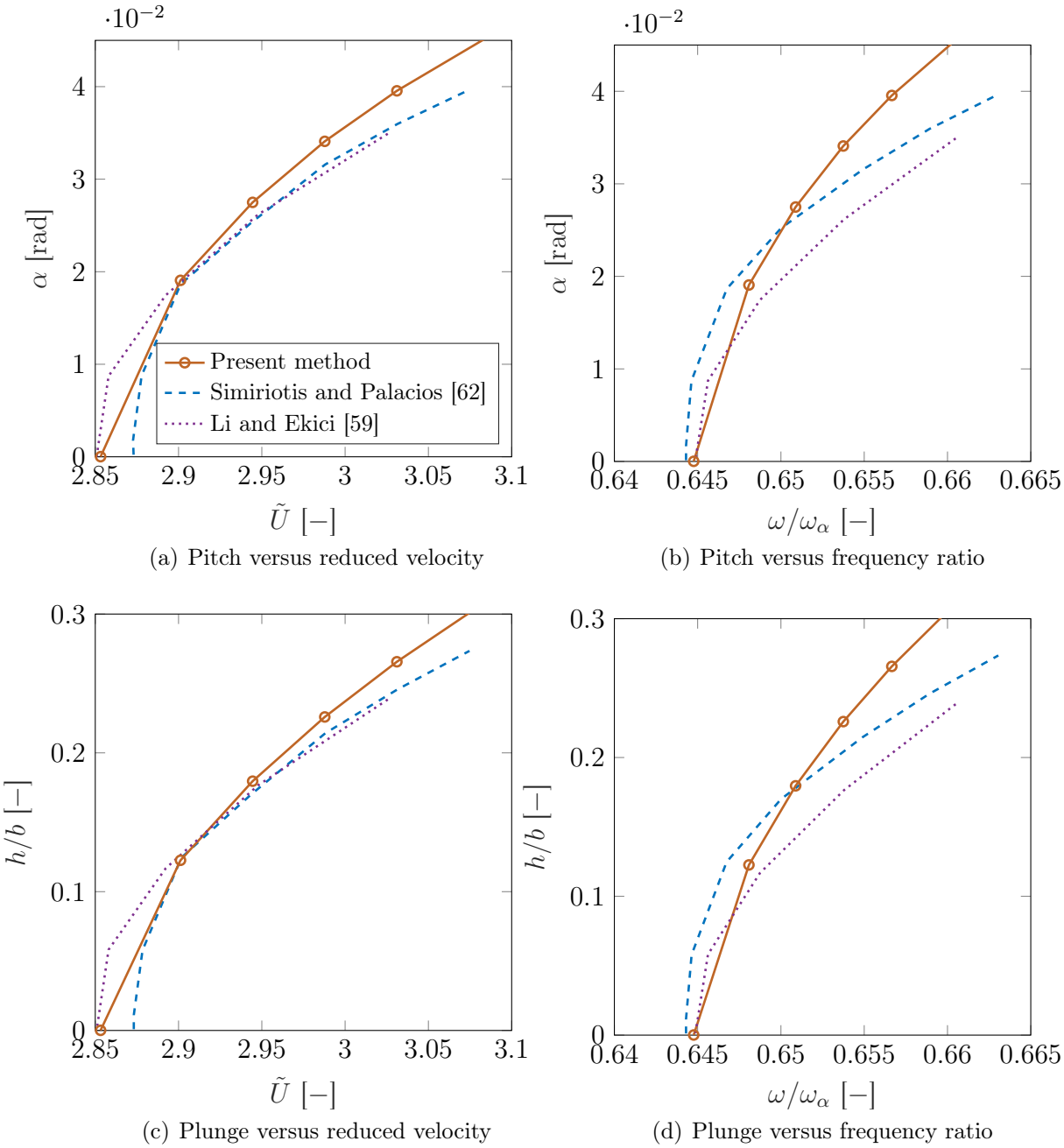
(c) Plunge versus reduced velocity



(d) Plunge versus frequency ratio

Figure 2.16: Limit-cycle oscillation amplitudes obtained by means of the present frequency-varying harmonic balance algorithm

and Ekici used three. Both groups studied the lower amplitude range of the results shown in Fig. 2.16.



(a) Pitch versus reduced velocity

(b) Pitch versus frequency ratio

(c) Plunge versus reduced velocity

(d) Plunge versus frequency ratio

Figure 2.17: Comparison of movement amplitudes between harmonic balance methods

The quantities compared are the movement amplitudes, α and h/b, and the ratio of the frequency of motion and the uncoupled wind-off natural frequency of the pitching degree of freedom, ω/ω_α. The trends are similar: the amplitude increases with higher freestream velocity, and so does the frequency. However, the present method predicts a larger amplitude and lower frequency than the other two. As expected for a Hopf bifurcation, close to the flutter point the gradient of the amplitude, $\mathrm{d}\alpha/\mathrm{d}\tilde{U}$, is very large. This may explain the large changes in amplitude with the number of harmonics and time step observed in the previous section: small differences in the pressure distribution will have a strong impact on the amplitude.

2.8.6 High-amplitude case

In the previous section, limit-cycle oscillations were predicted over a range of airspeeds. Their amplitude increases with airspeed; the higher the amplitude the more important the nonlinear behaviour of the flow. In this section, a high-amplitude LCO is studied in further detail.

This case's parameters are shown in Table 2.6 as Case 9. The pitching amplitude obtained by the present method is $\alpha_1 = 0.0736\,\mathrm{rad} \simeq 4.22°$, while the plunging amplitude is $h_1/b = 0.531$. Figure 2.18 compares the mean pressure coefficient distribution for this case, in orange, and the steady pressure coefficient for $\alpha = 0$, in blue. Since the HB mean flow is asymmetric, the upper side is represented as a solid line while the lower side is a dashed line. The pressure coefficient is defined as $C_p = \frac{p-p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2}$. The mean pressure is obtained by averaging the pressure over the three time instances at each node on the aerofoil surface. It is equal to the zeroth harmonic of the pressure obtained by using the discrete Fourier transform matrix shown in Eq. (2.53).

There are two main differences between the steady and harmonic balance mean flows: the shock is further downstream in the unsteady case and the steady C_p is symmetric while the mean C_p is not. The reason behind this asymmetry is the interaction between the shock and the time discretisation. The steady flow has one shock on each side of the aerofoil. In the unsteady case, the two shocks are still present but they move upstream and downstream and their strength changes depending on the phase of the oscillation. At high enough motion amplitudes, the shocks can completely disappear over part of the cycle. Since the number of time instances at which the flow is calculated using the time-domain harmonic balance method is low, estimating the mean flow from these calculations results in an asymmetry.

The flowfield around the aerofoil obtained by means of the HB method is shown in Fig. 2.19. The figure plots the Mach number distribution at three phases of the

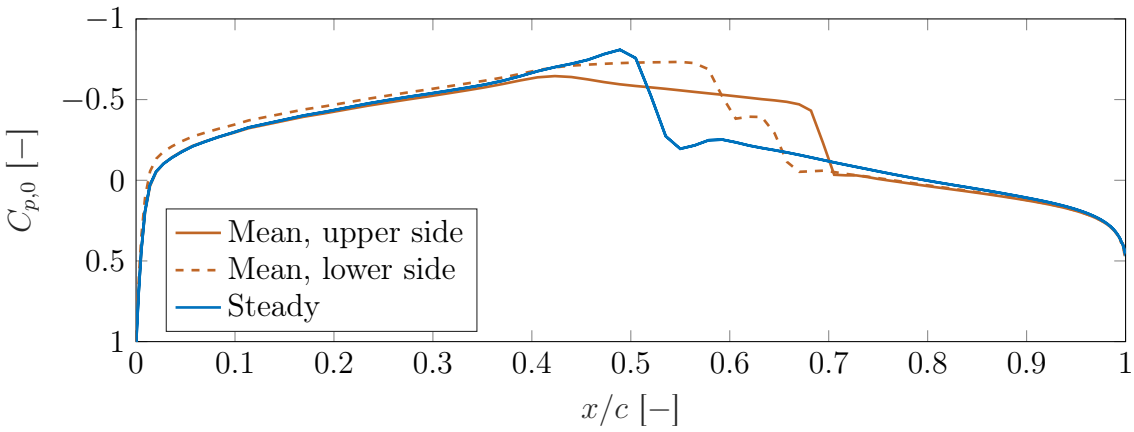
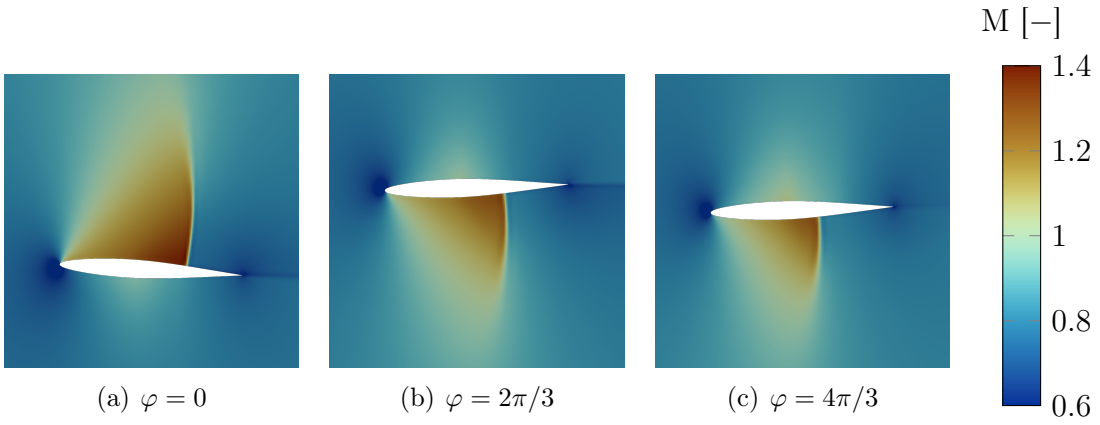


Figure 2.18: Mean pressure coefficient distribution for Case 9 compared to steady pressure coefficient distribution



(a) $\varphi = 0$ (b) $\varphi = 2\pi/3$ (c) $\varphi = 4\pi/3$

Figure 2.19: Mach number around the aerofoil for each time instance in Case 9

cycle, corresponding to the time instances used in the TDHB flow calculations. At $\varphi = \omega \cdot t = 0$, shown in Fig. 2.19(a), the shock appears only on the upper surface. This time instance corresponds to the maximum value of the pitch. Meanwhile, in Fig. 2.19(b) and Fig. 2.19(c), the pitch is negative. This leads to the shock appearing only on the lower surface in both time instances, which in turn results in the asymmetric C_p distribution in Fig. 2.18.

The chordwise axis is an axis of symmetry of the studied case. Thus, symmetric results should be expected *a priori*. However, as shown in Fig. 2.18, this is not the case.

The waveforms used to study how this asymmetric behaviour changes with phase

Waveform 1	$\alpha(t) = \alpha_0 + \alpha_{1,c}\cos(\omega t)$
Waveform 2	$\alpha(t) = \alpha_0 + \alpha_{1,s}\sin(\omega t)$
Waveform 3	$\alpha(t) = \alpha_0 + \alpha_{1,c}\cos(\omega t) + \alpha_{2,c}\cos(2\omega t) + \alpha_{2,s}\sin(2\omega t)$
Waveform 4	$\alpha(t) = \alpha_0 + \alpha_{1,c}\cos(\omega t) + \alpha_{2,c}\cos(2\omega t) + \alpha_{2,s}\sin(2\omega t) + \alpha_{3,c}\cos(3\omega t) + \alpha_{3,s}\sin(3\omega t)$

Table 2.8: Waveforms used to study the asymmetric flow behaviour

and the number of harmonics are shown in Table 2.8. They are two one-harmonic setups with different phases (waveforms 1 and 2), one two-harmonics waveform (waveform 3) and one with three harmonics (waveform 4). Waveform 1 was used with positive and negative values of $\alpha_{1,c}$, finding that its sign only affected the sign of the results, not their magnitude. Therefore, only the values corresponding to a positive $\alpha_{1,c}$ are presented.

Table 2.9 compares the mean pitch and lift coefficient for these four waveforms. These values should be 0 because of the symmetry of the problem. However, they are small but non-zero. This is more pronounced in the case of waveform 1, which is the one used in order to obtain the bifurcation diagrams.

Waveform	α_0 [rad]	$c_{l,0}$ [−]
Waveform 1	-2.60×10^{-3}	-0.0261
Waveform 2	5.26×10^{-4}	-0.0062
Waveform 3	-1.36×10^{-4}	-0.0009
Waveform 4	-8.39×10^{-5}	-0.0008

Table 2.9: Comparison of mean quantities for four different waveforms

Figure 2.20 shows the mean, real and imaginary parts of the pressure coefficient scaled by the amplitude of the first harmonic of the pitching motion for the one-harmonic studies. The harmonic balance results obtained using waveform 1 are plotted in solid orange, while those obtained using waveform 2 are in dash-dotted green. The time marching approach is in dashed blue. Figure 2.21 compares the mean and the first harmonic of the pressure distributions obtained from HB simulations using one, two and three harmonics. Like in Fig. 2.20, the single-harmonic approach using waveform 1 is in solid orange and the time marching results are in dashed blue. Waveform 3, which uses two harmonics, is in dash-dotted green and waveform 4, with three harmonics, is in dotted purple.

The five setups' behaviours are similar in the first 40% of the chord. In this region they match the trends of the time-marching approach for the mean flow and the real and imaginary parts of the first harmonic. However, there is a clear difference between waveform 1 and the others: the mean flow in this part of the aerofoil is not symmetric. This is due to the mean pitch of $\alpha_0 = -0.0026\,\text{rad} \simeq -0.15°$. For all other waveforms the mean pitch is significantly smaller, as shown in Table 2.9.

Around the shock, the time-marching results show a smooth transition of all three components studied, while the harmonic balance shows jumps. These jumps appear at the location of the shock, or shocks, in each time instance. As the number of harmonics increases, the harmonic balance signal approaches the time-marching response. This can be seen for the real part in Fig. 2.21(b). Waveform 4, in purple, shows a smoother behaviour than the other results. This waveform used three harmonics and required seven time instances.

The imaginary part is smaller in magnitude than the real part. The time-marching case predicts a symmetric reduction of its value around the shockwave, with a minimum at $x/c \simeq 65\%$. The harmonic balance method, however, does not. This is especially marked for waveforms 1 and 2, which use one harmonic. The results obtained using these two waveforms are shown in Fig. 2.20(c). The value of $\mathcal{I}\left(C_{p,1}/\alpha_1\right)$ for the original approach, in orange, goes from around 0.3 to around 8.5 on the upper side. However, it does not change significantly on the lower side. Waveform 2, in green, shows an increase in the value of $\mathcal{I}\left(C_{p,1}/\alpha_1\right)$ on the upper side as well. On the lower side, the value changes sign contrary to the time-marching behaviour. On both sides there is a plateau of high $\mathcal{I}\left(C_{p,1}/\alpha_1\right)$ values.

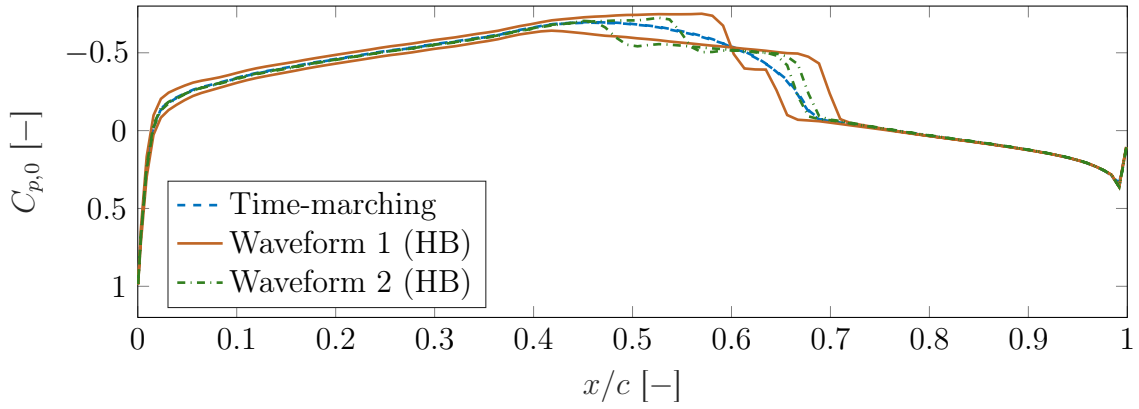
Downstream of the shock, the values become similar again. There is a change in the sign of the real part of the response, shown in Figs. 2.20(b) and 2.21(b). This change is recovered by the harmonic balance method for all cases studied as well.

A Fourier analysis of the time-marching simulation of the problem is shown as a blue curve in Fig. 2.22. There are prominent peaks in odd multiples of the fundamental frequency up to the third harmonic. However, its even multiples are absent from the signal. The figure also includes the amplitudes obtained using waveforms 1 (orange stem plot), 3 (greenstem plot) and 4 (purple stem plot).

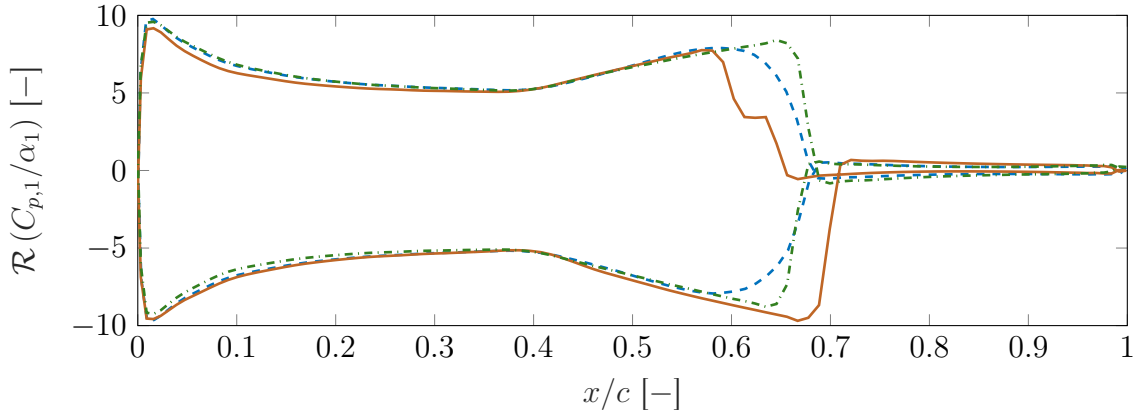
The case is symmetric with respect to the chordwise axis. Express the structural displacements as a function of time:

$$\alpha(t) = \alpha_0 + \sum_{n=1}^{N_h} \alpha_{n,c} \cdot \cos\left(n\omega t\right) + \alpha_{n,s} \cdot \sin\left(n\omega t\right). \tag{2.77}$$

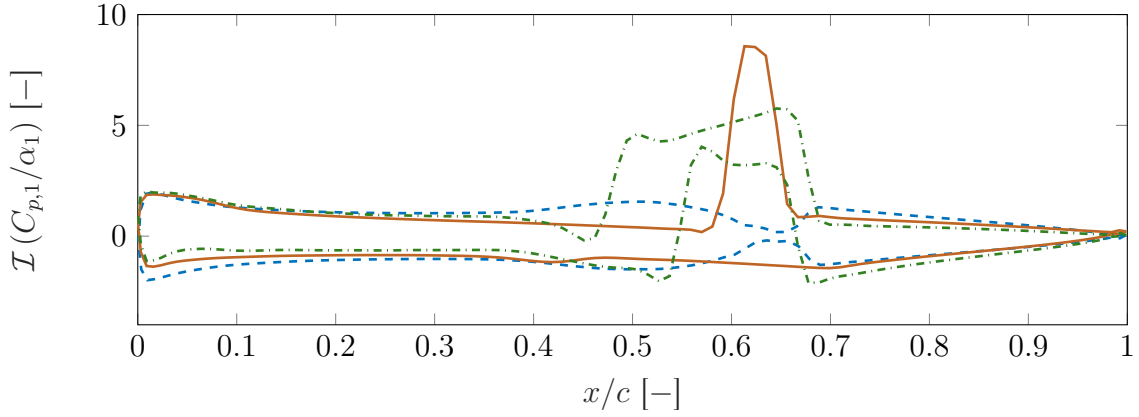
Since the setup is symmetric, $-\alpha(t)$ is a solution of the problem as well. Evaluating



(a) Mean

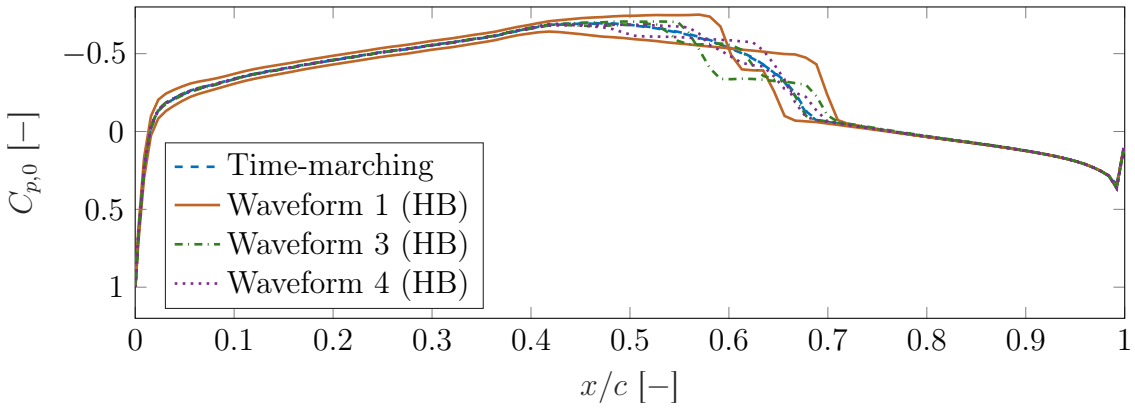


(b) Real part

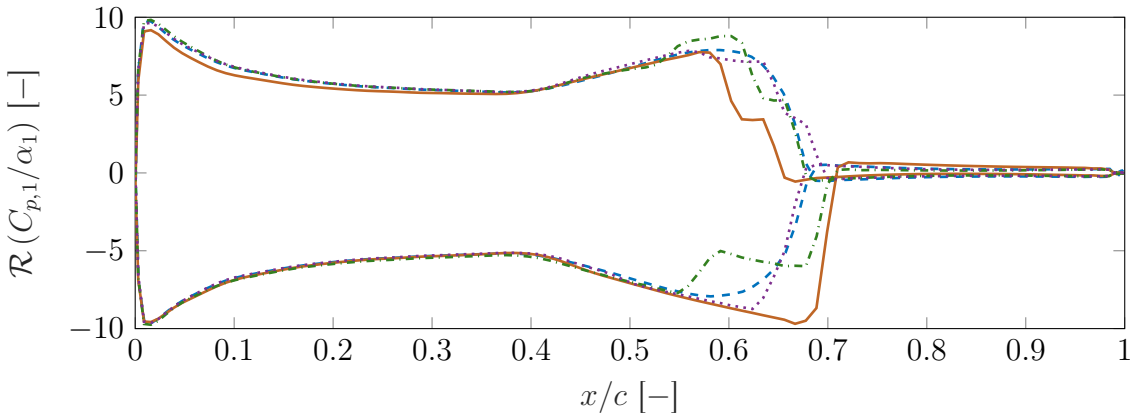


(c) Imaginary part

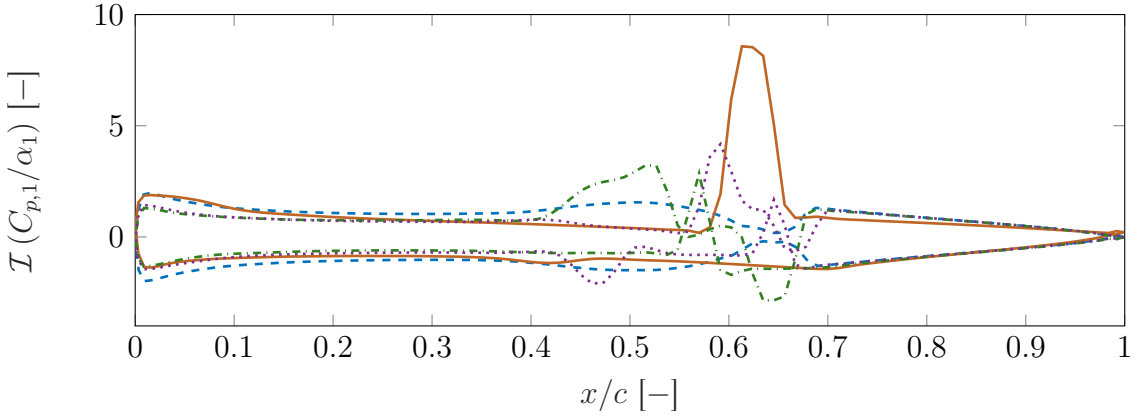
Figure 2.20: Mean, real and imaginary parts of the pressure coefficient distribution for Case 9



(a) Mean



(b) Real part



(c) Imaginary part

Figure 2.21: Mean, real and imaginary parts of the pressure coefficient distribution for Case 9

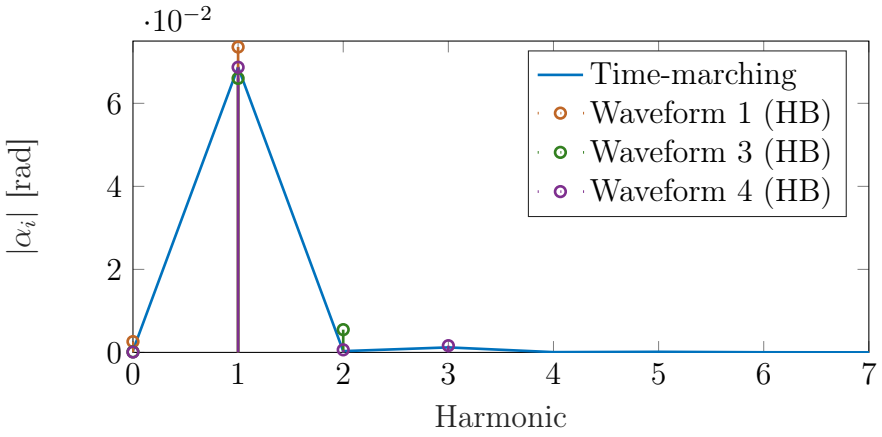


Figure 2.22: Harmonic amplitudes of the pitch using time-marching and harmonic balance methods

the expression in Eq. (2.77) at $t = t + T/2$:

$$\begin{aligned} \alpha\left(t + T/2\right) &= \alpha_0 + \sum_{n=1}^{N_h} \alpha_{n,c} \cdot \cos\left(n\omega t + n\pi\right) + \alpha_{n,s} \cdot \sin\left(n\omega t + n\pi\right) = \\ &= \alpha_0 + \sum_{n=1}^{N_h} \left(-1\right)^n \left[\alpha_{n,c} \cdot \cos\left(n\omega t\right) + \alpha_{n,s} \cdot \sin\left(n\omega t\right)\right], \end{aligned} \tag{2.78}$$

which is equal to $-\alpha(t)$ if and only if $\alpha_{2n,c} = \alpha_{2n,s} = 0 \quad \forall n \in \mathbb{N}$. Otherwise, there would be at least two possible solutions to the problem. Multiple solutions are not observed in the time-marching simulations and the second harmonic's amplitude is significantly reduced using three harmonics (waveform 4), as shown in Fig. 2.22. It goes from 5.5×10^{-3} rad $\simeq 0.31°$ for waveform 3 to 6.2×10^{-4} rad $\simeq 0.036°$ for waveform 4. Therefore, the improvement in behaviour observed using two harmonics (waveform 3) with respect to one harmonic (waveform 1) is probably caused by the better time resolution of the shock and not due to the effect of even harmonics.

2.8.7 FSI convergence

The previous sections have shown the results obtained by the coupled FSI harmonic balance technique with unknown frequency for a transonic LCO. The number of FSI iterations it takes to achieve convergence is important in order to evaluate the efficiency of the method. There are two main outputs: the amplitude of the limit-cycle oscillation and its corresponding frequency. Both the frequency and the nodal displacements, which are a function of the movement amplitude, appear as

convergence criteria in Fig. 2.5. The present section describes the convergence of one low-amplitude case and one high-amplitude case.

Low-amplitude case

In low-amplitude cases the rate of convergence of a problem depends on the total damping. Close to the flutter point, the damping is negligible. In time-marching simulations, this leads to a very high number of cycles required in order to reach the final, converged LCO amplitude. In the harmonic balance approach, at each FSI iteration the structural displacements change by a small amount. Many FSI iterations are then needed to achieve convergence.

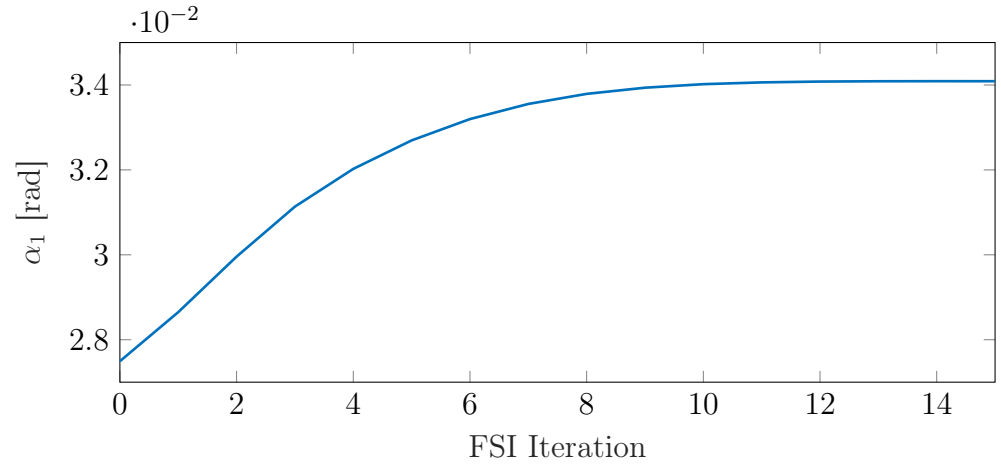
One example of a low-amplitude result is Case 2 in Table 2.6. This case was started with the structural displacements and frequency obtained from Case 1. The initial guess for the pitching amplitude was $0.0275\,\text{rad} \simeq 1.58°$. Figure 2.23 shows the evolution of the pitching amplitude (Fig. 2.23(a)) and the frequency (Fig. 2.23(b)). Low-amplitude cases converged in 10 to 40 iterations, with Case 0 requiring the largest number of iterations.

The method used to find the frequency of the coupled LCO is one of the main differences with respect to previous work. Therefore, it is important to compare its convergence with previous results. Li and Ekici showed the evolution of the LCO frequency and the structural residuals with each FSI iteration for one case. As in the present method, they used a constant freestream conditions approach. However, they used three harmonics instead of one [59]. Simiriotis and Palacios, who used one harmonic, did not include the convergence history of any case [62].

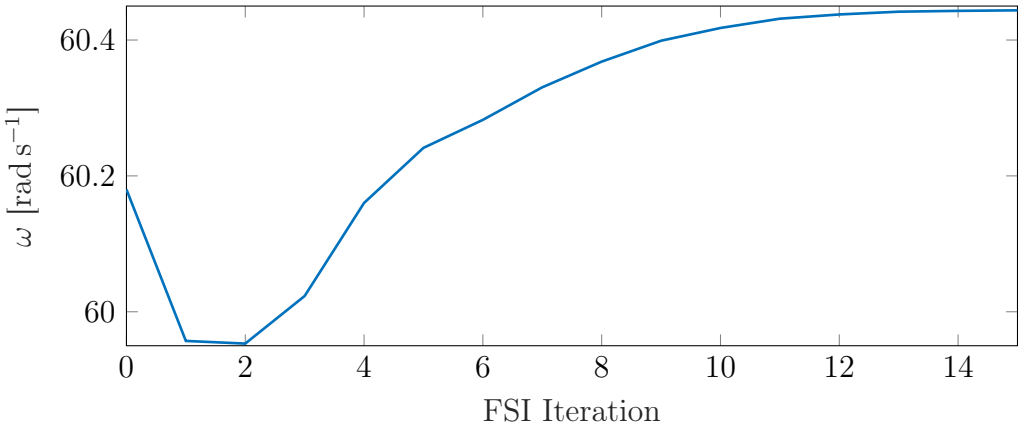
Li and Ekici compared their method to a constant amplitude approach, from which they obtained the freestream conditions for an amplitude of $2° \simeq 0.0350\,\text{rad}$. They used an amplitude of $0.0367\,\text{rad}$ and the flutter frequency as the starting point of the simulation [59].

In order to compare the two approaches, a case computed using the present method was chosen. The setup with the closest pitching amplitude to the case described by Li and Ekici was Case 2. It resulted in a pitching amplitude of $0.0341\,\text{rad} \simeq 1.95°$. The convergence of Case 2 as a function of the FSI iteration was shown in Fig. 2.23.

Figure 2.24 shows the ratio between the final frequency and that at each FSI iteration for the present method (in orange) and the method using the L^2 norm of the residual (in blue) by Li and Ekici [59]. The difference in number of FSI iterations needed to converge is of an order of magnitude. However, it should be noted that Li and Ekici used fewer fluid iterations in each FSI iteration. They used 100 while the present method used around 5000.



(a) Pitching amplitude



(b) Frequency

Figure 2.23: Convergence of results with FSI iterations for the present method for Case 2

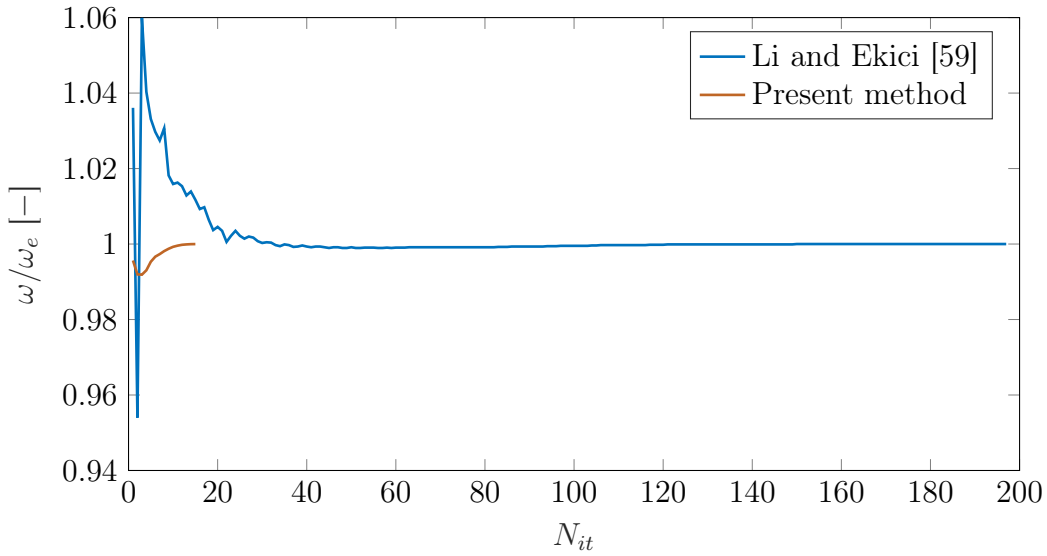


Figure 2.24: Frequency convergence as a function of FSI iterations

High-amplitude case

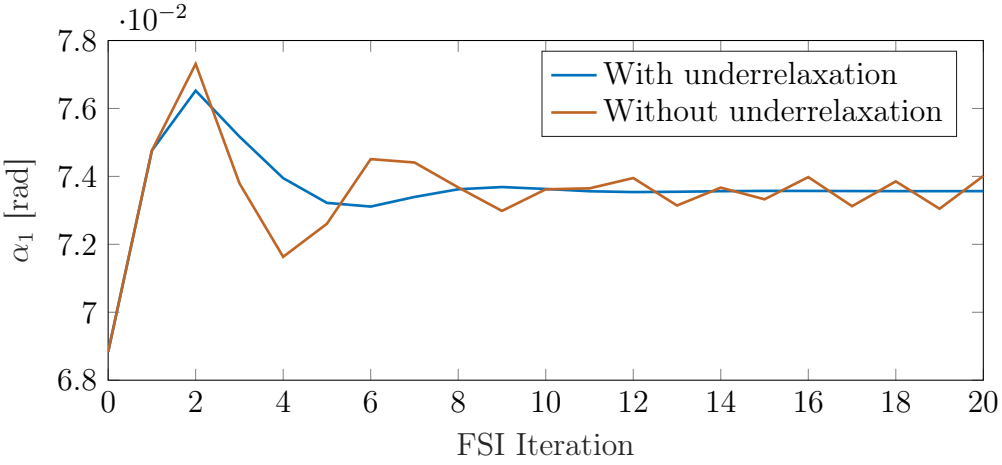
Higher amplitude cases presented a different set of challenges. When using the method without underrelaxation some oscillations appeared between FSI iterations, with convergence stalling and the solution eventually diverging. Therefore, the value of the relaxation parameter had to be lowered.

Figure 2.25 compares the convergence of Case 9 as described in Table 2.6 with underrelaxation, in blue, and without underrelaxation, in orange. For the case with underrelaxation, the relaxation parameter was set to $\Omega = 0.8$. After 10 to 15 iterations, the oscillations in the results of the simulation without underrelaxation start to grow, while in the case with underrelaxation they damp out very quickly.

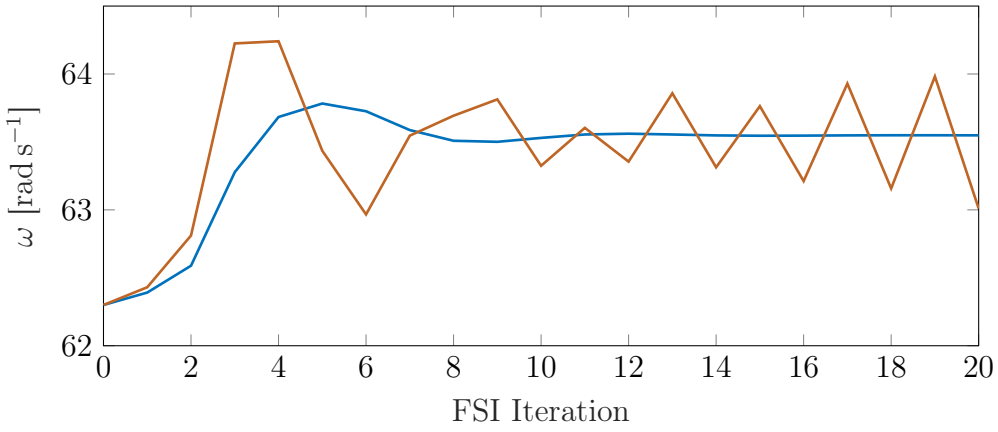
2.9 Summary

The present chapter has demonstrated a novel partitioned harmonic balance method for the study of time-periodic fluid-structure interaction problems with unknown frequency. It differs from previous partitioned approaches in the frequency iteration technique used. The method reduces the computational cost in CPU time of the problem compared to time-marching simulations while increasing RAM usage. The fluid sub-problem dominates the computational cost.

The harmonic balance technique has been applied to a transonic limit-cycle



(a) Pitching amplitude



(b) Frequency

Figure 2.25: Convergence of results with FSI iterations for the present method for Case 9

oscillation test case. If only one harmonic is used, the method overestimates the LCO amplitude compared to a time-marching approach in the cases studied. Time-marching simulations are highly sensitive to the time step used.

The presence of a moving shock that disappears during parts of the cycle leads to some differences in the flow between the time-marching and harmonic balance techniques. The moving and disappearing shock results in the harmonic balance method predicting a slightly asymmetric behaviour of the flow. When the number of harmonics is increased, these differences are reduced. Three harmonics are sufficient to accurately describe the flow, but this results in an increased computational cost compared to one harmonic. One way to increase the accuracy at a reduced computational cost could be by means of high-dimensional harmonic balance approaches that use more time instances than harmonics, such as the fast Fourier transform-based method of Ling and Wu [85].

Convergence of the approach proposed in the present chapter depends on the amplitude of the LCO. At low amplitudes many FSI iterations may be required. At high amplitudes underrelaxation is needed to reach convergence. Despite these limitations, the proposed harmonic balance method leads to a reduced number of FSI iterations for convergence compared to other HB methods.

Chapter 3

Adjoint method for the optimisation of steady fluid-structure interaction problems

The optimisation of structures immersed in flows is becoming an increasingly important design objective. Adjoint methods help to speed up optimisation in the early design stage by reducing the number of calculations required to study the effect of many design variables. They have been widely applied in many different fields, including aerodynamics and structural mechanics.

In this chapter, a description of different optimisation algorithms and their classification is presented first. Then, some variational techniques to obtain the gradients are described. The adjoint method, with a focus on steady problems, is presented in contrast to variational approaches. Afterwards, the difference between continuous and discrete adjoint techniques is explained.

The adjoint method requires one computation per objective function. Therefore, two approaches to reduce the number of objective functions are shown: penalty methods and constraint aggregation. Subsequently, the adjoint method is applied to partitioned fluid-structure interaction (FSI) problems. Finally, the coupled adjoint FSI algorithm is applied to a simple problem using two different structural solvers in order to verify its implementation.

3.1 Optimisation methods

An optimisation algorithm minimises or maximises[1] an objective function, J, that depends on a series of design variables $\boldsymbol{\xi}$ through the state variables $\mathbf{U}$, subject to

[1]Maximising a function f is equivalent to minimising its negative $-f$

a set of constraints, $\mathbf{K}$. The optimisation problem can be written as

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & J(\mathbf{U}(\boldsymbol{\xi})) \\ \text{subject to} \quad & \mathbf{R}(\mathbf{U}(\boldsymbol{\xi})) = 0 \\ & \mathbf{K}(\boldsymbol{\xi}, \mathbf{U}(\boldsymbol{\xi})) = 0 \end{aligned} \quad , \tag{3.1}$$

where $\mathbf{R}$ is the residual of the physical problem.

There are two main families of algorithms to solve the optimisation problem: gradient-free and gradient-based. The first family allows the exploration of the whole design space, while gradient-based methods are able to find local optima. When using gradients, it is generally not possible to ensure that the optimum found is global. Meanwhile, gradient-free techniques, while they can explore a large section of the design space, they cannot converge to optima. In many cases within the aerospace industry, current expertise allows designers to start from a good design. In such cases, a gradient-based approach may be better suited. Both kinds of methods can be used in the same process: first a gradient-free algorithm can find an initial solution close enough to the global optimum and then a gradient-based algorithm can converge towards this optimal solution.

3.1.1 Gradient-free methods

Gradient-free methods are a natural choice if the objective function has many local optima, if it is not smooth, or if its gradients are not available or difficult to compute. Many gradient-free optimisation algorithms are inspired by other disciplines, such as metallurgy or biology.

Within biologically-inspired methods, one of the most widely applied families is that of genetic algorithms. They broadly involve generating a starting set of designs. Each design's parameters make up a chromosome. The designs are evaluated and then a process of selection occurs. If the objective function is to be minimised the designs with the lowest value are kept in the population, while the others are eliminated. Of the remaining designs, some mate, generating new chromosomes that are a mixture of the original ones. In order to fully explore the design space, these chromosomes then are modified. This process is called mutation. Then, the new value of the design parameters is decoded from the chromosomes. The full process of selection, mating and mutation is repeated until enough iterations have occurred [86].

These kinds of optimisation methods have been applied to aerodynamic and aeroelastic problems. Lyu et al. minimised the drag coefficient at a constant lift of an aircraft wing by modifying its twist. They found that gradient-free methods

required two to three orders of magnitude more computations than the gradient-based methods they used [87]. For aeroelasticity, Khodaparast et al. applied both genetic algorithms and the bacterial foraging optimisation algorithm [89] to an aircraft subjected to gust loads [88].

3.1.2 Gradient-based methods

Gradient-based algorithms are useful in cases with smooth objective functions and in which the starting point of the design is good. While gradient-free methods only need the value of the objective function, gradient-based techniques require obtaining the gradients of the objective function or functions with respect to the design variables. Generally, this gradient vector is used in order to determine a direction of descent, $\mathbf{s}$, such that

$$\mathbf{s}^T \frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}} < 0. \tag{3.2}$$

Then, a step length, $\Delta\xi$, multiplies this direction so that for iteration k the value of the design variables is

$$\boldsymbol{\xi}^k = \boldsymbol{\xi}^{k-1} + \Delta\xi^k \cdot \mathbf{s}^k. \tag{3.3}$$

This step length can either be predetermined or obtained by means of a line search algorithm. These algorithms obtain a value of $\Delta\xi^k$ in order to minimise J in direction $\mathbf{s}^k$. While ideally the global minimum would be found, in practice the computational cost is too high. Therefore, inexact line search algorithms were developed [90].

Finally, convergence of the optimisation algorithm can be checked in several ways. If $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}} = \mathbf{0}$, a local minimum of the objective function has already been found and the gradient-based optimisation process should stop. Therefore, many convergence criteria rely on the norms of the gradient vector. Another option is to check the change in value of the objective function from optimisation iteration to optimisation iteration. If this change is below a certain threshold, the optimisation procedure can be considered converged.

Steepest-descent algorithm

One simple way to ensure that a descent direction is found is by using the conjugate of the gradient vector as the direction. Substituting $\mathbf{s}^T = -\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}}$ into Eq. (3.2), it can be seen that if $\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}} \neq \mathbf{0}$ the condition is met. This is called the steepest-descent approach.

Despite its simplicity, this method along with a backtracking line search technique [90, Ch. 3] has been successfully used in order to minimise fatigue loads in wind turbine blades [91] and for flutter suppression in 2D aerofoils [92].

3.2 Calculation of gradients using variational methods

The previous section has presented two families of optimisation methods: gradient-free and gradient-based approaches. Gradient-based techniques use the value of the gradients of the objective function with respect to the design variables in order to obtain a direction of descent. The two main groups of algorithms to obtain these gradients are variational and adjoint methods. Variational approaches are described in the present section.

The simplest variational gradient calculation technique, both conceptually and with regard to implementation, is finite differencing. In this approach, a design variable is modified by a small amount, and the problem is recalculated.

There are three main ways to perform the modification: forward, backward and central differences. Start from the Taylor series expansion of the objective function, J, around the original value of a design variable, ξ_0

$$J(\xi) = J(\xi_0) + \sum_{j=1}^{+\infty} \frac{1}{j!} \cdot \left.\frac{\mathrm{d}^j J}{\mathrm{d}\xi^j}\right|_{\xi=\xi_0} \cdot (\xi - \xi_0)^j \,. \tag{3.4}$$

A step for the value of the design variable, $\delta\xi$, is defined. Forward differencing sets the new value of the design variable to $\xi = \xi_0 + \delta\xi$. Substitute it into Eq. (3.4) and expand the Taylor series

$$J(\xi_0 + \delta\xi) = J(\xi_0) + \left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} \cdot \delta\xi + \frac{1}{2} \cdot \left.\frac{\mathrm{d}^2 J}{\mathrm{d}\xi^2}\right|_{\xi=\xi_0} \cdot \delta\xi^2 + \ldots . \tag{3.5}$$

Operating and isolating the first derivative term, the estimate of the gradient for forward differences is

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} = \frac{J(\xi_0 + \delta\xi) - J(\xi_0)}{\delta\xi} + O\left(\delta\xi\right). \tag{3.6}$$

For backward differences, the sign of the change in the design variable is negative.

$$J(\xi_0 - \delta\xi) = J(\xi_0) - \left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} \cdot \delta\xi + \frac{1}{2} \cdot \left.\frac{\mathrm{d}^2 J}{\mathrm{d}\xi^2}\right|_{\xi=\xi_0} \cdot \delta\xi^2 + \ldots . \tag{3.7}$$

Therefore, the gradient is estimated as

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} = \frac{J(\xi_0) - J(\xi_0 - \delta\xi)}{\delta\xi} + O\left(\delta\xi\right). \tag{3.8}$$

Both the backward and forward difference methods are first-order. They require calculating the objective function once for the original value of $\boldsymbol{\xi}$ and an extra time per design variable. Therefore, the number of computations of J is $n+1$, where n is the number of design variables.

Central differences increase the accuracy of the gradient estimation by eliminating second-order effects. The method achieves this by evaluating the objective function twice per design variable. Subtracting Eq. (3.7) from Eq. (3.5) only the odd derivative terms remain

$$J(\xi_0 + \delta\xi) - J(\xi_0 - \delta\xi) = 2 \left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} \cdot \delta\xi + \frac{1}{3} \cdot \left.\frac{\mathrm{d}^3 J}{\mathrm{d}\xi^3}\right|_{\xi=\xi_0} \cdot \delta\xi^3 + \dots . \tag{3.9}$$

The estimate of the gradient for central differences is

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} = \frac{J(\xi_0 + \delta\xi) - J(\xi_0 - \delta\xi)}{2 \cdot \delta\xi} + O\left(\delta\xi^2\right). \tag{3.10}$$

In this case, the objective function needs to be computed $2 \cdot n + 1$ times for n design variables.

In order to properly apply finite differencing, one ought to study how the gradient estimate changes with $\delta\xi$. If this change is too large, there will be truncation errors related to it, $O(\delta\xi)$ for forward and backward differences and $O(\delta\xi^2)$ for central differences. These orders of magnitude come from the neglected terms of the Taylor series expansion in Eqs. (3.6), (3.8) and (3.10). However, if the change is too small, condition error and a consequent loss of precision can appear. Two sources of condition error are the use of a solution that is not sufficiently converged and the lack of precision in the value of the objective function.

A complex-step method was proposed to solve this latter issue. Substitute a complex step of $\mathrm{i} \cdot \delta\xi$ in the series expansion in Eq. (3.4)

$$J(\xi_0 + \delta\xi) = J(\xi_0) + \left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} \cdot \mathrm{i} \cdot \delta\xi - \frac{1}{2} \cdot \left.\frac{\mathrm{d}^2 J}{\mathrm{d}\xi^2}\right|_{\xi=\xi_0} \cdot \delta\xi^2 + \dots . \tag{3.11}$$

Note that the second-order term is real-valued and has a change of sign. Therefore, applying the imaginary part operator, $\mathcal{I}$, eliminates this term but not the third-order term. The gradients of objective function J with respect to design variable ξ are estimated as

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\xi}\right|_{\xi=\xi_0} = \frac{\mathcal{I}\left(J(\xi_0 + \mathrm{i} \cdot \delta\xi)\right)}{\delta\xi} + O\left(\delta\xi^2\right). \tag{3.12}$$

The complex-step method improves the accuracy of the gradients for low values of $\delta\xi$, since cancellation errors are absent. This is because there is no subtraction in

Eq. (3.12), only an imaginary-part operator. Therefore, much smaller values of $\delta\xi$ can be used, further reducing the error. Furthermore, the accuracy of Eq. (3.12) is comparable to that of the central differences method because the $O\left(\delta\xi^2\right)$ term in Eq. (3.11) is real, not imaginary. For n design variables, 1 real and n complex calculations are required. However, the complex-step method usually requires adapting the code in order to use complex numbers, while finite differences methods do not [93].

For higher-order derivatives, the complex-step method also suffers from cancellation errors. In order to calculate second-order derivatives without these errors, other variational methods such as hyper-dual numbers have been developed [94].

3.3 Basics of adjoint methods

The generic optimisation problem, algorithms to solve it and variational methods to obtain the gradients have been presented in previous sections. For gradient-based algorithms, the gradients of the objective function with respect to each one of the design variables must be calculated. However, calculating these gradients with finite differences or the complex-step method can be very expensive when there are many design variables. This is the case, for example, in early stages of the design process. In contrast, the adjoint method obtains the gradient of a given objective function with respect to many design variables with one computation.

Adjoint methods have a long history in fluid dynamics. Starting with Pironneau [95] and Jameson [96] for aerodynamics, they represent a more efficient way of obtaining the gradient of an objective function, J, with respect to many design variables, $\boldsymbol{\xi}$. Jameson originally based his approach on control theory [96]. There are two main ways of deriving the adjoint approach, the duality and the Lagrange formulations [97].

In the present section, the duality formulation is introduced. Then, the adjoint problem is derived using the Lagrange formulation. The resulting equations are compared.

3.3.1 Duality formulation

In linear programming, the optimisation problem is defined as

$$\min\ \mathbf{g}^T\mathbf{u}, \quad \text{subject to } \mathbf{A}\mathbf{u} = \mathbf{f}, \quad \mathbf{u} \geq 0 \tag{3.13}$$

for a given matrix $\mathbf{A}$ and vector $\mathbf{f}$. The duality formulation allows to then rewrite $\mathbf{g}^T\mathbf{u}$ as $\mathbf{v}^T\mathbf{f}$, with $\mathbf{A}^T\mathbf{v} = \mathbf{g}$. The dual form can be derived from Eq. (3.13)

$$\mathbf{g}^T\mathbf{u} = \left(\mathbf{A}^T\mathbf{v}\right)^T\mathbf{u} = \mathbf{v}^T\left(\mathbf{A}\mathbf{u}\right) = \mathbf{v}^T\mathbf{f}. \tag{3.14}$$

The dual of the problem in Eq. (3.13) is

$$\max \ \mathbf{f}^T\mathbf{v}, \quad \text{subject to } \mathbf{A}^T\mathbf{v} \leq \mathbf{g}. \tag{3.15}$$

If this dual problem has a solution, then the primal problem from which it is derived also has a solution and vice-versa. Furthermore, the values of the maximum of $\mathbf{f}^T\mathbf{v}$ and the minimum of $\mathbf{g}^T\mathbf{u}$ are equal. This is the strong duality theorem [90].

For a given problem the equations solved can be written as

$$\mathbf{R}\left(\mathbf{x}, \boldsymbol{\xi}\right) = 0, \tag{3.16}$$

where $\mathbf{R}$, the residual, is a function of the solution $\mathbf{x}$ and the design variables in $\boldsymbol{\xi}$. The length of the solution vector is m and that of the design variables vector is n. Linearising Eq. (3.16) around the original solution, $\mathbf{x}_0$, and differentiating the objective function with respect to the design variables one obtains

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}} = \frac{\partial J}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} + \frac{\partial J}{\partial \boldsymbol{\xi}}, \tag{3.17}$$

subject to

$$\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} = 0. \tag{3.18}$$

On the right-hand side of Eq. (3.17), there are two terms: the direct dependence of the objective function on the design variables, $\frac{\partial J}{\partial \boldsymbol{\xi}}$, and the dependence through the solution, $\frac{\partial J}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}$. This second term is the product of a row vector of length m, $\frac{\partial J}{\partial \mathbf{x}}$, and a matrix of size $m \times n$, $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}$.

Equations (3.17) and (3.18) can be tied to the terms in the linear programming problem shown in Eq. (3.13). In this case, $\frac{\partial J}{\partial \mathbf{x}}$ is equivalent to $\mathbf{g}^T$, $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}$ to $\mathbf{u}$, $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$ to $\mathbf{A}$ and $-\frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}}$ to $\mathbf{f}$. A vector of adjoint variables, λ, can be introduced to fulfil the role of $\mathbf{v}$ in the dual of the linear programming problem, shown in Eq. (3.15). Its expression is

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right)^T \lambda - \left(\frac{\partial J}{\partial \mathbf{x}}\right)^T = \mathbf{0}. \tag{3.19}$$

Taking the first term on the right-hand side of Eq. (3.17) and substituting from Eq. (3.19) leads to

$$\frac{\partial J}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} = \left[\left(\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right)^T \lambda\right]^T \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} = \lambda^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}. \tag{3.20}$$

Then, substitute Eq. (3.18)

$$\lambda^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}} = \lambda^T \left(-\frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \right). \tag{3.21}$$

Finally, it is possible to substitute Eq. (3.21) into Eq. (3.17) such that the gradient vector is written as

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\xi}} = \frac{\partial J}{\partial \boldsymbol{\xi}} - \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}}. \tag{3.22}$$

This derivation is equal to the one in Eq. (3.14) for obtaining the dual of the linear programming optimisation problem.

The main differences between Eqs. (3.17) and (3.22) are on the evaluation of λ and $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}$. The adjoint variables, λ, as obtained from Eq. (3.19), depend exclusively on the solution, $\mathbf{x}_0$, and the objective function, J. On the other hand, $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\xi}}$ depends on each design variable in $\boldsymbol{\xi}$. Therefore, in the direct formulation one calculation is needed per *design variable*, while in the adjoint formulation one calculation is needed per *objective function*.

Besides the adjoint variables, the terms $\frac{\partial J}{\partial \boldsymbol{\xi}}$ and $\frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}}$ have to be calculated. While those two terms depend on the number of design variables studied, they are generally much faster to calculate than the adjoint variables since they do not require matrix inversions.

3.3.2 Formulation using Lagrange multipliers

While the duality formulation is based on control theory, Lagrange multipliers are very commonly used in many engineering problems. As such, an alternative derivation of the adjoint equations using this approach is widely used.

In the Lagrange formulation, a so-called augmented objective function, I is defined

$$I(\mathbf{x}, \boldsymbol{\xi}) = J(\mathbf{x}, \boldsymbol{\xi}) - \lambda^T \mathbf{R}(\mathbf{x}, \boldsymbol{\xi}), \tag{3.23}$$

where λ is the adjoint variable vector, acting as Lagrange multipliers. Since as per Eq. (3.16) the residual is 0, the value of the augmented objective function is equal to that of the original function, J.

Differentiate the function I with respect to the design and solution variables

$$\mathrm{d}I = \left(\frac{\partial J}{\partial \mathbf{x}} - \lambda^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \right) \mathrm{d}\mathbf{x} + \left(\frac{\partial J}{\partial \boldsymbol{\xi}} - \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \right) \mathrm{d}\boldsymbol{\xi}. \tag{3.24}$$

Then, choosing a λ that satisfies

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{x}} \right)^T \lambda - \left(\frac{\partial J}{\partial \mathbf{x}} \right)^T = \mathbf{0}, \tag{3.25}$$

the d$\mathbf{x}$ term in Eq. (3.24) cancels out. From Eq. (3.25), the adjoint variables depend only on the solution and the objective function, not on the design variables, $\boldsymbol{\xi}$.

Thus, with the adjoint variables as Lagrange multipliers, the gradient of the augmented objected function with respect to the design parameters is

$$\frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{\xi}} = \frac{\partial J}{\partial \boldsymbol{\xi}} - \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}}. \tag{3.26}$$

This expression is equivalent to the one obtained using the duality formulation, Eq. (3.22).

3.4 Continuous and discrete adjoint methods

There are two main ways of implementing adjoint methods: the continuous and the discrete approach. In the first case, the adjoint equations corresponding to the problem being studied are derived and then discretised. In the second case, the adjoint problem of the already-discretised equations is computed [98].

The continuous adjoint generally requires fewer operations and less memory [99]. However, it only provides the exact gradient of the problem if the mesh is sufficiently converged [98]. In fluid mechanics, it has been widely used for the Euler equations. In the case of the Reynolds-averaged Navier-Stokes (RANS) equations, the addition of turbulence modelling significantly complicates the procedure. Thus, in some cases the frozen viscosity assumption is applied. This assumes that the value of the viscosity does not change significantly with respect to the design variables [100, 101]. However, there have been derivations of the continuous adjoint of the RANS equations. For example, Bueno-Orovio et al. derived them for the Spalart-Allmaras turbulence model [102]. They obtained a better match with finite differences compared to the frozen viscosity approach [99].

The discrete adjoint provides the exact gradient of the discretised objective function independently of the grid used. However, since it applies the adjoint method to the already-discretised equations, it depends on the numerical methods used [98].

3.4.1 Algorithmic differentiation

The analytical discrete adjoint equations could be obtained in principle for given equations and discretisation. However, changing the discretisation results in the $\partial\mathbf{R}/\partial\mathbf{x}$ term required for Eq. (3.19) being modified. A code that uses algorithmic differentiation (AD) will adapt to the numerical changes introduced by changing the discretisation without needing to re-write the adjoint equations. Furthermore, in some cases the exact Jacobian is not available or is difficult to obtain. In those

cases, AD allows to apply the adjoint method without explicitly obtaining the Jacobian. The two main ways in which AD packages differentiate codes are: source-code transformation, in which the package outputs a differentiated code, and operator overloading, in which the original code's types and operators are overloaded by their differentiated counterparts [103].

Algorithmic differentiation can be used in either the forward mode or the backward mode. The forward mode obtains the gradient of many objective functions with respect to a given design variable. It was shown to be equivalent to the complex-step method [93]. The reverse mode allows for computing the gradients of an objective function with respect to many variables, which is equivalent to the adjoint method.

The two main strategies to store the values needed to compute the derivatives are primal value and Jacobian taping. In Jacobian taping, only the derivatives with respect to the arguments of the functions are used, while the value of the arguments and the operations are needed in primal value taping. The main advantage of this last method is that it requires less memory per elementary operation [104, 105].

3.5 Reduction of objective functions

As explained in Sec. 3.3, the adjoint method requires one computation for each function for which the gradient vector is needed. This can be either the objective function, J, or each of the constraints, K. In some cases there can be many different constraints. One example is during aerostructural optimisation. The structure has to withstand the loads that are applied to it. If using a finite element formulation, every element in the structural analysis must be under the limit stress. Therefore, there is a constraint for each element. In many practical applications, this can result in $\mathcal{O}(10^4)$ elements and constraints [106]. In order to reduce the number of constraints, several methods have been developed. The present section describes two families: penalty terms and constraint aggregation.

3.5.1 Penalty terms

Penalty terms are widely used in applications in which constraints appear. Besides optimisation, they are used to represent contact between solids in structural problems.

The quadratic penalty method adds a quadratic term to the objective function. For equality and inequality constraints the penalised objective function, Q, is given by

$$Q\left(\boldsymbol{\xi}, \mu\right) = J(\boldsymbol{\xi}) + \frac{\mu}{2} \sum_i K_i^2(\boldsymbol{\xi}) + \frac{\mu}{2} \sum_i \min\left(K_i(\boldsymbol{\xi}), 0\right)^2, \tag{3.27}$$

where $\mu > 0$ is the penalty parameter. This approach reduces the problem from a constrained optimisation to an unconstrained optimisation. However, it does not guarantee that the constraints are met. In order to converge towards a solution that meets them the penalty parameter can be increased. Sometimes, this is done by using an adaptation procedure [90, Ch. 17].

Another particular case of penalty terms is the augmented Lagrangian method. In order to ensure that the equality constraints are met, a set of Lagrange multipliers can be introduced to Eq. (3.27)

$$Q\left(\boldsymbol{\xi}, \mu\right) = J(\boldsymbol{\xi}) - \sum_i \lambda_i \cdot K_i(\boldsymbol{\xi}) + \frac{\mu}{2} \sum_i K_i^2(\boldsymbol{\xi}) + \frac{\mu}{2} \sum_i \min\left(K_i(\boldsymbol{\xi}), 0\right)^2, \quad (3.28)$$

where λ_i is the ith Lagrange multiplier. Note that these Lagrange multipliers are different from those introduced in order to derive the adjoint method in Eq. (3.23). Starting from an initial guess, the value of λ_i at iteration $k+1$ is given by

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k \cdot K_i(\boldsymbol{\xi}_k). \quad (3.29)$$

This method was used by Lambe et al. in order to minimise the weight of a wing box while preventing structural failure under climbing and diving conditions [107] and by Jansen and Perez to reduce the weight of truss structures with buckling constraints [108]. It can also be extended for inequality constraints [90, Ch. 17].

3.5.2 Constraint aggregation

Another way of reducing the number of constraints is by aggregating them into one. That is, to obtain a function f of the constraints such that

$$\max\left(K_0, \ldots, K_{N-1}\right) \leq f\left(K_0, \ldots, K_{N-1}\right), \quad (3.30)$$

where K_n is the nth constraint. The value of the aggregated function must be greater than or equal to the maximum of the constraint. Therefore, if any constraint is violated, so is f.

One example of functions used for constraint aggregation are Kreisselmeier-Steinhauser functions. They are defined as [109]

$$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi})] = \frac{1}{\rho} \ln\left[\sum_{j=0}^{N-1} e^{\rho K_j(\boldsymbol{\xi})}\right], \quad (3.31)$$

where $\mathbf{K}$ is the vector of constraints, N is the number of constraints and ρ is a positive parameter. This parameter controls the difference between the value of

$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi})]$ and that of the maximum constraint. As ρ increases, the allowable design space is expanded. However, for large values of ρ the problem becomes ill-conditioned.

This family of functions has some desirable properties for constraint aggregation [109, 110]

$$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho] > \max\left[\mathbf{K}(\boldsymbol{\xi})\right] \tag{3.32}$$

$$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho] \leq \max\left[\mathbf{K}(\boldsymbol{\xi})\right] + \frac{\ln N}{\rho} \tag{3.33}$$

$$\lim_{\rho \to \infty} \mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho] = \max\left[\mathbf{K}(\boldsymbol{\xi})\right] \tag{3.34}$$

$$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho_1] \geq \mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho_2] \quad \forall \rho_1 > \rho_2. \tag{3.35}$$

Per Eq. (3.32), $\max\left[\mathbf{K}(\boldsymbol{\xi})\right]$ is a lower bound of $\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi}), \rho]$. Equation (3.33) gives an upper bound for the value of the KS function depending on the number of constraints and the value of ρ. The properties described in Eq. (3.34) and Eq. (3.35) guarantee that as the parameter ρ increases, the value of the Kreisselmeier-Steinhauser function converges monotonously towards $\max\left[\mathbf{K}(\boldsymbol{\xi})\right]$.

In order to avoid cancellation error, a modification of the function was applied for aerostructural optimisation using the adjoint method by Poon and Martins [106]

$$\mathrm{KS}[\mathbf{K}(\boldsymbol{\xi})] = K_{\max}(\boldsymbol{\xi}) + \frac{1}{\rho} \ln \left[\sum_{j=1}^{N} e^{\rho(K_j(\boldsymbol{\xi}) - K_{\max}(\boldsymbol{\xi}))} \right], \tag{3.36}$$

where $K_{\max}$ is the maximum of the set of constraints, $\mathbf{K}$. The expressions for the Kreisselmeier-Steinhauser function given in Eq. (3.31) and Eq. (3.36) are equivalent [110].

As opposed to penalty terms, with Eq. (3.30) all the constraints must be met. However, for any finite ρ the maximum value of the constraints, $K_{\max}$, is lower than the value of the function as per Eq. (3.32). Therefore, the final results can be far from the boundary of the design envelope. This can result in a solution that is not close to the optimum.

3.6 Application to fluid-structure interaction

The adjoint method for one field has been presented in Sec. 3.3 and various techniques used to reduce the number of objective functions have been introduced in Sec. 3.5. In this section, the method is extended to partitioned fluid-structure interaction (FSI) problems, which can be generalised to other multi-domain problems.

First, the adjoint equations for FSI will be derived. Then, the interpolation strategies used in order to communicate the adjoint variables will be presented.

The three-field formulation for a direct FSI problem was presented in Sec. 2.1.2. This formulation is shown in Eq. (2.19). The objective function can depend on each of the three fields: structural ($\mathcal{S}$), fluid ($\mathcal{F}$) and mesh ($\mathcal{M}$). It can be written as

$$J = J_{\mathcal{S}} + J_{\mathcal{F}} + J_{\mathcal{M}}. \tag{3.37}$$

An example of structural objective function is the weight of the structure. A fluid objective function could be the drag coefficient. The mesh objective function is included for completeness. Then, by using the linearisation around the solution in Eq. (2.20) it is possible to define the corresponding adjoint equation

$$\begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \mathbf{u}}^T & 0 & \frac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \\ \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^T & \frac{\partial \mathcal{F}}{\partial \mathbf{w}}^T & 0 \\ \frac{\partial \mathcal{S}}{\partial \mathbf{z}}^T & \frac{\partial \mathcal{F}}{\partial \mathbf{z}}^T & \frac{\partial \mathcal{M}}{\partial \mathbf{z}}^T \end{bmatrix} \begin{bmatrix} \lambda_u \\ \lambda_w \\ \lambda_z \end{bmatrix} = \begin{bmatrix} \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{u}} + \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}} \\ \frac{\partial J_{\mathcal{F}}}{\partial \mathbf{w}} + \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}} \\ \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{z}} + \frac{\partial J_{\mathcal{F}}}{\partial \mathbf{z}} + \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{z}} \end{bmatrix}, \tag{3.38}$$

where λ_u, λ_w and λ_z are the respective adjoint variables for each solver. The individual sets of equations can be rewritten as

$$\begin{cases} \frac{\partial \mathcal{S}}{\partial \mathbf{u}}^T \lambda_u &= \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{u}} + \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}} - \frac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \lambda_z \\ \frac{\partial \mathcal{F}}{\partial \mathbf{w}}^T \lambda_w &= \frac{\partial J_{\mathcal{F}}}{\partial \mathbf{w}} + \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^T \lambda_u \\ \frac{\partial \mathcal{M}}{\partial \mathbf{z}}^T \lambda_z &= \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{z}} + \frac{\partial J_{\mathcal{F}}}{\partial \mathbf{z}} - \frac{\partial \mathcal{F}}{\partial \mathbf{z}}^T \lambda_w + \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{z}} - \frac{\partial \mathcal{S}}{\partial \mathbf{z}}^T \lambda_u \end{cases}. \tag{3.39}$$

Each set of equations can be solved in a staggered way, like they are for the direct FSI problem. Isolate the source terms introduced by the coupling in Eq. (3.39)

$$\begin{cases} \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}} - \frac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \lambda_z \\ \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^T \lambda_u \\ \frac{\partial J_{\mathcal{F}}}{\partial \mathbf{z}} - \frac{\partial \mathcal{F}}{\partial \mathbf{z}}^T \lambda_w + \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{z}} - \frac{\partial \mathcal{S}}{\partial \mathbf{z}}^T \lambda_u \end{cases}. \tag{3.40}$$

Comparing these coupling terms to Eq. (3.22), they represent the gradients of the objective functions with respect to the other solvers' output variables. Except in the case of the mesh, these terms are only non-zero for the boundary values that are transferred between solvers. The position of the mesh nodes has an effect on the fluid solver independently of their location on or off the boundary. An augmented objective function can be defined for each field; $I_{\mathcal{S}}$ for the solid, $I_{\mathcal{F}}$ for the fluid, $I_{\mathcal{M}}$ for the mesh:

$$\begin{cases} I_{\mathcal{S}} = J_{\mathcal{S}} + \left(\dfrac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}} - \dfrac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \lambda_z \right) \mathbf{u} \\ I_{\mathcal{F}} = J_{\mathcal{F}} + \left(\dfrac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}} - \dfrac{\partial \mathcal{S}}{\partial \mathbf{w}}^T \lambda_u \right) \mathbf{w} \\ I_{\mathcal{M}} = J_{\mathcal{M}} + \left(\dfrac{\partial J_{\mathcal{F}}}{\partial \mathbf{z}} - \dfrac{\partial \mathcal{F}}{\partial \mathbf{z}}^T \lambda_w + \dfrac{\partial J_{\mathcal{S}}}{\partial \mathbf{z}} - \dfrac{\partial \mathcal{S}}{\partial \mathbf{z}}^T \lambda_u \right) \mathbf{z} \end{cases} . \tag{3.41}$$

The gradients of these one-field augmented objective functions are equal to the gradients of the coupled objective function.

3.6.1 Interpolation of adjoint gradients

As in the direct case, presented in Sec. 2.1.3, solving the partitioned FSI adjoint equations requires some interpolation between the fluid and solid boundaries. The simplest case is that of matching boundary meshes, in which each node on the fluid boundary corresponds to one node on the solid boundary. In that case, the gradients are transferred from node to node. For other cases, the adjoint interpolation depends on the direct interpolation.

This interpolation affects the crossterms in Eq. (3.39). As per Eq. (2.23), for a linear interpolation

$$\mathbf{u}_f = \mathbf{H}\mathbf{u}_s, \tag{3.42}$$

where $\mathbf{u}_f$ and $\mathbf{u}_s$ are the nodal displacements at the fluid and solid boundary, respectively and $\mathbf{H}$ is the solid-to-fluid interpolation matrix. Define the source term introduced in Eq. (3.39) to the solid adjoint equation as a function of the fluid interface displacements

$$\left. \frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}_f} \right|_{\mathcal{M}} = \frac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}_f} - \frac{\partial \mathcal{M}}{\partial \mathbf{u}_f}^T \lambda_z. \tag{3.43}$$

This term is a row vector while the displacements are column vectors. For consistency, transpose this gradient and apply the chain rule based on the linear interpolation in Eq. (3.42)

$$\left. \frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}_s} \right|_{\mathcal{M}}^T = \mathbf{H}^T \left. \frac{\mathrm{d}J}{\mathrm{d}\mathbf{u}_f} \right|_{\mathcal{M}}^T = \mathbf{H}^T \left(\frac{\partial J_{\mathcal{M}}}{\partial \mathbf{u}_f} - \frac{\partial \mathcal{M}}{\partial \mathbf{u}_f}^T \lambda_z \right)^T . \tag{3.44}$$

This is the value of the source term as a function of the solid boundary displacements. The same procedure can be applied to the fluid variables at the boundary. The fluid-to-solid interpolation matrix $\mathbf{G}$ is defined such that

$$\mathbf{w}_s = \mathbf{G}\mathbf{w}_f, \tag{3.45}$$

where $\mathbf{w}_s$ represents the relevant fluid variables applied to the solid nodes and $\mathbf{w}_f$ are the fluid variables on the fluid nodes. Usually, for FSI problems the variables transferred from the fluid to the solid are the nodal forces. Express the coupling source term in the fluid adjoint equation in Eq. (3.39) as a function of the interpolated variables and then apply the chain rule:

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\mathbf{w}_s}\right|_{\mathcal{S}} = \frac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}_s} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}_s}^T \lambda_u; \tag{3.46}$$

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\mathbf{w}_f}\right|_{\mathcal{S}}^T = \mathbf{G}^T \left.\frac{\mathrm{d}J}{\mathrm{d}\mathbf{w}_s}\right|_{\mathcal{S}}^T = \mathbf{G}^T \left(\frac{\partial J_{\mathcal{S}}}{\partial \mathbf{w}_s} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}_s}^T \lambda_u\right)^T. \tag{3.47}$$

The source term at the boundary for the fluid adjoint equation is thus obtained.

In the case of a conservative approach, as seen in Sec. 2.1.3, the solid-to-fluid displacement interpolation matrix $\mathbf{H}$, in Eq. (2.23), is transposed to obtain the fluid-to-solid force interpolation matrix $\mathbf{G}$, in Eq. (2.26). Therefore, the matrices can be re-used for the adjoint problem. This is not the case in a consistent interpolation approach. In that case, transposing the direct transfer matrices $\mathbf{H}$ and $\mathbf{G}$ is required.

3.6.2 Coupled steady fluid-structure interaction adjoint algorithm

A coupled FSI adjoint method for partitioned solvers has been presented in the previous sections. The algorithm used for its implementation is detailed in the present section. It is based on the technique used to obtain the steady FSI direct solution. Figure 3.1 shows a flowchart of the algorithm.

First, the direct FSI solution has to be computed. The converged fluid and solid solutions obtained using the direct method are loaded in memory by the respective adjoint solvers. Then, the solid displacements obtained from the direct solution are interpolated and applied to the fluid boundary. The fluid mesh is deformed depending on the value of these displacements. This part of the process sets the converged direct FSI solution.

The fluid adjoint code solves the adjoint equations. Then, the gradient of the objective function with respect to the displacements of the fluid boundary is extracted. This process is undertaken by the mesh solver. The gradient vector is

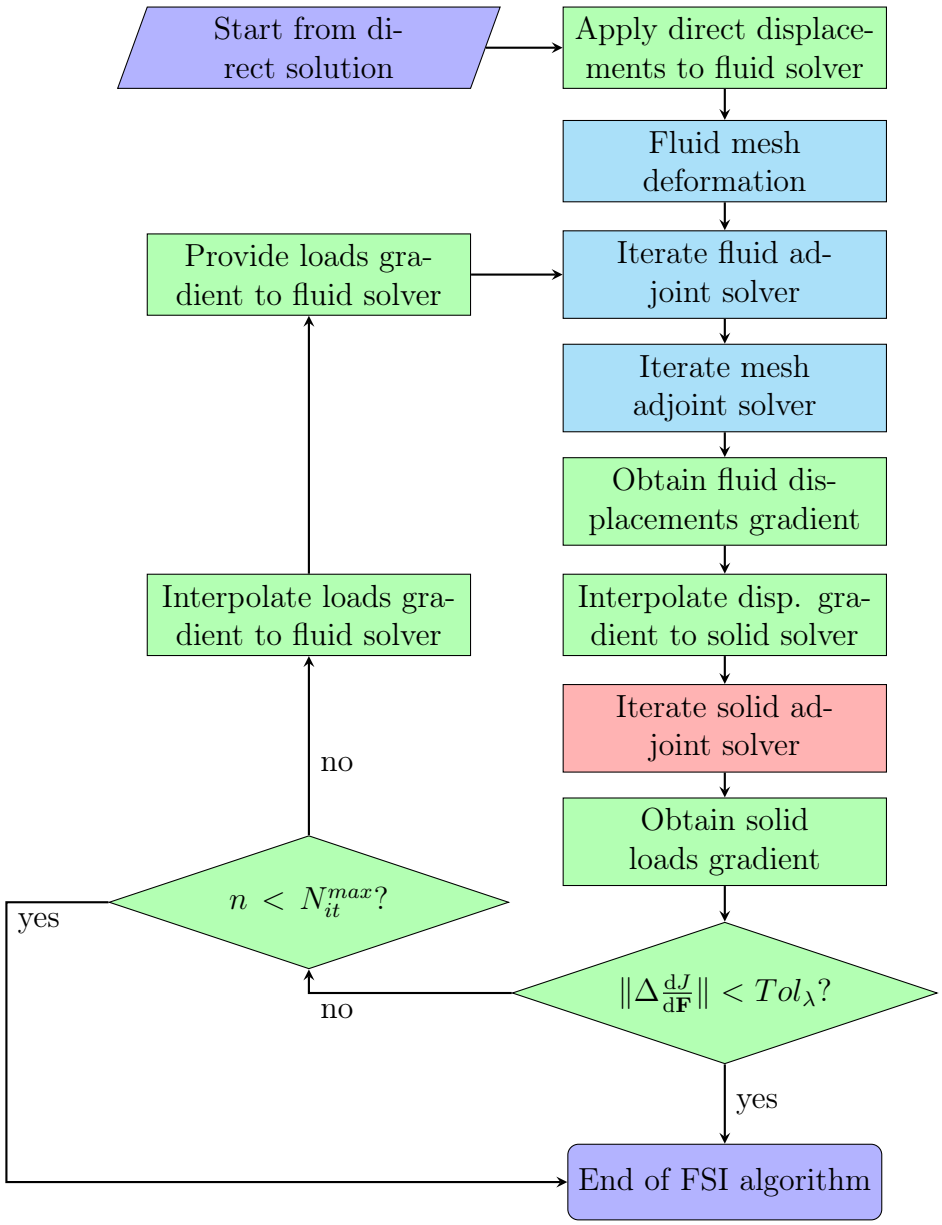


Figure 3.1: Flowchart describing the steady adjoint fluid-structure interaction algorithm. In light blue, the actions carried out by the adjoint fluid solver; in red, those by the adjoint structural solver; in green, those by the coupler.

interpolated to the solid boundary as described in Sec. 3.6.1. The solid adjoint code then solves its adjoint equations. This computation obtains the value of the gradient of the objective function with respect to the forces induced by the fluid at the solid boundary. The gradients with respect to the forces are interpolated to the fluid boundary. The fluid solver then takes them into account. This process is repeated until either the norm of the residual is smaller than the specified tolerance or the number of FSI adjoint iterations is higher than the specified maximum. If at the last iteration convergence is not reached, the solution should be rejected.

In SU2, both fluid and solid adjoint problems are solved implicitly as a fixed-point problem. Therefore, it is required to iterate in order to obtain the correct value of the gradients [36]. One convergence criterion for the structural adjoint solver is the change in displacement adjoint variables between iterations. The convergence criterion used for the FSI adjoint problem is based on the magnitudes transferred to the fluid solver from each solid boundary node, like in the direct case. For the adjoint solver, it is the magnitude of the change of the vector of gradients with respect to the fluid loads. This magnitude is represented as $\|\Delta\frac{\mathrm{d}J}{\mathrm{d}\mathbf{F}}\|$ in Fig. 3.1.

3.7 Verification of coupled gradients

The steady adjoint coupling algorithm described in Sec. 3.6.2 has to be verified. In order to perform this verification, a simple 2D case is solved using two different structural codes: the SU2 suite's solid solver and pyBeam. The fluid solver used in both cases was SU2 [8]. SU2 includes both a direct and an adjoint solver, with fluid-structure adjoint capabilities [36, 71]. It uses CoDiPack and the reverse mode of algorithmic differentiation in order to solve the discrete adjoint equations [105, 111]. It also has continuous adjoint capabilities but those are not used in the present work.

In order to calculate the coupled gradients, the source terms introduced in Eq. (3.41) are added using the `registerOutput` CoDiPack function call. This function records the objective function so that its gradients with respect to the design variables can be calculated using algorithmic differentiation.

For both structural codes, the coupling is performed by using and extending CUPyDO. It already could simulate direct FSI problems [37, 64]. For the present work it was extended with adjoint capabilities and provided interfaces for the two solid solvers. The implementation work performed is explained in further detail in Sec. B.1 of Appendix B.

The case consists of a beam subject to incompressible crossflow, as shown in Fig. 3.2. The flow is steady. The cantilever beam is clamped at its base. The height

of the beam is 0.01 m and its width is 5×10^{-4} m. The material's Young's modulus was set to $E = 50\,000$ Pa, while its Poisson ratio was $\nu = 0.35$.

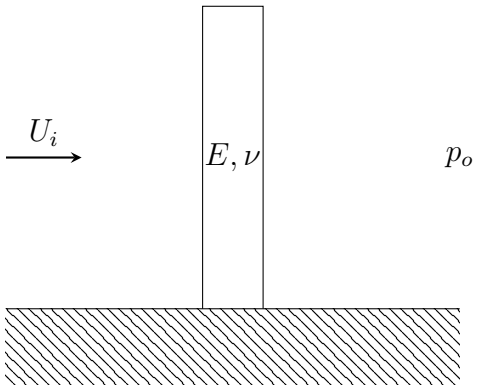


Figure 3.2: 2D beam in crossflow with relevant parameters

The Reynolds number of the flow is $\mathrm{Re} = \frac{\rho \cdot U_i \cdot h}{\mu} = 10$, where U_i is the inlet velocity and h is the height of the beam. Since the Reynolds number is very low, the flow should be laminar. Therefore, the fluid is modelled by the Navier-Stokes equations.

The numerical fluid domain used for the problem is shown in Fig. 3.3. A non-slip boundary condition is imposed on the upper and lower walls. The non-slip walls appear in blue in Fig. 3.3. The inlet and outlet are at a distance of 0.16 m from the beam upstream and downstream, respectively. They are drawn in black. A non-slip boundary condition is imposed on the beam itself, which is drawn in red in Fig. 3.3. This boundary is allowed to move, which in turn deforms the fluid mesh.

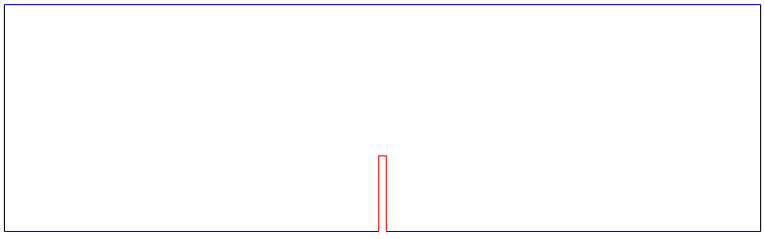


Figure 3.3: Fluid domain for 2D beam in crossflow

3.7.1 SU2 fluid solver and SU2 solid solver

The SU2 CFD suite [8] includes a solid-mechanics, geometrically nonlinear solver that implements the discrete adjoint method by using algorithmic differentiation [71].

Figure 3.4: Flow field around cantilever beam in crossflow using SU2

The test case is used in order to verify the coupling of the structural and fluid solvers through CUPyDO. In this setup, the nodes at the fluid and solid boundaries match. Therefore, a matching-meshes interpolator was used.

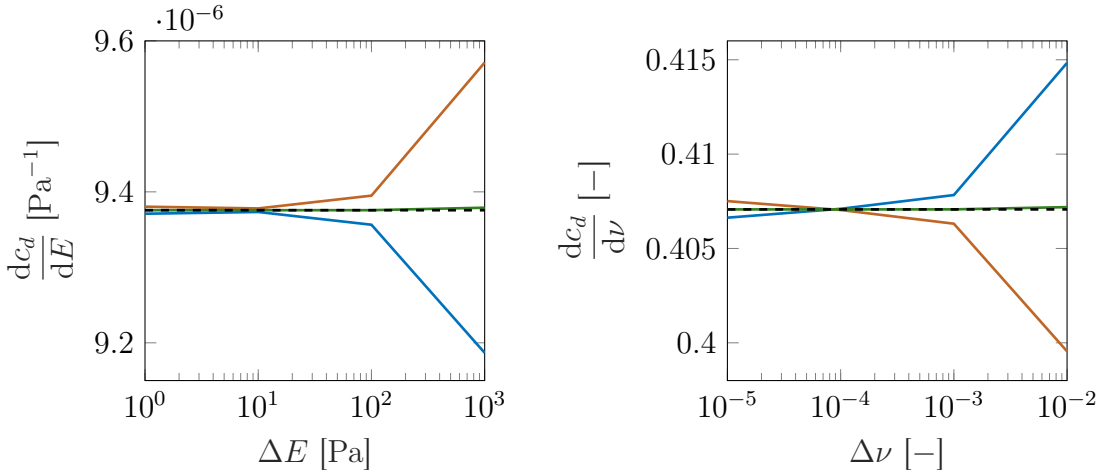
Direct simulation setup and results

Figure 3.4 shows the value of the pressure around the deflected beam. Upstream of the beam, the pressure increases as the velocity decreases. Downstream, a region of low pressure appears. At the tip of the beam, there is a region of localised, very low pressure. The loads applied cause the beam to undergo a large, nonlinear deflection that reduces its height. The top of the beam rotates. The drag coefficient was $c_d = 2.86$.

Verification of adjoint gradients

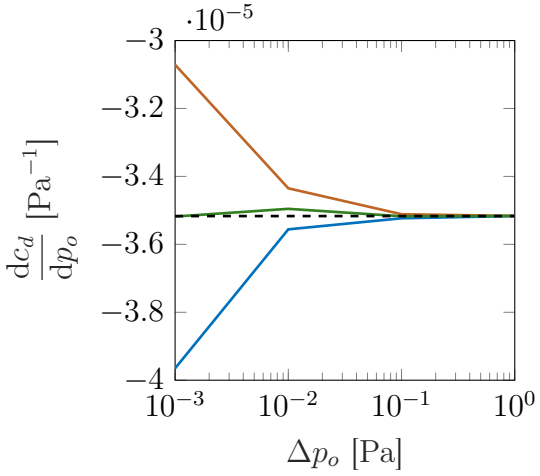
The objective function chosen is the drag coefficient (c_d). The structural parameters chosen to verify the gradients were the material's Young's modulus (E) and its Poisson ratio (ν). For the fluid solver, the pressure at the outlet (p_o) was selected.

The evolution of the gradients obtained using finite differences as a function of the design step is shown in Fig. 3.5. Forward finite differences are in blue, backward finite differences in orange and central finite differences in green. The coupled adjoint gradients are represented as a dashed black line. They do not depend on the step



(a) Young's modulus

(b) Poisson ratio



(c) Outlet pressure

Figure 3.5: Comparison of gradients obtained using forward finite differences (blue), backward finite differences (orange), central finite differences (green) and the coupled adjoint method (dashed black) as a function of the step

size. The final steps chosen in order to balance condition and truncation error are $\Delta E = \pm 10\,\text{Pa}$, $\Delta\nu = \pm 0.0001$ and $\Delta p_o = \pm 0.1\,\text{Pa}$. The outlet pressure step is large compared to the values used. As shown in Fig. 3.5(c), this magnitude is subject to important cancellation error at lower values of the finite differences step. Table 3.1 shows a comparison between adjoint and central finite differences results for the sensitivities of the drag objective function with respect to three design parameters. The digits that differ between the two methods are bold.

Method	$\frac{\mathrm{d}c_d}{\mathrm{d}E}$ [Pa^{-1}]	$\frac{\mathrm{d}c_d}{\mathrm{d}\nu}$ [$-$]	$\frac{\mathrm{d}c_d}{\mathrm{d}p_o}$ [Pa^{-1}]
Adjoint	$9.3756\mathbf{26} \times 10^{-6}$	$0.40707070\mathbf{45}$	$-3.51\mathbf{68} \times 10^{-5}$
Finite differences	$9.3756\mathbf{36} \times 10^{-6}$	$0.40707070\mathbf{85}$	$-3.51\mathbf{73} \times 10^{-5}$

Table 3.1: Verification of 2D beam in cross flow test case using the SU2 structural solver

The finite differences and adjoint gradients are quite close. Taking finite differences results as a reference, the largest relative difference is of around 0.01%. This difference is for the outlet pressure, for which there were issues with cancellation error and a relatively large step was required.

Figure 3.6 shows the evolution of the gradient of the drag coefficient with respect to Young's modulus ($\frac{\mathrm{d}c_d}{\mathrm{d}E}$) as the adjoint FSI simulation progresses. After the first iteration there is an overestimation of its value which then quickly decreases. After 10 FSI iterations the gradient is sufficiently converged, with the difference between iterations being lower than the difference between central differences and the adjoint method shown in Table 3.1.

3.7.2 SU2 fluid solver and pyBeam solid solver

pyBeam is a geometrically nonlinear beam solver that was developed by Bombardieri et al. It implements flexible beam elements and rigid elements that allow to transfer the rotation to the boundaries. The discrete adjoint method is implemented by using CoDiPack's [105] reverse mode of algorithmic differentiation. The code includes FSI coupling functions so that it can be used in aero-structural applications [112].

The main objective of the SU2 and pyBeam test case was to verify the implementation of conservative interpolation methods for the coupling of adjoint problems. A secondary objective was the verification of the pyBeam interface implemented in CUPyDO, for both the direct and adjoint solvers. The direct interface had already been started before this project. The adjoint extensions, for both the interpolation methods and the interface, were written for this work.

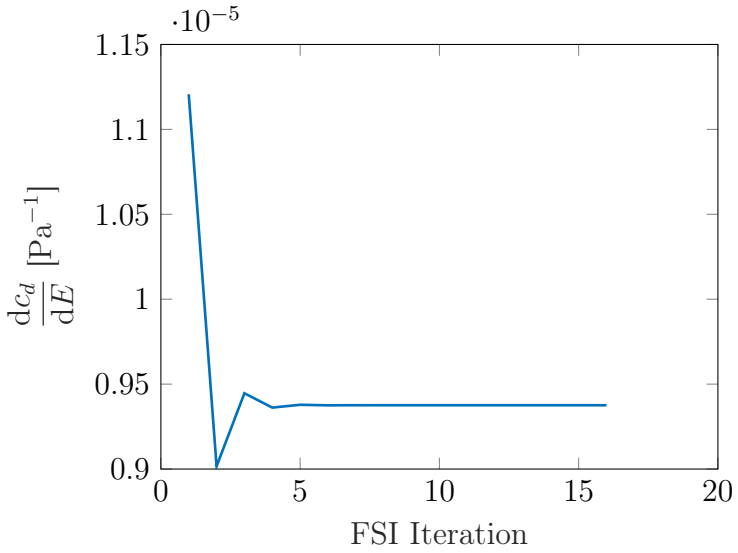


Figure 3.6: Evolution of derivative of drag coefficient with respect to Young's modulus

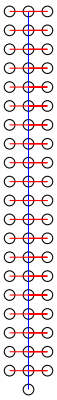


Figure 3.7: Solid mesh for 2D beam test case using beam elements

The structural model used 20 beam elements and 40 rigid elements perpendicular to the direction of the beam. These last elements rotate rigidly around the elastic axis of the beam, transferring its rotation to the fluid boundary. The model is shown in Fig. 3.7. The x-axis coordinates are exaggerated in order to show the rigid elements more clearly. Each node, of which there were 63, appears as a circle in the figure. The flexible elements are plotted in blue and the rigid elements in red. The lowest node was fixed in both the translation and rotation degrees of freedom in order to clamp the beam.

Figure 3.8: Flow field around cantilever beam in crossflow using SU2 and pyBeam

The interpolation of loads and displacements was performed using a conservative Compact $\mathcal{C}^2$ radial-basis function technique [43] already implemented in CUPyDO [64]. If all boundary nodes lie on the same straight line for a 2D case or on the same plane for a 3D case, one of the interpolation matrices, $\mathbf{C}_{ss}$, is not invertible. This results in an infinite number of possible solutions, which is not physical. The rigid elements define the outer boundary of the beam. This outer boundary, unlike the elastic axis, is not a single straight line.

The value of the pressure field around the beam is shown in Fig. 3.8. The behaviour is similar to the one observed in Sec. 3.7.1 for the SU2 solid solver, with a region of high pressures upstream and a downstream low-pressure region. The structural deformation is large. While a geometrically linear solver would maintain the height of the beam constant, the present case shows significant shortening due to the structural deformation. The drag coefficient obtained was $c_d = 2.80$. This value is smaller than the one obtained using the SU2 structural solver.

Figure 3.9 compares the undeformed geometry of the beam, in blue, the deformed shape obtained using pyBeam, in green, and the deformed shape obtained using the SU2 structural solver, in orange. The pyBeam beam is more deflected than the SU2 beam, which leads to a lower value of the drag coefficient. Using pyBeam assumes that the beam is rigid in the transverse direction, which is not the case in SU2. This could account for the difference in beam deflection.

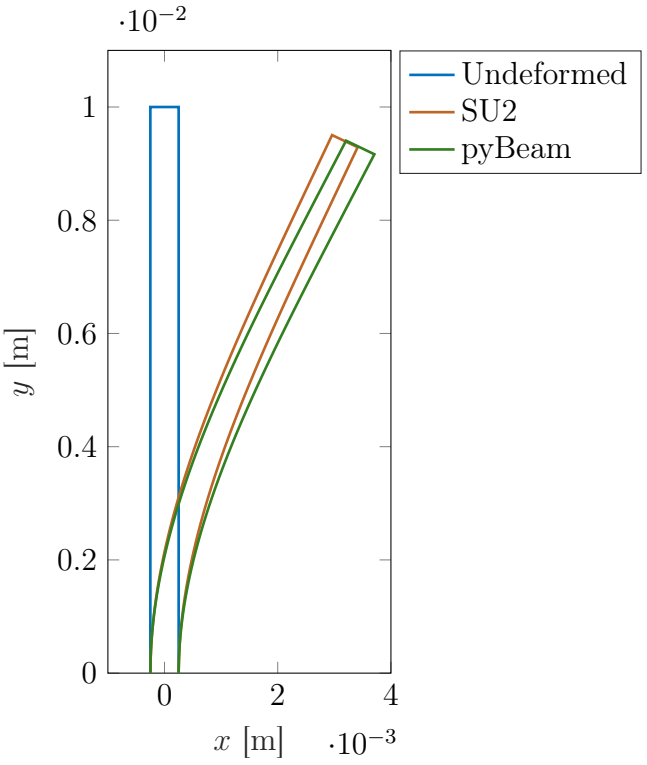


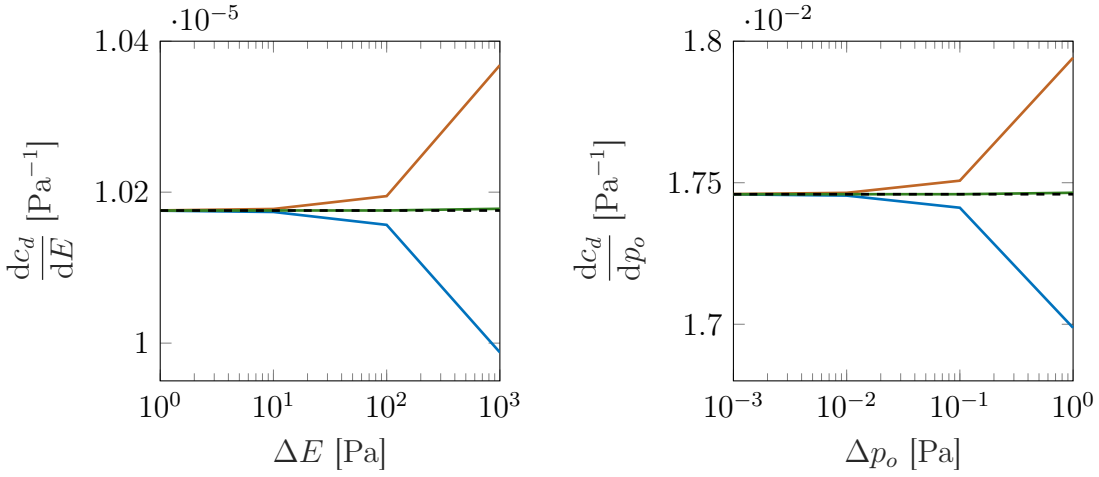
Figure 3.9: Comparison of deformed and undeformed beam geometries

Adjoint gradients

As in the previous case, the drag coefficient was chosen as the objective function. The material's Poisson's ratio does not affect the solution because the beam is rigid in the transverse direction. Therefore, the two parameters chosen are the Young's modulus of the solid material, E, and the pressure at the outlet, p_o.

The convergence of the finite difference gradients as a function of the step is shown in Fig. 3.10. Forward finite differences are in blue, backward finite differences in orange and central finite differences in green. As a reference value, the coupled adjoint gradients, which are independent of the step size, appear as a dashed black line. The steps chosen for the comparison were $\Delta E = \pm 1\,\text{Pa}$ and $\Delta p_o = \pm 0.001\,\text{Pa}$. Table 3.2 compares the gradients obtained using the adjoint method and those obtained by central finite differences.

The gradients obtained by the present method are close to those obtained by finite differences, within a few parts per million. Compared to the gradients obtained using the SU2 solid solver, the gradient with respect to the outlet pressure is approximately three orders of magnitude larger and its sign is different. This



(a) Young's modulus

(b) Outlet pressure

Figure 3.10: Comparison of gradients obtained using forward finite differences (blue), backward finite differences (orange), central finite differences (green) and the coupled adjoint method (dashed black) as a function of the step

Method	$\frac{dc_d}{dE}$ [Pa^{-1}]	$\frac{dc_d}{dp_o}$ [Pa^{-1}]
Adjoint	1.017567×10^{-5}	1.745935×10^{-2}
Finite differences	1.017564×10^{-5}	1.745933×10^{-2}

Table 3.2: Verification of 2D beam in cross flow test case using the beam structural solver

may be because of the use of a conservative interpolation approach that is not consistent. In interpolation approaches that are not consistent, a constant pressure is not recovered on the solid boundary when applied on the fluid side. Therefore, the average pressure level can have a much more significant impact on the result. The SU2 test case used a matching-meshes interpolator, which is consistent. Hence, the main effect of the increased pressure in the SU2 test case is a slight shortening of the beam that barely affects the drag coefficient. This could explain the difference in value of the gradient between the two test cases.

The gradient of the drag coefficient with respect to the material's Young's modulus is similar in magnitude to the gradient obtained using the SU2 solid solver. It is slightly larger, probably because of the larger deflection of the beam.

Figure 3.11 shows the convergence of $\frac{dc_d}{dE}$ with the number of adjoint FSI itera-

tions. The gradient shows some slight oscillations around its converged value, first overestimating it. These oscillations quickly damp out and after around 10 FSI iterations the difference between iterations is smaller than the one between finite differences and the adjoint method shown in Table 3.2. This behaviour is similar to the one observed in the previous setup with the SU2 solid solver.

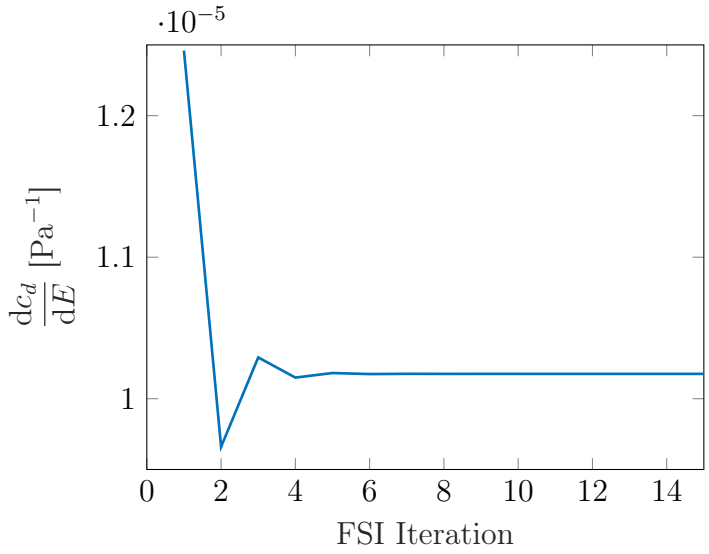


Figure 3.11: Evolution of derivative of drag coefficient with respect to Young's modulus

3.8 Summary

The present chapter has presented a partitioned approach for the computation of gradients of steady, coupled FSI problems by means of the adjoint method. This approach has been applied to a simple 2D test case in order to verify its implementation. The gradients obtained using the partitioned adjoint technique have been compared to those obtained using central finite differences. Two different structural codes have been used. The first one was a finite element-based code in which the boundary nodes of the fluid and structure matched. The second code was a beam solver that required some interpolation between the fluid and solid boundary. This interpolation was performed with a conservative radial-basis function technique. In the two test cases the gradients obtained by the present method are consistent with those obtained by finite differences.

Chapter 4

Adjoint method for the optimisation of unsteady fluid-structure interaction problems

As previously explained, the optimisation of fluid-structure interaction (FSI) problems is of growing importance. Some FSI problems are steady, but many others are unsteady. A partitioned harmonic balance (HB) approach for the calculation of unsteady time-periodic FSI problems with unknown frequency has been presented in Chapter 2. The adjoint method for partitioned steady FSI problems has been described in Chapter 3. A combination of these two approaches in order to reduce the computational cost of optimisation of unsteady FSI problems is presented in this chapter.

First, the time-domain unsteady adjoint approach is presented for a single field. The continuous-in-time and discrete-in-time derivations of the unsteady adjoint are included and compared. Since the unsteady adjoint approach has high storage requirements, some techniques for reducing this cost are introduced. Then, the definition of the unsteady objective function is discussed, with an emphasis on windowing and averaging. Afterwards, the coupled FSI HB adjoint method is derived. Some specifics of the coupling are discussed. Subsequently, aerodynamic shape optimisation and the definition of design variables is explained.

Finally, the FSI HB adjoint method is verified using one of the limit-cycle oscillation (LCO) setups from Sec. 2.8. Two objective functions are chosen: the pitching amplitude of the LCO and the mean drag. The gradients obtained by the present method are compared to those obtained via finite differences for aerodynamic shape, structural and fluid design variables.

4.1 Unsteady adjoint methods in fluid dynamics

While many cases in design where optimisation is required are steady, there has been growing interest in unsteady cases. In order to apply the adjoint method to unsteady problems, the formulation has to be adapted to the kinds of differential equations being solved to model these problems. These are unsteady adjoint methods.

Usually, unsteady problems can be modelled by a differential equation in the time domain. Similar to the difference between continuous and discrete adjoint solutions explained in Sec. 3.4, there are two ways of applying the adjoint method to these equations. The first procedure is to obtain the adjoint of the unsteady equations and then discretise them in time and the second one is to obtain the adjoint of the already-discretised equations.

4.1.1 Continuous adjoint

The partial differential equation solved by the unsteady direct problem is

$$\frac{\partial}{\partial t}\mathbf{U}(t) - \mathbf{R}\left(\mathbf{U}(t), t\right) = 0, \qquad t \in [0, t_f]\,, \tag{4.1}$$

where $\mathbf{U}(t)$ is the vector of conservative variables at time t, $\mathbf{R}$ is the residual and t_f is the final integration time. The design variables, $\boldsymbol{\xi}$, can be introduced to the expression of the residual as

$$\frac{\partial}{\partial t}\mathbf{U}(t) - \mathbf{R}\left(\mathbf{U}(t), \boldsymbol{\xi}, t\right) = 0, \qquad t \in [0, t_f]\,. \tag{4.2}$$

The objective function, J, can be defined as an integral in time

$$J = \int_0^{t_f} j\left(\mathbf{U}, \boldsymbol{\xi}\right) \mathrm{d}t, \tag{4.3}$$

where j is a function of the conservative and design variables. Linearise Eq. (4.3) around $\mathbf{U}$ and $\boldsymbol{\xi}$

$$\mathrm{d}J = \int_0^{t_f} \left(\frac{\partial j}{\partial \mathbf{U}}\mathrm{d}\mathbf{U} + \frac{\partial j}{\partial \boldsymbol{\xi}}\mathrm{d}\boldsymbol{\xi}\right) \mathrm{d}t. \tag{4.4}$$

Then, linearise Eq. (4.2) with respect to the same variables

$$\frac{\partial}{\partial t}\mathrm{d}\mathbf{U} - \left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\mathrm{d}\mathbf{U} + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}}\mathrm{d}\boldsymbol{\xi}\right) = 0. \tag{4.5}$$

The augmented objective function, I, can be obtained by introducing a Lagrange multiplier, λ, augmenting Eq. (4.4) with Eq. (4.5)

$$\begin{aligned} \mathrm{d}I = &\int_0^{t_f} \left(\frac{\partial j}{\partial \mathbf{U}} \mathrm{d}\mathbf{U} + \frac{\partial j}{\partial \boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi} \right) \mathrm{d}t - \\ &- \int_0^{t_f} \lambda^T \left[\frac{\partial}{\partial t} \mathrm{d}\mathbf{U} - \left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \mathrm{d}\mathbf{U} + \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi} \right) \right] \mathrm{d}t. \end{aligned} \tag{4.6}$$

Since as per Eq. (4.5) the second integral is 0, the augmented and original objective functions have the same value. A similar procedure, also using Lagrange multipliers, was applied to the steady case in Sec. 3.3.2.

Integrate by parts the time derivative term of the second integral in Eq. (4.6)

$$\begin{aligned} &\int_0^{t_f} \lambda^T \frac{\partial}{\partial t} \mathrm{d}\mathbf{U} \, \mathrm{d}t = \lambda^T \mathrm{d}\mathbf{U} \Big|_0^{t_f} - \int_0^{t_f} \frac{\partial \lambda^T}{\partial t} \mathrm{d}\mathbf{U} \, \mathrm{d}t = \\ &= \lambda(t_f)^T \mathrm{d}\mathbf{U}(t_f) - \lambda(0)^T \mathrm{d}\mathbf{U}(0) - \int_0^{t_f} \frac{\partial \lambda^T}{\partial t} \mathrm{d}\mathbf{U} \, \mathrm{d}t. \end{aligned} \tag{4.7}$$

Assume that the initial conditions depend exclusively on the design variables[1], $\mathrm{d}\mathbf{U}(0) = \frac{\mathrm{d}\mathbf{U}(0)}{\mathrm{d}\boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi}$. Substituting Eq. (4.7) into Eq. (4.6) and grouping the terms multiplying each variational, the equation becomes

$$\begin{aligned} \mathrm{d}I = &\lambda(0)^T \frac{\mathrm{d}\mathbf{U}(0)}{\mathrm{d}\boldsymbol{\xi}} \, \mathrm{d}\boldsymbol{\xi} - \lambda(t_f)^T \mathrm{d}\mathbf{U}(t_f) + \\ &+ \int_0^{t_f} \left[\left(\frac{\partial j}{\partial \mathbf{U}} + \frac{\partial \lambda^T}{\partial t} + \lambda^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right) \mathrm{d}\mathbf{U} + \left(\frac{\partial j}{\partial \boldsymbol{\xi}} + \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \right) \mathrm{d}\boldsymbol{\xi} \right] \mathrm{d}t. \end{aligned} \tag{4.8}$$

Define the adjoint variables, λ, so that they are a solution to the differential equation

$$\frac{\partial \lambda}{\partial t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \lambda + \frac{\partial j}{\partial \mathbf{U}}^T = 0. \tag{4.9}$$

It follows that the adjoint variables are independent of the design variables, $\boldsymbol{\xi}$. Substituting into Eq. (4.8), the $\mathrm{d}\mathbf{U}$ term cancels out leaving

$$\mathrm{d}I = \lambda(0)^T \frac{\mathrm{d}\mathbf{U}(0)}{\mathrm{d}\boldsymbol{\xi}} \, \mathrm{d}\boldsymbol{\xi} - \lambda(t_f)^T \mathrm{d}\mathbf{U}(t_f) + \int_0^{t_f} \left(\frac{\partial j}{\partial \boldsymbol{\xi}} + \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \right) \mathrm{d}\boldsymbol{\xi} \, \mathrm{d}t. \tag{4.10}$$

The final conditions of the adjoint variables can be introduced in order to eliminate the term multiplied by $\mathrm{d}\mathbf{U}(t_f)$

$$\lambda(t_f) = \mathbf{0}. \tag{4.11}$$

[1] One example of this is the freestream conditions of the flow being used to initialise the flowfield in iteration 0

Thus, the final expression of the gradient is

$$\frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{\xi}} = \lambda(0)^T \frac{\mathrm{d}\mathbf{U}(0)}{\mathrm{d}\boldsymbol{\xi}} + \int_0^{t_f} \left(\frac{\partial j}{\partial \boldsymbol{\xi}} + \lambda^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\xi}} \right) \mathrm{d}t, \tag{4.12}$$

where the adjoint variables, λ, follow the differential equation in Eq. (4.9) and have as final condition the one expressed in Eq. (4.11). Two important features of unsteady adjoint calculations appear: there is a change in sign of the residual between the direct differential equation in Eq. (4.1) and the adjoint differential equation in Eq. (4.9) and because of the final condition, the adjoint equation is integrated backwards in time.

A derivation of the unsteady, continuous adjoint equations for the unsteady Euler equations was obtained by Nadarajah and Jameson. They applied it to a minimisation of the average drag of a pitching aerofoil [113].

4.1.2 Discrete adjoint

The previous section has shown the derivation of the continuous-in-time adjoint equations. However, the direct problem is already discretised in time. The adjoint of these discrete equations can be obtained from the formulation used. The continuous time interval $[0, t_f]$ is discretised into $N+1$ time steps. For simplicity, the time step is kept constant, at $\Delta t = t_f/N$. Using a first-order implicit backward differentiation approach, the result of discretising Eq. (4.1) can be written as

$$\mathbf{R}^{*n}\left(\mathbf{U}^{n-1}, \mathbf{U}^n, \boldsymbol{\xi}\right) := \frac{\mathbf{U}^n - \mathbf{U}^{n-1}}{\Delta t} + \mathbf{R}\left(\mathbf{U}^n, \boldsymbol{\xi}\right) = 0 \qquad n \geq 1, \tag{4.13}$$

where $\mathbf{R}^{*n}$ is the unsteady residual at time step n, $\mathbf{U}^n$ is the vector of conservative variables in time step n and $\mathbf{R}$ is the steady residual. The initial conditions, applied in time step 0, are a function of the design variables:

$$\mathbf{U}^0 = \mathbf{U}_0\left(\boldsymbol{\xi}\right). \tag{4.14}$$

The objective function for a time-marching simulation is defined as

$$J\left(\boldsymbol{\xi}\right) = \sum_{n=0}^{N} \Delta t \cdot J^n\left(\mathbf{U}^n, \boldsymbol{\xi}\right), \tag{4.15}$$

where J^n is the value of the objective function at time step n. It is the discrete equivalent of the continuous function in Eq. (4.3).

Define the augmented objective function by adding the adjoint variables, acting as Lagrange multipliers, to Eq. (4.15) as

$$\begin{aligned} I\left(\boldsymbol{\xi}\right) &= J\left(\boldsymbol{\xi}\right) + \sum_{n=1}^{N} \lambda_n^T \mathbf{R}^{*n} = \\ &= \Delta t \cdot J^0\left(\mathbf{U}^0, \boldsymbol{\xi}\right) + \sum_{n=1}^{N} \Delta t \cdot J^n\left(\mathbf{U}^n, \boldsymbol{\xi}\right) + \lambda_n^T \mathbf{R}^{*n}\left(\mathbf{U}^{n-1}, \mathbf{U}^n, \boldsymbol{\xi}\right), \end{aligned} \tag{4.16}$$

where λ_n are the adjoint variables in time instance n. The unsteady residual, $\mathbf{R}^{*n}$, was defined to be 0 in Eq. (4.13). Therefore, the augmented objective function, I, is equal in value to the original one, J. Equation (4.16) can then be linearised around $\boldsymbol{\xi}$ and $\mathbf{U}$

$$\begin{aligned} \mathrm{d}I = \Delta t \frac{\partial J^0}{\partial \mathbf{U}^0} \mathrm{d}\mathbf{U}^0 &+ \sum_{n=1}^{N} \left[\Delta t \cdot \frac{\partial J^n}{\partial \mathbf{U}^n} + \lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \mathbf{U}^n} \right] \mathrm{d}\mathbf{U}^n + \\ &+ \sum_{n=1}^{N} \left[\lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \mathbf{U}^{n-1}} \right] \mathrm{d}\mathbf{U}^{n-1} + \\ &+ \sum_{n=1}^{N} \left[\Delta t \cdot \frac{\partial J^n}{\partial \boldsymbol{\xi}} + \lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \boldsymbol{\xi}} \right] \mathrm{d}\boldsymbol{\xi}. \end{aligned} \tag{4.17}$$

Reorganise Eq. (4.17) to combine $\mathrm{d}\mathbf{U}^{n-1}$ and $\mathrm{d}\mathbf{U}^n$ terms

$$\begin{aligned} \mathrm{d}I = &\left[\Delta t \frac{\partial J^0}{\partial \mathbf{U}^0} + \lambda_1^T \frac{\partial \mathbf{R}^{*1}}{\partial \mathbf{U}^0} \right] \mathrm{d}\mathbf{U}^0 + \left[\Delta t \cdot \frac{\partial J^N}{\partial \mathbf{U}^N} + \lambda_N^T \frac{\partial \mathbf{R}^{*N}}{\partial \mathbf{U}^N} \right] \mathrm{d}\mathbf{U}^N + \\ &+ \sum_{n=1}^{N-1} \left[\Delta t \cdot \frac{\partial J^n}{\partial \mathbf{U}^n} + \lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \mathbf{U}^n} + \lambda_{n+1}^T \frac{\partial \mathbf{R}^{*n+1}}{\partial \mathbf{U}^n} \right] \mathrm{d}\mathbf{U}^n + \\ &+ \sum_{n=1}^{N} \left[\Delta t \cdot \frac{\partial J^n}{\partial \boldsymbol{\xi}} + \lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \boldsymbol{\xi}} \right] \mathrm{d}\boldsymbol{\xi}. \end{aligned} \tag{4.18}$$

This leaves a term that depends on $\mathrm{d}\mathbf{U}^0$. Per Eq. (4.14), these initial conditions only depend on the design variables. Therefore, the first term is

$$\left[\Delta t \frac{\partial J^0}{\partial \mathbf{U}^0} + \lambda_1^T \frac{\partial \mathbf{R}^{*1}}{\partial \mathbf{U}^0} \right] \mathrm{d}\mathbf{U}^0 = \left[\Delta t \frac{\partial J^0}{\partial \mathbf{U}^0} + \lambda_1^T \frac{\partial \mathbf{R}^{*1}}{\partial \mathbf{U}^0} \right] \frac{\mathrm{d}\mathbf{U}^0}{\mathrm{d}\boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi}. \tag{4.19}$$

The adjoint variables, λ_n^T, are defined such that they follow

$$\begin{cases} \lambda_N^T \dfrac{\partial \mathbf{R}^{*N}}{\partial \mathbf{U}^N} + \Delta t \cdot \dfrac{\partial J^N}{\partial \mathbf{U}^N} = 0 & n = N \\ \lambda_n^T \dfrac{\partial \mathbf{R}^{*n}}{\partial \mathbf{U}^n} + \Delta t \cdot \dfrac{\partial J^n}{\partial \mathbf{U}^n} + \lambda_{n+1}^T \dfrac{\partial \mathbf{R}^{*n+1}}{\partial \mathbf{U}^n} = 0 & n \in [1, N-1] \end{cases}. \tag{4.20}$$

This results in the elimination of all $\mathrm{d}\mathbf{U}^n$ terms in Eq. (4.18). The expression of the unsteady residual is shown in Eq. (4.13). Therefore, the partial derivative of $\mathbf{R}^*$ with respect to the conservative variables can be expressed as a function of the steady residual, $\mathbf{R}$, and the time step, Δt. Substitute this expression in Eq. (4.20)

$$\begin{cases} \lambda_N^T \left(\dfrac{1}{\Delta t} + \dfrac{\partial \mathbf{R}^N}{\partial \mathbf{U}^N} \right) + \Delta t \cdot \dfrac{\partial J^N}{\partial \mathbf{U}^N} = 0 & n = N \\ \lambda_n^T \left(\dfrac{1}{\Delta t} + \dfrac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n} \right) + \Delta t \cdot \dfrac{\partial J^n}{\partial \mathbf{U}^n} - \lambda_{n+1}^T \dfrac{1}{\Delta t} = 0 & n \in [1, N-1] \end{cases} . \tag{4.21}$$

Finally, the gradient of the augmented objective function with respect to the design variables is

$$\frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{\xi}} = \Delta t \cdot \frac{\partial J^0}{\partial \boldsymbol{\xi}} + \lambda_1^T \frac{\partial \mathbf{R}^{*1}}{\partial \mathbf{U}^0} \frac{\mathrm{d}\mathbf{U}^0}{\mathrm{d}\boldsymbol{\xi}} + \sum_{n=1}^{N} \left[\Delta t \cdot \frac{\partial J^n}{\partial \boldsymbol{\xi}} + \lambda_n^T \frac{\partial \mathbf{R}^{*n}}{\partial \boldsymbol{\xi}} \right], \tag{4.22}$$

with λ_n^T obtained from Eq. (4.21). Rumpfkeil and Zingg presented a similar derivation for a second-order implicit backward differentiation time-stepping approach [114].

Comparing the discretised form of the unsteady adjoint problem in Eq. (4.21) and that of the forward problem in Eq. (4.13) the change in sign of the residual is apparent. Furthermore, the value of the adjoint variables in time step n, λ_n, depends on the value of $\lambda_{n+1} \forall n < N$. Therefore, the adjoint solution is integrated backwards in time, from the last time step in which the objective function is evaluated to the first time step of the solution. This is consistent with the continuous-in-time approach.

4.1.3 Techniques for reducing the cost

There are two main concerns regarding the cost of unsteady adjoint solutions: the computational cost needed to calculate the adjoint variables and the data storage required for the direct solution. Thus, techniques for reducing both have been developed and applied.

Computational cost

Reducing the computational cost of unsteady adjoint solutions is important for practical applications of the method. A gradient-based optimisation algorithm requires a direct solution in order to obtain the value of the objective function and an adjoint solution in order to obtain the corresponding gradient. The optimisation process usually takes $\mathcal{O}(10-100)$ gradient evaluations. Any gain in speed will be multiplied by this value.

Nadarajah and Jameson presented some techniques for reducing the computational cost. Besides the usual full unsteady method, they used a steady adjoint approach with the unsteady flow solution, a steady adjoint with a time-averaged flow solution and multipoint design. The objective was to reduce the time-averaged drag coefficient of a pitching aerofoil at constant mean lift coefficient. While these techniques resulted in significant time savings and a reduction in the drag, the results were slightly different from those obtained by the full unsteady method [113]. In a periodic problem, if the frequency is known *a priori*, a limited number of cycles may be used.

Data storage

Since the adjoint solution depends on the direct solutions in future time steps, previous solutions have to be stored. If the mesh has many elements, the solution files will take up a large amount of storage space. In order to mitigate this problem, some algorithms have been developed.

One of the first families of methods used to reduce storage usage were checkpointing algorithms. These techniques first solve the forward problem only storing a subset of all the solutions. Then, the adjoint problem is solved starting from the last time iteration, backwards. However, the solutions not stored have to be recomputed. This increases the computational cost of the adjoint solution while reducing the data storage required [115–117].

Some of the methods that reduce computational cost also reduce storage requirements. However, they do not solve the unsteady adjoint equations, leading to a reduction of fidelity. One such technique is the steady adjoint using a time-averaged flow solution used by Nadarajah and Jameson [113].

More recently, Papadimitriou proposed a two-step approach to estimate the initial conditions for the adjoint problem. In this method, the unsteady flow calculations are integrated first but only the time-averaged solution is stored. Then, the adjoint equations are integrated backwards in time using the averaged solution. This gives an estimate of the initial values of the unsteady adjoint problem. Finally, the flow and adjoint equations are integrated forwards in time, simultaneously. This decreases the number of solutions that have to be stored on disk at the expense of approximately doubling the computational cost compared to a traditional unsteady adjoint method. However, the adjoint solution obtained is not exact [118].

4.2 Definition of unsteady objective functions

In unsteady problems, the objective function is defined in principle at all time steps. Depending on the specific problem, some definitions will make more sense. Examples include minimising the average drag coefficient or the maximum stress on a structure. The transient part before the problem converges in time might not need to be studied. In those cases, a window in which the objective function is defined can be used.

4.2.1 Averaging

One way to define the objective function is to average it through the whole time domain. The objective function is

$$J\left(\boldsymbol{\xi}\right) = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} f\left(t, \boldsymbol{\xi}\right) \mathrm{d}t, \tag{4.23}$$

where t_0 and t_f are the start and end times of the simulation. Its discretised form is

$$J\left(\boldsymbol{\xi}\right) = \frac{1}{N_f} \sum_{n=0}^{N_f} f^n, \tag{4.24}$$

where N_f is the total number of time steps and f^n is the value of function f at time step n.

Zhang et al. implemented this approach and used it in order to improve the flutter characteristics of the Isogai model. Their objective function was the damping obtained by using the Hilbert transform [92]. Mani and Mavriplis used this method in order to minimise the average drag of a pitching 2D aerofoil in transonic conditions under an average lift constraint [119]. Averaging over the whole time domain includes the transient part of the solution in the definition of the objective function.

4.2.2 Maximum

If instead of the average, the maximum value is important there are some methods similar to those described for constraints in Sec. 3.5. For example, Jacobson used the Kreisselmeier-Steinhauser (KS) function in time in order to keep the gust response of an aerofoil within bounds [120] and in time and space to prevent the structural failure of a pitching wing while minimising its weight [120, 121]. The function is defined as

$$\mathrm{KS}[K(\boldsymbol{\xi}, t), \rho] = K_{\max}(\boldsymbol{\xi}) + \frac{1}{\rho} \ln \int_0^{t_f} \left[e^{\rho(K(\boldsymbol{\xi},t) - K_{\max}(\boldsymbol{\xi}))} \right] \mathrm{d}t, \tag{4.25}$$

where $K_{\max}$ is the maximum value of the constraint in time. Unlike the discrete constraint aggregation in Sec. 3.5.2, the inequality $\mathrm{KS}[K(\boldsymbol{\xi}, t), \rho] > K_{\max}$ does not hold. However, as ρ increases the KS function approaches the maximum [120].

4.2.3 Windowing

In most unsteady cases there is a transient part before time convergence is reached. Often, the value of the objective function during this transient is not of interest for optimisation. A time window that only includes the steady state can be defined.

In problems in which the base frequency of the problem is known *a priori*, such as the oscillations of helicopter rotor blades or turbomachinery, a limited number of cycles can be kept. The start time of the window is then $t_{start} = t_f - n \cdot T$ for n cycles of period T. If using an averaged objective function, it can be defined as

$$J(\boldsymbol{\xi}) = \frac{1}{n \cdot T} \int_{t_{start}}^{t_f} f(t, \boldsymbol{\xi}) \, \mathrm{d}t, \tag{4.26}$$

where f is the chosen objective function. Discretise the equation in time

$$J(\boldsymbol{\xi}) = \frac{1}{n \cdot N_T} \sum_{i=N_{start}}^{N_f} f^i, \tag{4.27}$$

where N_f is the total number of time steps, N_T is the number of time steps in one time period, $N_{start} = N_f - n \cdot N_T$ is the first iteration of the window and f^n is the value of function f at time step n. Note that even though outside the window $\partial f^i / \partial \mathbf{U}^i = 0$, the adjoint term in Eq. (4.22) is non-zero.

This approach has been used for the optimisation of many time-periodic problems. Zhou et al. used windowing in order to reduce the noise produced by a pitching aerofoil in transonic flow conditions. They eliminated the first two motion cycles out of 10 in the calculation [72]. Nielsen and Diskin targeted a specific value of torque produced by a wind turbine. They also maximised the thrust produced by a flapping wing. In both cases, they only kept the last period of the simulation [122].

If the frequency of the motion is unknown and the effect of the transient part is not desired, a fixed-length window may be used. The process is similar to the one previously described for known-frequency cases but with T being a free parameter. Therefore, its influence in the solution obtained has to be studied.

Windowing was applied by Srinath and Mittal to a low-Reynolds aerofoil using the Navier-Stokes equations. One of the objective functions was to match an average lift coefficient of $c_l = 0.75$. In those conditions, vortex shedding appeared. They

compared three different window lengths: a very short window, a window that corresponded to approximately one vortex-shedding cycle and one that corresponded to approximately three vortex-shedding cycles. They found that using the second and third windows led to similar designs, while the optimisation algorithm diverged when using the shortest window [123]. Kiviaho et al. wrote the calculation of the flutter point as an optimisation problem. They used the matrix pencil method in a time window including approximately five oscillations in order to find the minimum damping of the modes of a two degree of freedom aerofoil [124].

Mishra et al. simulated one revolution of a helicopter rotor. They defined the objective function window as the last sixth of the time interval studied. Their objective was to reduce the torque coefficient while keeping the thrust produced by the rotor approximately constant. They used a penalty method in order to achieve this. The optimised geometry resulted in a better performance even after time convergence had been reached beyond the optimisation window [125].

4.3 Fluid-structure interaction harmonic balance adjoint method

While the adjoint method is faster when the number of design variables is higher than the number of objective functions, adjoint time marching simulations are still very costly. Furthermore, even with checkpointing they require storing the solution at many time steps. For periodic problems the harmonic balance can be used. Besides the reduction in computational cost, it only requires storing $2 \cdot N_h + 1$ solutions for N_h harmonics.

Another advantage of the harmonic balance method is that defining the objective function can be more straightforward. The amplitudes and average of magnitudes of interest are obtained directly from the solver, without needing to use windowing techniques. In some periodic problems the frequency depends on the design parameters, further complicating windowing. Furthermore, the harmonic balance approach is not affected by changes in time convergence. If the rate of time convergence is adversely affected during the optimisation process, a time-marching simulation may not reach the desired level at the original end point. Conversely, the problem may converge in time faster, leading to superfluous computations. The harmonic balance technique, on the other hand, ensures a certain level of convergence of the solution.

Choi et al. applied the discrete adjoint harmonic balance to a helicopter rotor in forward flight. In that case, they used a computational fluid dynamics code to obtain the periodic aerodynamic loads around one of the rotor's blades [126]. More recently, Prasad et al. presented an FSI harmonic balance method for unknown-frequency

problems. In order to solve the direct problem, they used a fixed-energy approach. Then, they derived the adjoint equations corresponding to their approach [7].

Apply the adjoint transformation to the formulation in Eq. (2.69):

$$\begin{bmatrix} \frac{\partial \mathcal{S}}{\partial \mathbf{u}}^T & 0 & \frac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \\ \frac{\partial \mathcal{S}}{\partial \omega}^T & \frac{\partial \mathcal{F}}{\partial \omega}^T & \frac{\partial \mathcal{M}}{\partial \omega}^T \\ \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^T & \frac{\partial \mathcal{F}}{\partial \mathbf{w}}^T & 0 \\ \frac{\partial \mathcal{S}}{\partial \mathbf{z}}^T & \frac{\partial \mathcal{F}}{\partial \mathbf{z}}^T & \frac{\partial \mathcal{M}}{\partial \mathbf{z}}^T \end{bmatrix} \begin{bmatrix} \lambda_u \\ \lambda_w \\ \lambda_z \end{bmatrix} = \begin{bmatrix} \frac{\partial J}{\partial \mathbf{u}} \\ \frac{\partial J}{\partial \omega} \\ \frac{\partial J}{\partial \mathbf{w}} \\ \frac{\partial J}{\partial \mathbf{z}} \end{bmatrix}. \tag{4.28}$$

Compared to the steady system of equations in Eq. (3.38), there are four extra terms: the dependence on the fundamental frequency of the objective function, the solid, the fluid and the mesh solvers. Equation (4.28) can be rearranged as

$$\begin{cases} \frac{\partial \mathcal{S}}{\partial \mathbf{u}}^T \lambda_u &= \frac{\partial J}{\partial \mathbf{u}} - \frac{\partial \mathcal{M}}{\partial \mathbf{u}}^T \lambda_z \\ \frac{\partial \mathcal{S}}{\partial \omega}^T \lambda_u &= \frac{\partial J}{\partial \omega} - \frac{\partial \mathcal{F}}{\partial \omega}^T \lambda_w - \frac{\partial \mathcal{M}}{\partial \omega}^T \lambda_z \\ \frac{\partial \mathcal{F}}{\partial \mathbf{w}}^T \lambda_w &= \frac{\partial J}{\partial \mathbf{w}} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^T \lambda_u \\ \frac{\partial \mathcal{M}}{\partial \mathbf{z}}^T \lambda_z &= \frac{\partial J}{\partial \mathbf{z}} - \frac{\partial \mathcal{F}}{\partial \mathbf{z}}^T \lambda_w - \frac{\partial \mathcal{M}}{\partial \mathbf{z}}^T \lambda_u \end{cases}, \tag{4.29}$$

showing that there is an additional equation to be solved. Each adjoint equation corresponds to one unknown of the direct problem, while each adjoint variable corresponds to one direct equation. If the combined phase-fixing and frequency iteration technique from Sec. 2.5.3 is used to solve the direct problem, the equation corresponding to the fixed degree of freedom is not needed. Therefore, the number of equations in the direct problem matches that of the adjoint problem.

4.3.1 Derivation of the adjoint equations of the time-domain harmonic balance

The direct time-domain harmonic balance (TDHB) approach is presented in Sec. 2.4. It assumes a periodic solution which is expanded using a Fourier series. The TDHB uses $2 \cdot N_h + 1$ time instances equispaced along a cycle to approximate the solution.

The generic expression of this problem is in Eq. (2.61). Apply the TDHB to a generic solver,

$$\mathcal{R}\left(\mathbf{U}^*, \omega, \boldsymbol{\xi}\right) := \mathbf{D}\mathbf{U}^* - \mathbf{R}^*\left(\mathbf{U}^*, \boldsymbol{\xi}\right) = \mathbf{0}, \tag{4.30}$$

where $\mathbf{D}$ is the time-derivative matrix described in Eq. (2.58), $\mathbf{U}^*$ is the vector of conservative variables of the problem at each time instance and $\mathbf{R}^*$ is the steady residual at each time instance. Recall that $\mathbf{D}$ is proportional to the base frequency and differentiate Eq. (4.30) with respect to ω:

$$\frac{\partial \mathcal{R}}{\partial \omega}^T = \left(\frac{1}{\omega}\mathbf{D}\mathbf{U}^*\right)^T. \tag{4.31}$$

The derivative with respect to the conservative variables is

$$\frac{\partial \mathcal{R}}{\partial \mathbf{U}^*}^T = \mathbf{D}^T - \left(\frac{\partial \mathbf{R}^*}{\partial \mathbf{U}^*}\right)^T. \tag{4.32}$$

The adjoint equation for problem $\mathcal{R}\left(\mathbf{U}^*, \omega, \boldsymbol{\xi}\right)$ is

$$\left(\frac{\partial \mathcal{R}}{\partial \mathbf{U}^*}\right)^T \lambda - \left(\frac{\partial J}{\partial \mathbf{U}^*}\right)^T = \mathbf{0}. \tag{4.33}$$

This expression can be obtained from the steady result in Eq. (3.19). Substituting the harmonic balance derivative in Eq. (4.32) into the adjoint equation, the expression for the time-domain harmonic balance adjoint variables, λ, is

$$\mathbf{D}^T\lambda - \left(\frac{\partial \mathbf{R}^*}{\partial \mathbf{U}^*}\right)^T \lambda - \left(\frac{\partial J}{\partial \mathbf{U}^*}\right)^T = \mathbf{0} \tag{4.34}$$

for all conservative variables. The frequency equation obtained by substituting Eq. (4.31) into the second equation in Eq. (4.29) is

$$\left(\mathbf{D}\mathbf{u}^*\right)^T \lambda_u = \omega\frac{\partial J}{\partial \omega} - \left(\mathbf{D}\mathbf{w}^*\right)^T \lambda_w - \left(\mathbf{D}\mathbf{z}^*\right)^T \lambda_z, \tag{4.35}$$

where $\mathbf{u}^*, \mathbf{w}^*, \mathbf{z}^*$ are the vectors of solid displacements, fluid conservative variables and mesh positions at every time instance.

The time-domain harmonic balance equation is given in Eq. (4.30) for a time-domain problem following the differential equation

$$\frac{\partial}{\partial t}\mathbf{U}(t) - \mathbf{R}\left(\mathbf{U}(t), t\right) = 0. \tag{4.36}$$

The adjoint differential equation corresponding to the direct problem in Eq. (4.36) is

$$\frac{\partial \lambda}{\partial t} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \lambda + \frac{\partial j}{\partial \mathbf{U}}^T = 0. \tag{4.37}$$

Apply the TDHB to the adjoint problem

$$\mathbf{D}\lambda + \left(\frac{\partial \mathbf{R}^*}{\partial \mathbf{U}^*}\right)^T \lambda + \frac{\partial J}{\partial \mathbf{U}^*}^T = 0. \tag{4.38}$$

As per Eq. (2.60), matrix $\mathbf{D}$ is antisymmetric. This means that the matrix used in Eq. (4.34) is $\mathbf{D}^T = -\mathbf{D}$. Substituting this into Eq. (4.34) and comparing it to Eq. (4.38), the adjoint solution obtained would be the same. Therefore, applying the time-domain harmonic balance method to the time-domain adjoint equation is equivalent to applying the adjoint method to the direct time-domain harmonic balance equations under the conditions described in Sec. 2.4.2.

4.3.2 Derivation of the adjoint equations of a frequency-domain rigid body motion integrator

Besides the time-domain harmonic balance, a frequency-domain approach is also used for the structural solver in the present work. In particular, a frequency-domain rigid body motion integrator with pitching and plunging degrees of freedom has been implemented. The derivation of the equations for the direct solver are described in further detail in Appendix C. The state-space equations of the problem are

$$\mathbf{x} = \begin{bmatrix} \dot{\alpha} \\ \dot{h} \\ \alpha \\ h \end{bmatrix}; \quad \begin{bmatrix} m & S_\alpha & 0 & 0 \\ S_\alpha & I_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} c_h & 0 & k_h & 0 \\ 0 & c_\alpha & 0 & k_\alpha \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \mathbf{x} - \begin{bmatrix} -L \\ M \\ 0 \\ 0 \end{bmatrix} = 0, \tag{4.39}$$

where $S_\alpha = m \cdot (x_f - x_{CG})$ is the dimensional static imbalance, m is the mass of the aerofoil, c_h is the plunge damping, k_h is the plunge stiffness, I_α is the moment of inertia around the flexural axis, c_α is the pitch damping, k_α is the pitch stiffness, h and α are the plunge and pitch displacements. The plunge is positive downwards and the pitch is positive clockwise. This convention is shown in Fig. 2.9. The corresponding frequency-domain equations are

$$\left(\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}\right)\hat{\mathbf{x}} - \mathbf{E}\mathbf{F}^* = 0, \tag{4.40}$$

where $\hat{\mathbf{M}}$ is the frequency-domain mass matrix, $\mathbf{A}$ is the frequency-domain time-derivative matrix, $\hat{\mathbf{K}}$ is the stiffness matrix, $\mathbf{E}$ is the discrete Fourier transform matrix described in Eq. (2.53), $\mathbf{F}^*$ is the vector of loads at each time instance and $\hat{\mathbf{x}}$ is the frequency-domain vector of displacements. Substituting the values of Eq. (4.40) into the first equation in Eq. (4.29), the adjoint equations of the solid solver are

$$\left(\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}} - \frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{x}}}\right)^T \hat{\lambda} = \frac{\partial J}{\partial \hat{\mathbf{x}}} - \frac{\partial \mathcal{M}}{\partial \hat{\mathbf{x}}}^T \hat{\lambda}_z, \tag{4.41}$$

where $\hat{\lambda}$ is the vector of frequency-domain adjoint variables, $\hat{\mathbf{F}} = \mathbf{E}\mathbf{F}^*$ is the frequency-domain vector of loads, J is the objective function and $-\frac{\partial \mathcal{M}}{\partial \hat{\mathbf{x}}}^T \hat{\lambda}_z$ is the mesh component of the gradient.

The derivative of Eq. (4.40) with respect to the frequency was obtained in order to implement the combined phase-fixing/frequency-iteration technique described in Sec. 2.5.3. It is

$$\frac{\partial \mathcal{S}}{\partial \omega} = \frac{1}{\omega}\left(\hat{\mathbf{M}}\mathbf{A}\hat{\mathbf{x}}\right). \tag{4.42}$$

Substitute this derivative into the second equation in Eq. (4.29)

$$\frac{1}{\omega}\left(\hat{\mathbf{M}}\mathbf{A}\hat{\mathbf{x}}\right)^T \hat{\lambda} = \frac{\partial J}{\partial \omega} - \frac{\partial \mathcal{F}}{\partial \omega}^T \lambda_w - \frac{\partial \mathcal{M}}{\partial \omega}^T \lambda_z. \tag{4.43}$$

The right-hand side includes the fluid dependence $-\frac{\partial \mathcal{F}}{\partial \omega}^T \lambda_w$ and the mesh dependence terms $-\frac{\partial \mathcal{M}}{\partial \omega}^T \lambda_z$.

Equation (4.41) includes an explicit, linear dependence that appears as the $\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}$ term. Furthermore, there is a dependence of the load with respect to the displacement, $\frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{x}}}$. While the lift depends only on the sum of the fluid forces at each node, the moment depends on the relative location of the nodes themselves. The moment at each time instance is given by

$$M = \sum_{n=0}^{N_n} F_y^n \cdot (x_n - x_c) + F_x^n \cdot (y_n - y_c), \tag{4.44}$$

where N_n is the number of nodes at the boundary, F^n is the force applied on node n, x_n and y_n are the x and y coordinates of node n and x_c and y_c are those of the centre of rotation. The derivative can be obtained by applying the chain rule

$$\frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{x}}} = \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{F}^*}\frac{\partial \mathbf{F}^*}{\partial \mathbf{x}^*}\frac{\partial \mathbf{x}^*}{\partial \hat{\mathbf{x}}} = \mathbf{E}\frac{\partial \mathbf{F}^*}{\partial \mathbf{x}^*}\mathbf{E}^{-1} \tag{4.45}$$

where $\mathbf{E}^{-1}$ is the inverse Fourier transform matrix described in Eq. (2.57). The gradient of the moment with respect to the pitching angle in the time domain can be easily obtained from the expression of the moment in Eq. (4.44) and that of the node coordinates in Eq. (C.8)

$$\frac{\partial M}{\partial \alpha} = \sum_{n=0}^{N_n} F_x^n \cdot (x_n - x_c) + F_y^n \cdot (y_n - y_c). \tag{4.46}$$

The other gradients $\left(\frac{\partial L}{\partial h}, \frac{\partial L}{\partial \alpha}, \frac{\partial M}{\partial h}\right)$ are 0. These terms are then substituted into Eq. (4.45). They have to be calculated once for each optimisation iteration. They are then substituted into the left-hand side of Eq. (4.41). A derivation of the partial gradients of objective functions of interest and the total gradients with respect to shape and structural design variables is presented in Appendix D.

Coupling terms

The adjoint structural solution needs the fluid and mesh components of the gradients with respect to the frequency and the boundary displacements in order to be coupled. This stands in contrast with the direct solution, in which the structural solver provides the frequency and boundary displacements to the fluid and mesh solvers. The coupling source term on the adjoint frequency equation is represented by the last two terms of the right-hand side of Eq. (4.43). The boundary conditions of the adjoint problem are represented by the last term of Eq. (4.41) for the structural displacements' gradients.

The gradients to be provided to the fluid solver can be calculated from the adjoint variables. Recall the expression of the fluid source term in Eq. (4.29)

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\mathbf{w}}\right|_{\mathcal{S}} = \frac{\partial J}{\partial \mathbf{w}} - \frac{\partial \mathcal{S}}{\partial \mathbf{w}}^{T} \lambda_u, \tag{4.47}$$

where $\mathbf{w}$ are the fluid conservative variables, $\mathcal{S}$ is the solid residual and λ_u is the vector of solid adjoint variables. In the present setup, the variables transferred from the fluid solver to the solid solver are the nodal forces at each time instance. However, the solid solver, $\mathcal{S}$, is written in the frequency domain. Therefore, Eq. (4.47) is in this case

$$\left.\frac{\mathrm{d}J}{\mathrm{d}\mathbf{F}^*}\right|_{\mathcal{S}} = \frac{\partial J}{\partial \mathbf{F}^*} - \frac{\partial \mathcal{S}}{\partial \mathbf{F}^*}^{T} \hat{\lambda}_u, \tag{4.48}$$

where $\mathbf{F}^*$ is the vector of nodal forces at each time instance and $\hat{\lambda}_u$ is the vector of frequency-domain adjoint variables of the solid solver. The calculation of the derivative $\frac{\partial \mathcal{S}}{\partial \mathbf{F}^*}$ is easier in the time domain. Isolate the term dependent on the frequency-domain adjoint variables in Eq. (4.48) and operate

$$\frac{\partial \mathcal{S}}{\partial \mathbf{F}^*}^{T} \hat{\lambda}_u = \left(\frac{\partial \mathcal{S}}{\partial \mathbf{S}^*}\frac{\partial \mathbf{S}^*}{\partial \mathbf{F}^*}\right)^{T} \hat{\lambda}_u = \frac{\partial \mathbf{S}^*}{\partial \mathbf{F}^*}^{T} \mathbf{E}^T \hat{\lambda}_u, \tag{4.49}$$

where $\mathbf{S}^*$ is the time-domain equivalent of the solid solver at each time instance. The derivative of the time-domain equations with respect to the nodal forces, $\frac{\partial \mathbf{S}^*}{\partial \mathbf{F}^*}$, can be calculated from the expressions in Appendix C. The transformation of the adjoint variables from the frequency domain to the time domain is

$$\lambda = \mathbf{E}^T \hat{\lambda}. \tag{4.50}$$

Since the objective function generally does not directly depend on the nodal forces, the first term on the right-hand side of Eq. (4.48) is 0. Therefore, the

gradient with respect to the force at boundary node n is given by

$$\frac{\mathrm{d}J}{\mathrm{d}F_x^n} = \lambda_\alpha \cdot (y_n - y_c) \tag{4.51}$$

$$\frac{\mathrm{d}J}{\mathrm{d}F_y^n} = -\lambda_h - \lambda_\alpha \cdot (x_n - x_c)\,, \tag{4.52}$$

where λ_α is the time-domain adjoint variable from the pitch equation and λ_h that of the plunge equation. The x axis nodal forces only modify the moment, while the y axis nodal forces affect both the lift and moment. The dependence of the moment on the nodal forces can be calculated by differentiating Eq. (4.44) with respect to them. As per Eq. (4.39), the sign of the lift is negative. Note that the sign of the second term on the right-hand side of Eq. (4.47) is negative.

4.3.3 Coupling of the adjoint FSI HB partitioned approach

The coupling is similar to the partitioned direct harmonic balance approach described in Sec. 2.6.1 and to the partitioned steady adjoint described in Sec. 3.6.2. The process is shown as a flowchart in Fig. 4.1.

First, a direct solution must be obtained. The direct fluid and structural solutions are loaded. Then, the structural displacements and frequency are imposed to the fluid solver. The fluid mesh is deformed based on the boundary displacements. Then, the fluid and mesh adjoint solvers calculate a solution without the solid adjoint coupling terms. The gradient of the objective functions with respect to the frequency and the boundary displacement is obtained. The gradient with respect to boundary displacement is then interpolated as described in Sec. 3.6.1 and imposed on the solid boundary. Since as described in Sec. 2.5 the direct solid solver calculates the frequency, the adjoint solid solver requires the corresponding gradient. The adjoint solid solver then obtains its adjoint variables, calculates the gradients with respect to the boundary forces and transfers them to the coupler. For the following iteration, the coupler interpolates the gradients from the solid to the fluid boundary as described in Sec. 3.6.1. These are then used by the fluid solver. The fluid adjoint solution is then a coupled solution.

This process is repeated until either convergence or the maximum number of FSI iterations is reached. The convergence criterion chosen is based on the boundary conditions imposed in the fluid solver. For the harmonic balance adjoint solver, as was the case in the steady coupling described in Sec. 3.6.2, it is the magnitude of the change of the vector of gradients with respect to the fluid loads. If convergence is not reached, the solution should be rejected.

For the rigid body motion integrator, the solid adjoint solver is written explicitly as described in Sec. 4.3.2. Therefore, there is no need to iterate. After the first solid

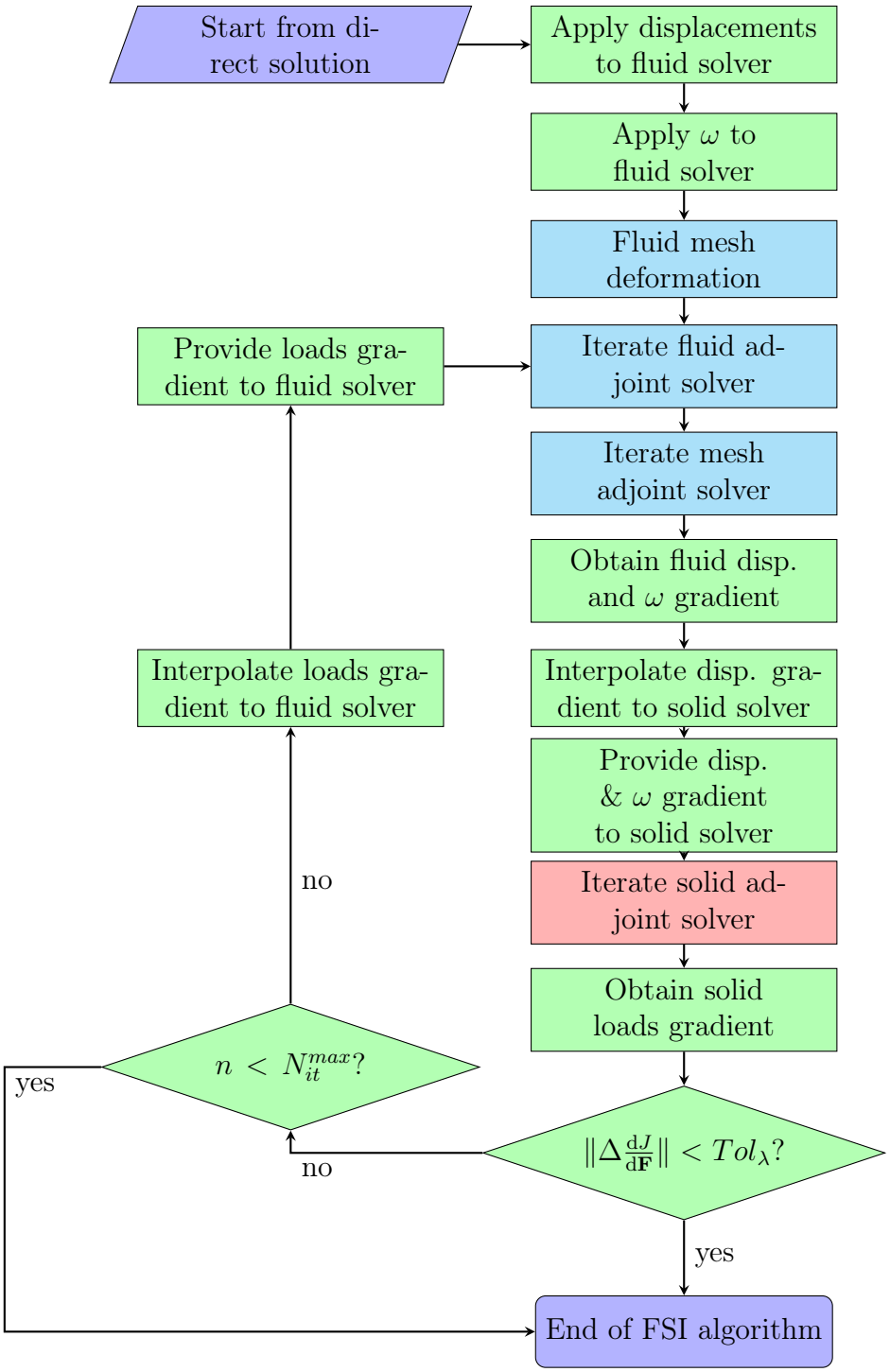


Figure 4.1: Flowchart describing the harmonic balance adjoint fluid-structure interaction algorithm. In light blue, the actions carried out by the adjoint fluid solver; in red, those by the adjoint structural solver; in green, those by the coupler.

adjoint iteration the solution of the structural sub-problem is converged. However, if a structural code using an implicit calculation of the adjoint variables were to be used, there would be a need to iterate.

4.4 Design variables for shape optimisation

Shape optimisation is the discipline that deals with modifying the shape of bodies in order to maximise or minimise an objective function. An appropriate definition of the shape-related design variables is important in order to define the design space to be studied.

Masters et al. described two broad families of design variables: constructive and deformative [5]. Constructive design variables define a surface from a set of geometric characteristics. Some examples of this approach are the NACA 4- and 5-digit series aerofoils [127, 128]. They are two families of aerofoils of varying thickness, camber and maximum camber position. Another constructive approach is the PARSEC parametrisation method described by Sobieczky. It defines a sixth-order polynomial based on the values of 11 parameters: leading edge radius, upper- and lower-crest location and curvature and a set of trailing edge geometric parameters [129]. Deformative design variables, on the other hand, modify the shape of an already-existing surface [5].

The design variables can define either the surface or the whole volume. In the first case, either a mesh deformation or a remeshing step needs to be included. Remeshing steps are often very expensive, especially around complex, three-dimensional geometries.

One example of the use of aerodynamic shape optimisation is matching a given pressure distribution around an aerofoil. The objective of these tests is to recover the correct aerofoil shape. By comparing the final and reference shapes, the shape parametrisation method's application in a computational fluid dynamics context can be verified and its limits can be tested. Other common objective functions include lift to drag ratio maximisation.

4.4.1 Node position

The simplest design variables that one can consider are the location of the nodes at the boundary. However, any modification of these is, by its very nature, discontinuous. This results in high-frequency components that can lead to ill-conditioned problems [130]. Furthermore, the design variables are applied to the grid being studied, not to the underlying geometry. If the same geometry needs to be used in a different solver (structural, higher- or lower-fidelity...) the changes would need

to be interpolated. Jameson applied a smoothing procedure to the objective function for reverse design to ensure its continuity even in the presence of transonic flows with shocks [96]. Mani and Mavriplis used this method for various inverse design problems, including unsteady load- and pressure distribution-matching of 2D aerofoils [119].

4.4.2 Hicks-Henne bumps

Hicks and Henne presented, among other functions, a family of bumps used to minimise the drag in a supercritical wing [131]. Unlike the node positions, these functions are continuous and smooth in the open interval $]0,1[$. They do not include high-frequency components. The bumps only modify the direction perpendicular to the chord. These functions are known as Hicks-Henne bumps.

The generic equation for them is [132]

$$y(x) = y_0(x) + \sum_{i=0}^{N} \xi_i \cdot f_i(x) \tag{4.53}$$

$$f_i(x) = \begin{cases} 0, & x = 0 \\ \left[\sin\left(\pi \cdot x^{\frac{\log 0.5}{\log t_1}}\right)\right]^{t_2}, & x \in]0,1] \end{cases}, \tag{4.54}$$

where y is the non-dimensional vertical position, y_0 is the original value of y, x is the non-dimensional position along the chord, N is the number of bumps, ξ is the value of the design variable, f_i is the ith bump, $t_1 \in]0,1[$ is a parameter representing the value of x where $f_i(x)$ is maximum and $t_2 > 0$ is a parameter controlling the width of the bump. The maximum value of the bump function is $f_i(t_1) = 1$. For the second parameter, Masters et al. used a value of $t_2 = 1$ after performing a parametric study on an aerofoil database [5]. Hicks and Henne originally proposed $t_2 = 3$ [131].

These bumps were first defined for two-dimensional profiles. Therefore, in order to modify a 3D wing some sort of interpolation between profiles is required. One issue that limits their use is that, for $t_2 > 1$, they cannot modify the thickness or curvature very close to the trailing edge. This problem occurs because the left limit of the derivative of f_i at $x = 1$ is 0 [92]. Furthermore, the bump functions are not orthogonal [130].

Hicks-Henne bumps have been used in order to minimise the average drag around an aerofoil in transonic flow [100]. They have also been used to minimise the torque-to-thrust ratio of a helicopter rotor blade. Since the case was three-dimensional, Choi et al. defined the bumps in nine sections along the span and interpolated between the sections [126].

4.4.3 Basis splines

Another method is to parametrise the surface using basis splines, also called B-splines. Since this technique is widely used in computer-aided design (CAD) packages, it provides a direct link between the CAD model and the optimised design. Unlike Hicks-Henne bumps or using the node positions as design variables, this method does not modify a base shape. Instead, the original shape is parametrised and then those parameters are used as design variables. For m control points the positions are a function of a parameter, $t \in [t_0, t_m]$

$$\mathbf{x}(t) = \sum_{i=0}^{n-1} \mathbf{P}_i N_{i,k}(t), \tag{4.55}$$

where $n = m - k$, $\mathbf{P}_i$ is the position of the ith control point and $N_{i,k}$ is the B-spline of order $k < m$, given by

$$N_{i,0}(t) = \begin{cases} 1 & t \in [t_i, t_{i+1}] \\ 0 & t \notin [t_i, t_{i+1}] \end{cases} \tag{4.56}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{(t_{i+k+1} - t)}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t), \tag{4.57}$$

where t_i is the ith control point's position in the parametrised space. The values are chosen such that $t_i \leq t_{i+1}$. The order of the basis spline, k, controls how local the changes are. If k increases, the influence of a given control point extends farther away. If $k = n$, the B-splines are called Bézier curves. The parameter t can be bounded between 0 and 1, in which case the Bézier curve is a linear combination of Bernstein basis polynomials. For aerofoil parametrisation, usually the chordwise coordinate of the control points is kept constant [5]. When the number of control points is large, B-splines can lead to high-frequency oscillations of the surface [130, 133].

One extension of the B-spline technique is the non-uniform rational basis spline (NURBS) method. This approach allows a non-uniform spacing of the control points in t and changes the basis functions used. The basis functions are rational and given by

$$R_{i,k}(t) = \frac{N_{i,k}(t) \cdot \omega_i}{\sum_{j=0}^{n-1} N_{j,k}(t) \cdot \omega_j}, \tag{4.58}$$

where ω_i is the ith point's weight [134]. The interpolated positions are

$$\mathbf{x}(t) = \sum_{i=0}^{n-1} \mathbf{P}_i R_{i,k}(t). \tag{4.59}$$

Lepine et al. performed an inverse design, matching a target pressure distribution around aerofoils in transonic conditions. They used 13 control points, from which they selected 11 design parameters [134]. Srinath and Mittal used the NURBS method with 10 design variables and three fixed points to parametrise a 2D aerofoil in low-Reynolds number conditions [123].

4.4.4 Free-form deformation boxes

The free-form deformation (FFD) approach was proposed by Sederberg and Parry for deforming solids in computer graphics [135]. The method creates an overset box over the original, undeformed geometry. The overset box, which is a parallelepiped, has a set of $l \times m \times n$ equispaced control points. Their displacement deforms the box which in turn deforms the geometry. Express the coordinates x, y, z as a function of the box's system of coordinates s, t, u

$$\mathbf{x} = \mathbf{x}_0 + s\mathbf{S} + t\mathbf{T} + u\mathbf{U}, \tag{4.60}$$

where $\mathbf{x} = [x\ y\ z]^T$, $\mathbf{x}_0$ is the physical coordinate of the $(0, 0, 0)$ point in the box's system of coordinates and $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$ are the vectors defining the parallelepiped's edges. The transformed coordinates are

$$\begin{bmatrix} s \\ t \\ u \end{bmatrix} = [\mathbf{S}\ \mathbf{T}\ \mathbf{U}]^{-1} (\mathbf{x} - \mathbf{x}_0) . \tag{4.61}$$

The transformed coordinates of the box are bounded to $[0, 1]$, which means that the interpolation can be calculated as a linear combination of Bernstein basis polynomials. The jth Bernstein basis polynomial of order i is

$$B_{i,j}(x) = \binom{j}{i} (1 - x)^{j-i} x^i. \tag{4.62}$$

The parametrisation of the deformation is then

$$\mathbf{x} = \sum_{i=0}^{l} B_{i,l}(s) \left[\sum_{j=0}^{m} B_{j,m}(t) \left[\sum_{k=0}^{n} B_{k,n}(u) \mathbf{P}_{ijk} \right] \right], \tag{4.63}$$

where $\mathbf{P}_{ijk}$ is the vector of coordinates of the respective control point. As is the case in the B-spline method in Eq. (4.55), the control point coordinates become the coefficients of the basis polynomials. These coordinates are the design variables.

Lamousin and Waggenspack extended the FFD approach from parallelepipeds to generic NURBS-based volumes. These were used in order to more easily relate

the shape of the deformed box and that of the surface in computer graphics. Unlike in the case of the parallelepiped, the position of the points in the box's system of coordinates has to be found iteratively [136]. Samareh used an FFD technique employing a bidimensional surface defined by NURBS instead of a three-dimensional box. This method was applied to maximise the lift-to-drag ratio of a morphing aircraft [137]. The MASSOUD shape parametrisation tool also provides a relation between the FFD variables and geometry parameters of interest for aerostructural optimisation [138, 139].

Within aerodynamic optimisation, the FFD approach has been used to minimise the average drag at constant average lift of a wing [100]. In aeroelasticity, MASSOUD has been used for the minimisation of take-off weight of a wing under structural failure and lift constraints for steady and unsteady problems [120, 121, 140].

4.5 Verification of unsteady coupled gradients

The partitioned HB FSI adjoint algorithm described in Sec. 4.3.3 is verified by applying it to a transonic limit-cycle oscillation (LCO) test case. This test case was used in Sec. 2.8 to verify the direct harmonic balance approach with unknown frequency. It is a 2D NACA 64A010 aerofoil that can move in the pitch and plunge degrees of freedom. The reduced velocity of the case lies beyond the flutter point, which leads to an oscillation. However, the nonlinear behaviour of the flow limits its amplitude. The flow is modelled using the Euler equations, which are inviscid. One of the high-amplitude setups in Sec. 2.8, Case 9, is used to perform the verification. The fluid freestream conditions are given in Table 2.6 and the structural parameters in Table 2.2. The direct harmonic balance simulation uses one harmonic, which requires 3 time instances. The flow field is solved using SU2 [8], which was extended for adjoint harmonic balance calculations for this project as detailed in Sec. B.2.1 of Appendix B.

Two different objective functions are defined: the squared norm of the pitching amplitude and the mean drag coefficient. The first objective function is computed by the structural solver, while the second one is computed by the fluid solver. For each objective function, the gradients obtained using the adjoint method are then compared to those obtained by means of a finite differencing scheme.

The original aerofoil is plotted in blue in Fig. 4.2. In order to verify the gradients of the two objective functions, a deformative method is used. Twelve Hicks-Henne bumps on each side of the aerofoil are defined as design variables. The chordwise

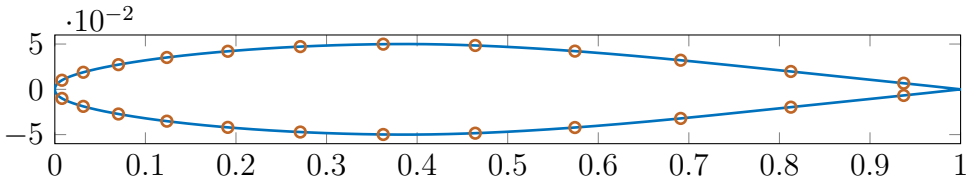


Figure 4.2: NACA 64A010 aerofoil, in blue, with the peak of the Hicks-Henne design variables marked as orange circles

location of the ith bump's peak is obtained using a half-cosine distribution

$$x_i = 1 - \cos\left(\frac{\pi \cdot i}{2 \cdot n + 1}\right), \tag{4.64}$$

where n is the number of bumps in each side. This distribution leads to a higher concentration of bumps close to the leading edge and wider spacing at the trailing edge. The locations of the peaks, which correspond to parameter t_1 in Eq. (4.54), appear as orange circles in Fig. 4.2. Parameter t_2 was set to 3, as was the case in the original definition of the bumps. The shape of the twelve bumps on each side of the aerofoil is shown in Fig. 4.3. If the value of the design variable is positive, the boundary of the aerofoil is deformed away from the chord. Note that close to the trailing edge the value of the bumps is always very close to 0, even for the bump with $t_1 = 0.94$.

The Hicks-Henne bumps and their derivatives were implemented in the rigid body motion integrator for this work. The bumps were chosen because of their ease of implementation, extensive use in the literature for 2D profiles and smoothness.

Direct harmonic balance simulations were carried out to verify the gradients using central finite differences for both objective functions. The step length for the bump functions was $\Delta\xi = \pm 1 \times 10^{-4} \cdot c$. Economon performed a convergence analysis of the step length of Hicks-Henne bumps in a transonic test case and found little difference between $1 \times 10^{-6} \cdot c$ and $1 \times 10^{-4} \cdot c$ [101].

4.5.1 Movement amplitude

One possible objective function is the movement amplitude. Generally, a large LCO amplitude is not desirable, but a small one might be tolerated under certain conditions. Therefore, reducing its value can be useful. For this test case the objective function is defined as $J = |\alpha_1|^2$, where α_1 is the amplitude of the first harmonic of the pitching motion. Because of the use of the harmonic balance method, this value is directly available from the structural solver without needing to calculate a maxi-

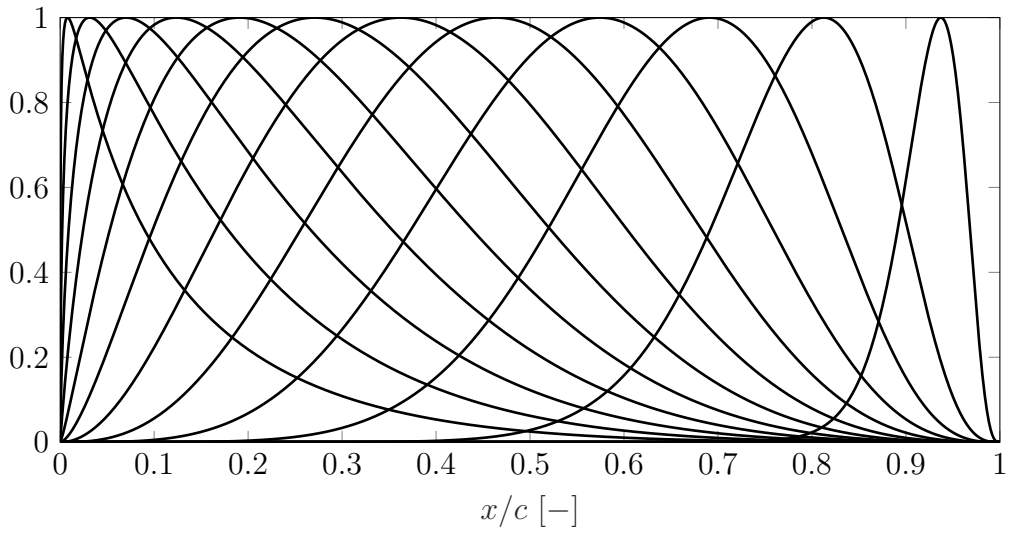


Figure 4.3: Shape of the Hicks-Henne bumps used as design variables

mum. The main objective of this case is to verify the implementation of gradients of a structural objective function within the coupled adjoint FSI approach.

The Hicks-Henne bumps that are used in order to modify the shape of the aerofoil deform the surface in a direction normal to the chord. Figure 4.4 shows the value of the gradients of the objective function calculated using the adjoint harmonic balance technique with respect to each boundary node's y coordinate at zero pitch. The expression used to calculate this value is shown in Appendix D in Eq. (D.16). The upper surface appears in blue while the lower surface appears in orange. In principle, the two curves should match because the test case is symmetric. However, as explained in Sec. 2.8.6, the direct harmonic balance solution is not symmetric because of the reduced number of harmonics. As a result of this asymmetry, the two curves do not match, but they follow similar trends.

From Fig. 4.4, the changes in the shape of the aerofoil needed to reduce the pitching amplitude of this LCO can be deduced. It should be thinner close to the leading edge until approximately 20% of the chord. At that point, the gradient turns negative and an increase in thickness would be preferred. Then, at around 60% of the chord and until the trailing edge, the sign of the gradient changes again. Even though the magnitude of the gradient is small, in this region the aerofoil should be thinner. Finally, it should be noted that the trailing edge has strong oscillations. Using the node coordinates as design variables could lead to a saw-tooth shape of the optimised aerofoil close to the trailing edge. The gradients have a larger magnitude in the first half of the chord, which justifies the higher density of design variables in

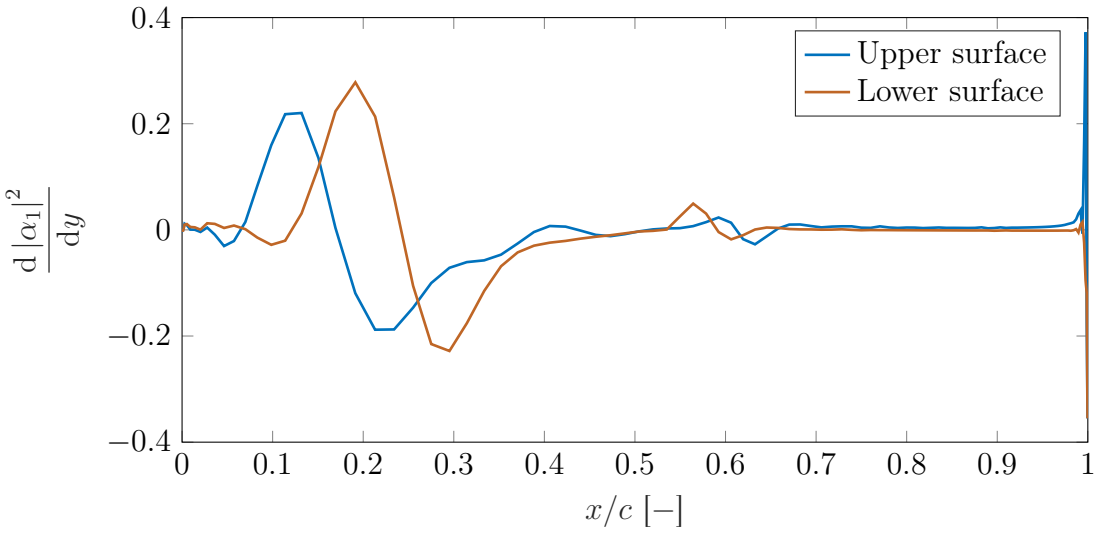


Figure 4.4: Sensitivity of the pitching amplitude with respect to each boundary node's y coordinate

that region.

Verification of gradients

Figure 4.5 shows the gradient of the amplitude with respect to the Hicks-Henne bumps. The centres of these bumps are shown in Fig. 4.2. Figure 4.5 compares the gradients obtained by using the coupled FSI HB adjoint method described in Sec. 4.3.3 and those obtained by central finite differences. The adjoint results appear as blue circles and the finite differences values appear as orange crosses. The differences are small.

The trends of the gradients on the upper and lower sides of the aerofoil are similar but the values are different. Close to the leading edge the values are positive, which means that increasing the aerofoil's thickness in that region would increase the amplitude. Then, in the middle region they are negative. Finally, the last design variable, the centre of which is very close to the trailing edge, has a positive value. This matches the behaviour observed in Fig. 4.4. The oscillations close to the trailing edge are smoothed out by the shape of the bump.

Table 4.1 compares the gradients of the amplitude objective function with respect to various structural parameters and the Mach number obtained by using finite differences and by using the present adjoint method. The steps used in the finite differences scheme for each design parameter appear in the last column of the table. They were chosen so that the change in the objective function was small enough

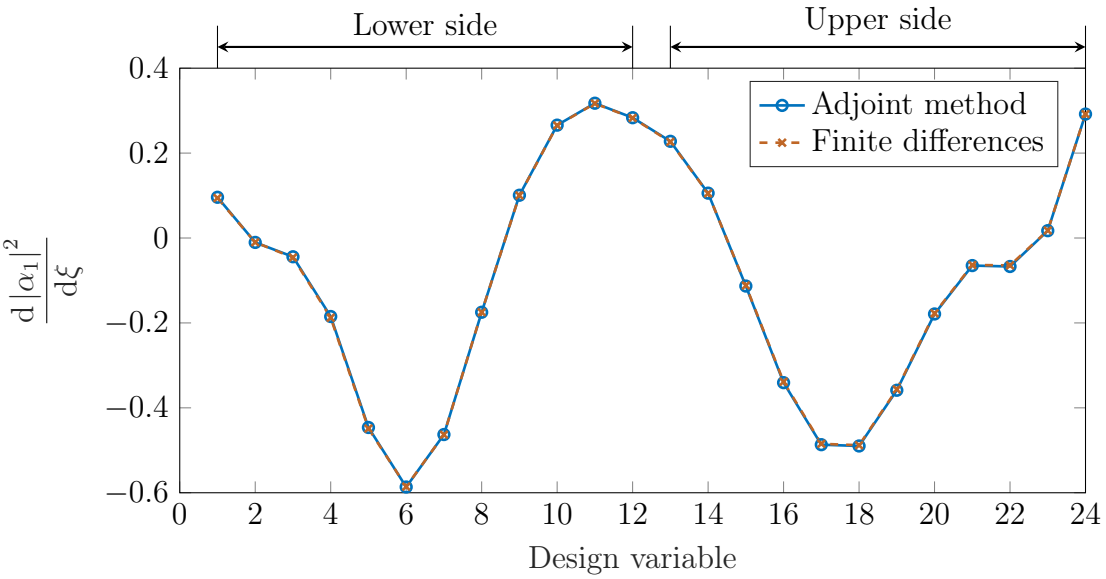


Figure 4.5: Comparison of adjoint and central finite difference gradients of the amplitude objective function with respect to the surface design variables

to eliminate third-order effects but still measurable. The results obtained using the adjoint method are very similar to those obtained using finite differences. The maximum difference is under 0.6% in the case of the moment of inertia, I_α. The other gradients are closer, in all cases the adjoint method producing a slightly larger value of the magnitude of the gradient.

Increases in mass and torsional stiffness lead to a decrease in LCO amplitude, while increases in plunging stiffness and moment of inertia increase the LCO amplitude. In the first case, the modes' natural frequencies move away from each other, which in turn increases the flutter speed. Therefore, the current freestream conditions would be closer to the bifurcation point so that, in the case of a supercritical Hopf bifurcation, the amplitude would decrease. The plunge damping decreases the amplitude, while the pitch damping increases it. This could be due to the fluttering mode being plunge-dominant.

In the present setup, the freestream velocity and dynamic pressure increase with increasing Mach number. Despite this, Table 4.1 shows that the amplitude decreases as the Mach number increases. This phenomenon occurs because for this case at $\mathrm{M}_\infty = 0.8$ the flutter speed increases with the Mach number.

Parameter	Gradients		Parameter values	
	Adjoint	Finite differences	ξ_o	$\Delta\xi$
k_h [N m^{-1}]	$1.8\mathbf{34} \times 10^{-7}$	$1.8\mathbf{25} \times 10^{-7}$	107 559.2	100
k_α [N m rad^{-1}]	$-3.8\mathbf{35} \times 10^{-7}$	$-3.8\mathbf{20} \times 10^{-7}$	80 669.4	100
c_h [kg s]	$-2.5\mathbf{43} \times 10^{-5}$	$-2.5\mathbf{29} \times 10^{-5}$	0	0.1
c_α [kg m^2 s]	$1.7\mathbf{86} \times 10^{-5}$	$1.7\mathbf{78} \times 10^{-5}$	0	0.1
m [kg]	$-2.1\mathbf{76} \times 10^{-4}$	$-2.1\mathbf{68} \times 10^{-4}$	50.33	0.1
I_α [kg m^2]	$1.5\mathbf{37} \times 10^{-3}$	$1.5\mathbf{28} \times 10^{-3}$	9.437	0.01
M_∞ [–]	$-3.6\mathbf{84} \times 10^{-1}$	$-3.6\mathbf{74} \times 10^{-1}$	0.8	0.001

Table 4.1: Verification of gradients of amplitude

Comparison of computational cost

Since the main objective of the present work is to make higher-fidelity FSI optimisation more feasible, it is crucial to compare the computational cost incurred when using the coupled adjoint method with that of the direct method.

The direct solution's numerical parameters are those used in Sec. 2.8.6. The case required using underrelaxation in order to converge. Therefore, the relaxation parameter was set to $\Omega = 0.7$, which led to a converged solution. This same value was used in order to obtain the adjoint solution.

Some numerical parameters differed between the fluid direct and adjoint solvers. The adjoint CFL number was set to 2 instead of 3. Consequently, the minimum number of iterations was increased from 2000 to 5000 in order to ensure a sufficient reduction in the residual. The relaxation parameter of the adjoint calculation was also set to $\Omega = 0.7$.

Table 4.2 compares the CPU time used by the original direct setup described in Sec. 2.8 (t_{dir}) and the adjoint calculation (t_{adj}) for this particular test case. The adjoint solution required 18 FSI iterations to reach convergence, which is close to the 20 required by the direct solution. In both cases mesh mapping is immediate because of the matching meshes method used. The adjoint solution required a much shorter time to perform the mesh deformation because the boundary only moves once, at the beginning of the adjoint calculation. Therefore, the mesh was already deformed at later FSI adjoint iterations. Communication of fluid and solid gradients took a similar time in the two solutions.

In both cases the most time-consuming part of the calculation was the fluid solver: 99.8% of the CPU time for the direct solver and 99.99% for the adjoint solver. The adjoint solver reduced significantly this cost by approximately 43%,

despite requiring a larger number of fluid iterations. This reduction drove the overall reduction in computational cost. The solid solver required a minimal time in both cases. The solid adjoint solver was approximately four times faster than the direct one.

Step	t_{dir} [s]	t_{adj} [s]
Mesh mapping	0.00	0.00
Mesh deformation	26.30	0.08
Communication	0.15	0.13
Fluid solver	14 511.43	8340.57
Solid solver	0.08	0.02
Total	14 540.49	8341.27

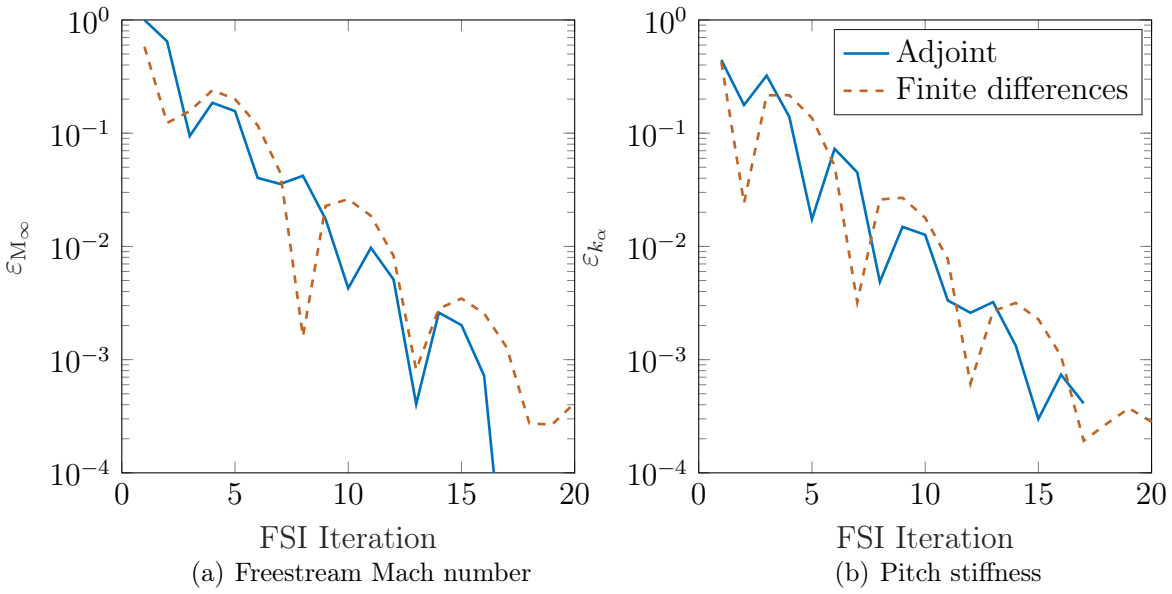
Table 4.2: Comparison of computational cost between direct and adjoint harmonic balance methods

Figure 4.6 compares the rate of convergence of gradients between the adjoint method, a blue solid line, and central finite differences, an orange dashed line. Its y axis is logarithmic in order to better show the rate of convergence. The error in the gradients is defined, for design parameter ξ, as the absolute value of the relative difference between the gradient at the corresponding FSI iteration and the converged value, normalised by the latter. The value of this gradient at FSI iteration i is expressed as $\left.\frac{\mathrm{d}|\alpha_1|^2}{\mathrm{d}\xi}\right|_i$. Then, for a simulation with n total iterations, the value of the error, ε_ξ, at FSI iteration i is

$$\varepsilon_{\xi,i} = \left| \frac{\left.\dfrac{\mathrm{d}\,|\alpha_1|^2}{\mathrm{d}\xi}\right|_i}{\left.\dfrac{\mathrm{d}\,|\alpha_1|^2}{\mathrm{d}\xi}\right|_n} - 1 \right|. \tag{4.65}$$

Two different parameters are selected, one structural and one fluid. They are the freestream Mach number, in Fig. 4.6(a), and the pitch stiffness, in Fig. 4.6(b). The slope of gradient convergence is similar for both the adjoint and finite differences methods for these parameters, with similar behaviour observed in others. Therefore, a similar number of FSI iterations would be required in order to reach the same level of convergence.

It should be noted that the adjoint approach needs one direct solution and one adjoint calculation per objective function. This results in a total of $1+2$ calculations



(a) Freestream Mach number (b) Pitch stiffness

Figure 4.6: Comparison of convergence rate between adjoint and central finite differences gradients of the amplitude objective function with respect to two parameters

for two objective functions. Therefore, the total CPU time spent to obtain the gradients using the proposed adjoint approach would be 31.2×10^3 s. The central finite differences technique needs $2n + 1$ direct simulations for n design variables. In the present case, with $n = 24 + 6 + 1 = 31$, it would be 63 direct simulations. Assuming that all direct simulations take the same time, it would take 901.5×10^3 s of CPU time to calculate the gradients using finite differences. This is approximately 30 times higher than the adjoint approach.

4.5.2 Inviscid mean drag coefficient

The gradients of a structural objective function with respect to shape design variables have been verified in the previous section. A fluid objective function is introduced in the present section: the mean drag coefficient of the LCO, $J = c_{d,0}$. The mean corresponds to the zeroth harmonic of the drag coefficient, which can be obtained from the sum of the drag coefficients in each of the time instances divided by the number of time instances. This is exactly equal to the value obtained using the discrete Fourier transform. The calculated drag coefficient does not take into account viscous effects because the Euler equations, which are used to solve the problem, are inviscid. The main objective of this section is to verify the gradients of a fluid objective function within the coupled adjoint framework. Combining these

results with those described in Sec. 4.5.1, the complete framework would be verified.

Verification of gradients

Figure 4.7 shows the gradient of the mean drag coefficient with respect to each of the design variables in Fig. 4.2. It compares the adjoint method results, which appear as blue circles, and central finite differences predictions, which appear as orange crosses.

Comparing these gradients to those obtained for the amplitude objective function in Fig. 4.5, they follow a broadly similar trend. In these flow conditions, the effect of the oscillation amplitude on the inviscid mean drag is very important.

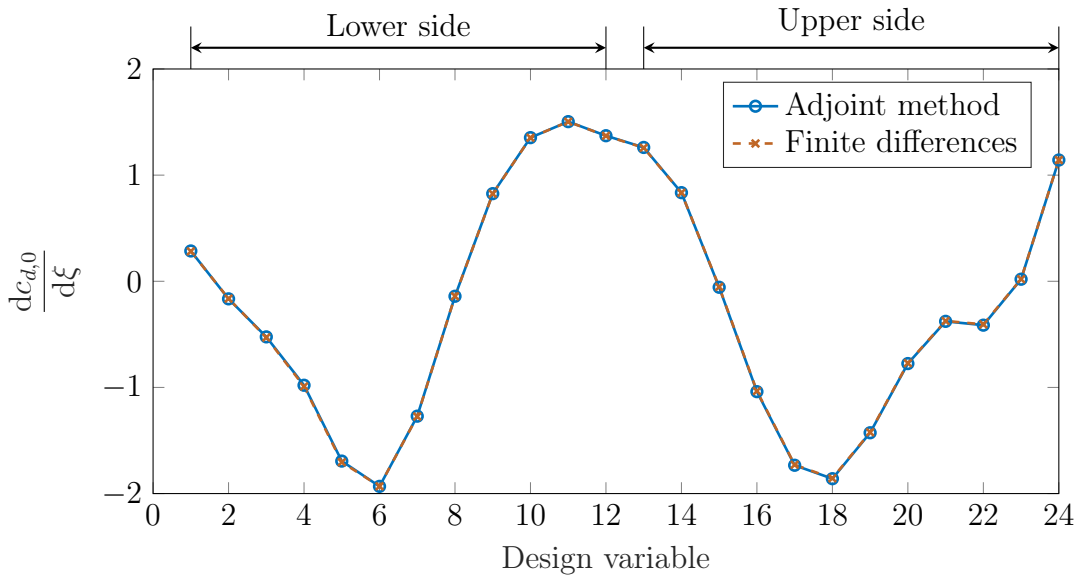


Figure 4.7: Comparison of adjoint and central finite difference gradients of the inviscid mean drag coefficient with respect to the surface design variables

The gradients of the inviscid mean drag coefficient with respect to the same parameters shown in the previous section are shown in Table 4.3. The gradients obtained by the adjoint and finite differences methods are compared. The same finite differences simulations were used for the amplitude objective function in Table 4.1 and for the mean drag objective function. The results follow broadly the same trends as those of the amplitude objective function, with the adjoint method predicting a slightly larger value of the magnitude of the gradients. The maximum difference between the predictions of the two methods is in the case of the moment of inertia as well, though in this case the discrepancy is under 0.35%. As expected, the signs

match those obtained for the amplitude objective function, albeit with a larger magnitude.

Parameter	Gradients		Parameter values	
	Adjoint	Finite differences	ξ_o	$\Delta\xi$
k_h [N m^{-1}]	$8.6\mathbf{50} \times 10^{-7}$	$8.6\mathbf{24} \times 10^{-7}$	107 559.2	100
k_α [N m rad^{-1}]	$-1.83\mathbf{4} \times 10^{-6}$	$-1.83\mathbf{0} \times 10^{-6}$	80 669.4	100
c_h [kg s]	$-1.2\mathbf{63} \times 10^{-4}$	$-1.2\mathbf{59} \times 10^{-4}$	0	0.1
c_α [kg m^2 s]	$7.\mathbf{609} \times 10^{-5}$	$7.\mathbf{590} \times 10^{-5}$	0	0.1
m [kg]	$-9.\mathbf{217} \times 10^{-4}$	$-9.\mathbf{199} \times 10^{-4}$	50.33	0.1
I_α [kg m^2]	$7.3\mathbf{55} \times 10^{-3}$	$7.3\mathbf{30} \times 10^{-3}$	9.437	0.01
M_∞ [−]	$-1.19\mathbf{92}$	$-1.19\mathbf{78}$	0.8	0.001

Table 4.3: Verification of gradients of inviscid mean drag coefficient

Evolution of gradients with the coupling

The evolution of $\mathrm{d}c_{d,0}/\mathrm{dM}_\infty$ as the FSI solution converges is shown in Fig. 4.8. In the first iteration, before the influence of the structure is included in the computation, the value is positive. That is, if the movement amplitudes and frequency were kept constant, the mean drag coefficient would increase with the Mach number. This behaviour is expected since a higher freestream Mach number leads to a stronger shock and, thus, to a higher shock drag.

However, as the simulation progresses the gradient changes sign. If the Mach number increases, the mean drag decreases. This is because for this test case the amplitude of the limit-cycle oscillation decreases with increasing Mach number. Since under these conditions the mean drag increases with the movement amplitude, the drag decreases as well. This phenomenon counteracts the positive contribution of the increased strength of the shock.

4.6 Summary

A partitioned technique that combines the harmonic balance and adjoint methods to obtain gradients for the optimisation of FSI problems has been proposed. The approach can be applied to problems with an *a priori* unknown frequency. In order to validate the technique, it has been applied to a 2D limit-cycle oscillation test case.

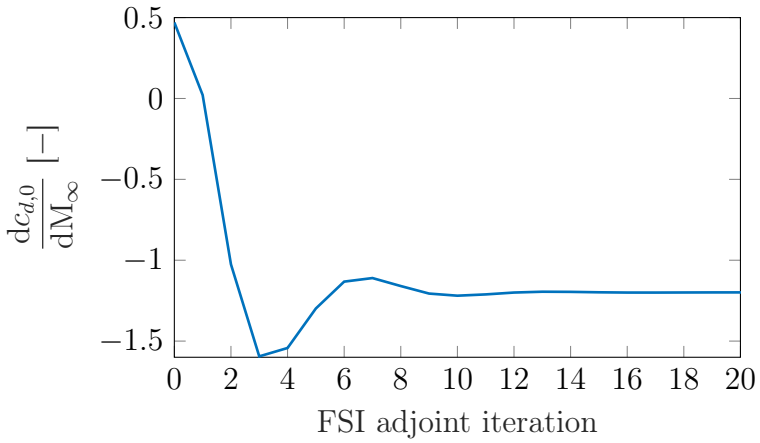


Figure 4.8: Evolution of the gradient of the inviscid drag coefficient with respect to the Mach number with FSI iteration

Two objective functions have been defined: a structural and a fluid objective function. The gradients of these objective functions with respect to shape, structural and fluid design parameters have been obtained. They have been compared to those obtained using central finite differences. Both objective functions show a very good match of the gradients, independently of the kind of design parameter.

The proposed approach shows several advantages compared to traditional time-marching methods: it only requires storing a very small number of flow solutions, it is less costly and many objective functions of interest, such as motion amplitude, are very easy to define. Since one harmonic is used to solve the direct problem, only 3 time instances have to be saved to disk. The computational cost of one harmonic balance adjoint calculation is lower than that of one harmonic balance direct calculation. For a case with 2 objective functions and 31 design variables, the adjoint technique is approximately 30 times faster than central finite differences.

Chapter 5

Optimisation of time-periodic fluid-structure interaction problems

A harmonic balance-based algorithm to obtain gradient vectors using the adjoint method has been described and verified in Chapter 4. The technique has been developed to optimise time-periodic fluid-structure interaction (FSI) problems in which the frequency is unknown. The results of the optimisation of four FSI test cases using this algorithm are shown in the present chapter.

The test cases act on a symmetric NACA 64A010 aerofoil in transonic conditions that undergoes a limit-cycle oscillation (LCO). The aerofoil is allowed to move in the pitching and plunging degrees of freedom restrained by linear springs. The setup is shown in Fig. 2.9. This case is the same one used to verify the gradients obtained using the proposed adjoint harmonic balance method in Sec. 4.5. It corresponds to Case 9, described in Sec. 2.8. The structural parameters are given in Table 2.2 and the flow conditions appear in Table 2.6. The original amplitude of the pitching motion was of $0.0735\,\text{rad} \simeq 4.21°$.

As was the case in the previous chapter, the solvers used are SU2 and the rigid body motion integrator. The flow is solved using the Euler equations, which are inviscid. The design variables are 24 Hicks-Henne bumps, 12 on the upper side and 12 on the lower side. One harmonic is used. The upper and lower bumps are allowed to take different values, which lets the aerofoil break its symmetry. They are the design variables used for verifying the adjoint gradients. The location of the peaks of the bumps is shown in orange in Fig. 4.2. As explained in Sec. 4.5, the Hicks-Henne bumps and their derivatives were implemented during the course of this project in the rigid body motion integrator, which also calculates the boundary deformation resulting from the bumps. Then, this deformation was imposed as a displacement of the boundary on the fluid solver.

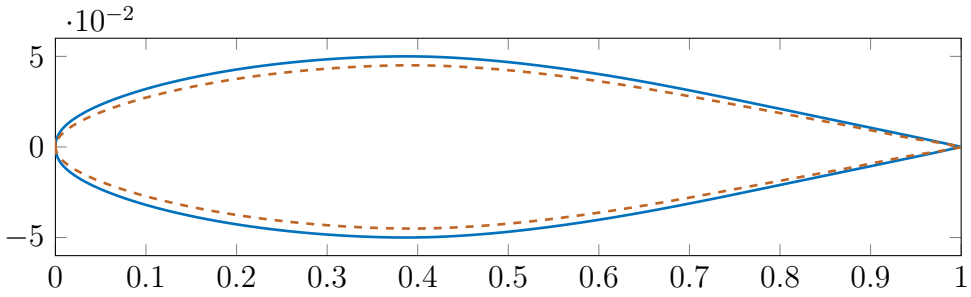


Figure 5.1: NACA 64A010 aerofoil, in blue, with the thinnest aerofoil within the design space in dashed orange

The amplitude for all the shape design variables, which is normalised by the chord, is bounded such that $\Delta\xi \in [-1.0 \times 10^{-3}, 1.0 \times 10^{-3}]$. The original aerofoil, in blue, and the thinnest aerofoil possible with these bounded amplitudes, in dashed orange, are plotted in Fig. 5.1. The original aerofoil has $\boldsymbol{\xi} = \mathbf{0}$, that is, the amplitude of all the bumps is 0. The thinnest aerofoil in the design space has the design variables take the value $\xi_i = -1.0 \times 10^{-3}\ \forall i \in [0, 23]$. The figure's scale on the y axis is exaggerated in order to better show the difference between the two aerofoils.

The present chapter introduces four optimisation test cases: two unconstrained and two constrained. The first case is an unconstrained minimisation of the amplitude of the LCO. The second case uses a fluid objective function to minimise the inviscid mean drag. Since the first two optimisation cases increase the inviscid drag coefficient at $\alpha = 0$, a constrained minimisation of the amplitude is performed. Then, a final case that maximises energy dissipation on the plunging degree of freedom's damper is included. This last optimisation case also includes a constraint on the pitching amplitude of the aerofoil.

5.1 Minimisation of LCO amplitude by shape optimisation

Large amplitudes of limit-cycle oscillations can be destructive. In those cases, reducing the amplitude of the LCO is important. The first optimisation test case uses the gradients obtained in Sec. 4.5.1, scaled by the original value of the objective

function. The optimisation problem is

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & J(\boldsymbol{\xi}) = \frac{|\alpha_1|^2}{|\alpha_{1,o}|^2}, \\ \text{subject to} \quad & \xi_i \in \left[-1.0 \times 10^{-3}, 1.0 \times 10^{-3}\right] \quad \forall i \in [0, 23], \end{aligned} \tag{5.1}$$

where $\boldsymbol{\xi}$ is the vector of design variables, ξ_i is the value of the ith design variable, $|\alpha_1|$ is the norm of the first harmonic's pitching amplitude and $\alpha_{1,o}$ is the value of said pitching amplitude for the original aerofoil. The squared norm was used to prevent a change in sign of the pitching amplitude leading to a maximisation. Since the sine amplitude was set to 0 as described in Sec. 2.8, the norm is equal to the absolute value of the cosine amplitude $|\alpha_1| = |\alpha_{1,c}|$. The objective function is then

$$J(\boldsymbol{\xi}) = \frac{\alpha_{1,c}^2}{|\alpha_{1,o}|^2}. \tag{5.2}$$

This greatly simplifies the calculation of derivatives. It is an unconstrained optimisation problem with bounded design variables. Therefore, the SciPy[1] implementation of a limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, L-BFGS-B [141, 142] is used.

5.1.1 Optimisation process

The optimisation process consisted of four iterations. After the fourth iteration, the aerofoil's flutter speed increased beyond the imposed freestream conditions. Therefore, the pitch amplitude and its gradients became 0. The path of reduction of the amplitude is shown in Fig. 5.2.

The resulting aerofoil's camber line and thickness distribution are shown in Fig. 5.3. The camber line is in Fig. 5.3(a) and the thickness distribution is in Fig. 5.3(b). The NACA 64A010 aerofoil used as a starting point is in blue and the optimised one is in dashed orange. As expected from the gradients in Sec. 4.5.1, the optimised aerofoil is thinner close to the leading and trailing edges and thicker in the middle. Its maximum thickness increases from 10% to 10.7% and the position of this maximum thickness moves downstream. Interestingly, the optimised aerofoil is slightly cambered with a reflex camber line. The maximum camber is of approximately 0.1%. The camber line appears because the results obtained by the harmonic balance method are not symmetric and the design variables can break the symmetry of the original aerofoil. The crossing point of the camber line is around 46% of the chord.

[1] `https://scipy.org/`

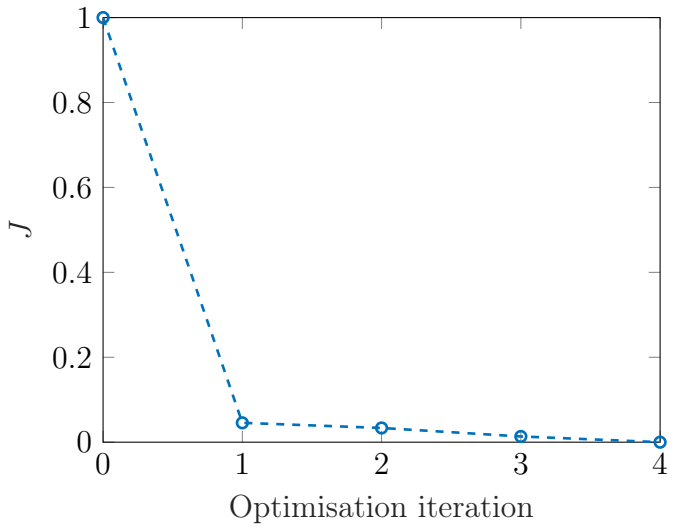


Figure 5.2: Evolution of the amplitude objective function, J, in blue circles

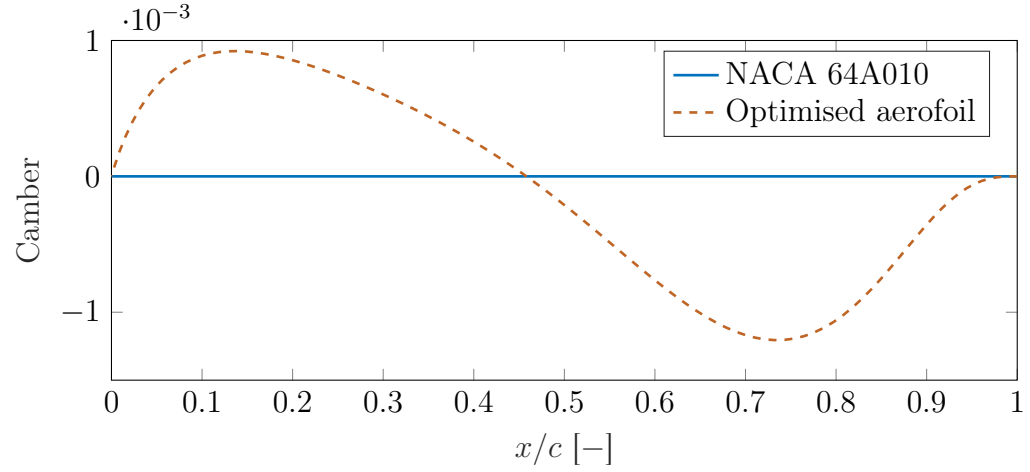
While using more harmonics results in better solutions, one harmonic is enough for subcritical and zero-amplitude cases. The nonlinear behaviour is either eliminated or significantly reduced, respectively, in such setups. Therefore, the asymmetry introduced by the harmonic balance method in these cases should be relatively small.

5.1.2 Steady results

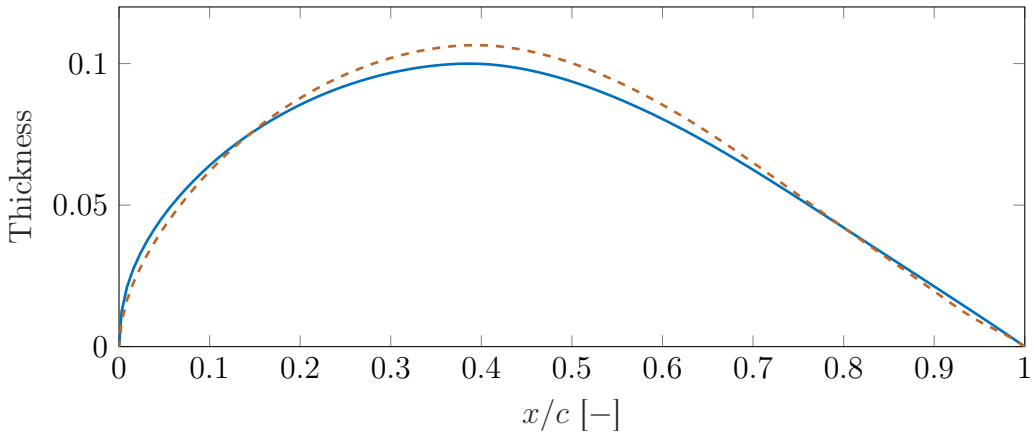
Figure 5.4 compares the steady pressure coefficient (C_p) distribution over the original, in blue, and optimised, in orange, aerofoils at $\alpha = 0$. Since the optimised aerofoil is asymmetric, the upper side is represented as a dashed line and the lower side is a dotted line. A steady CFD simulation using SU2 was used for both aerofoils. In order to generate the deformed mesh for the optimised aerofoil, the design capabilities of SU2 were used.

The optimised aerofoil has a stronger shock that appears farther downstream. In the original aerofoil it occurred at 52% of the chord, while in the optimised one it occurs at 56% and 58% of the chord on the upper and lower sides, respectively. The stronger shocks result in a larger value of the drag coefficient. The influence of the camber can also be observed, with slight differences in the pressure distribution between the upper and lower surface of the aerofoil.

This modified pressure distribution leads to a non-zero lift coefficient at $\alpha = 0$ of -0.015. Furthermore, the inviscid drag coefficient at $\alpha = 0$ increases from 0.00270



(a) Camber line



(b) Thickness distribution

Figure 5.3: Comparison of the original NACA 64A010 aerofoil and the minimum amplitude aerofoil

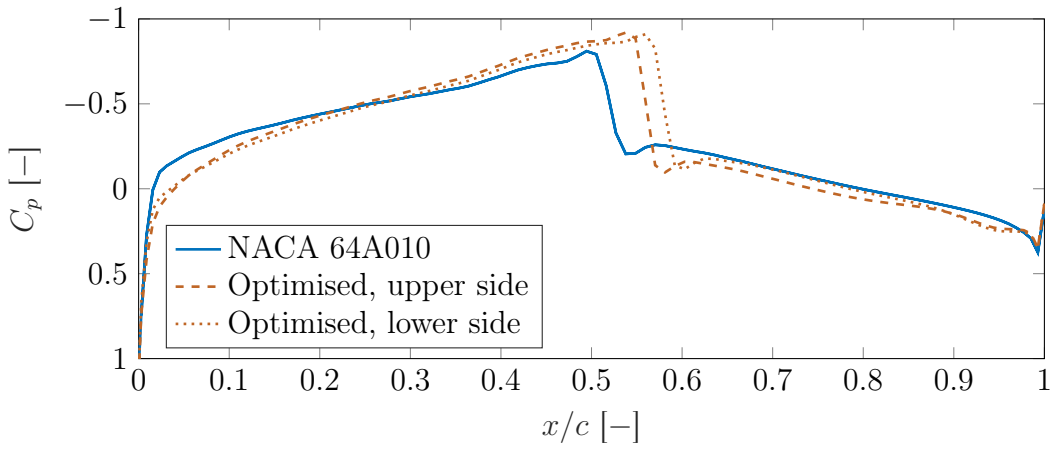


Figure 5.4: Comparison of the steady pressure coefficient distribution around the original NACA 64A010 aerofoil and the minimum amplitude aerofoil at $\alpha = 0$

to 0.00516, a relative increase of $\sim 91\%$.

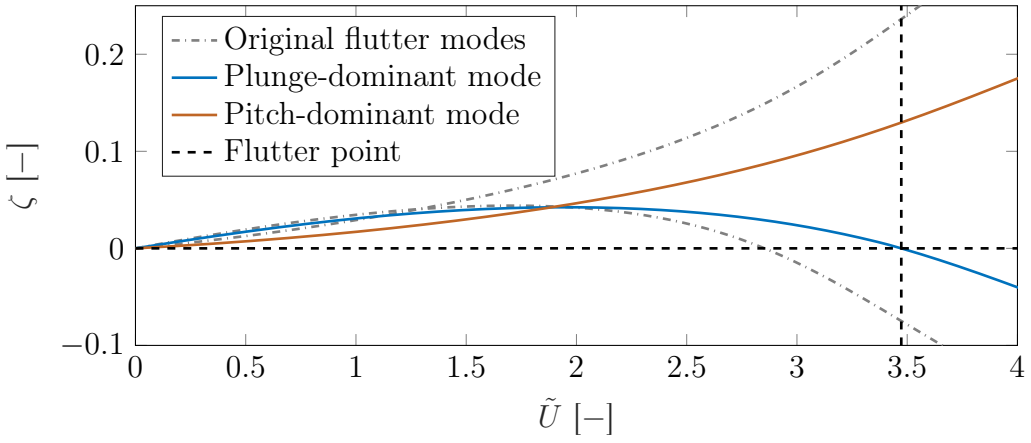
As shown in Sec. 4.5.1, the pitching amplitude is reduced for this inviscid test case as the Mach number increases. By increasing the thickness of the aerofoil, a similar behaviour to the one expected when increasing the Mach number is obtained: the shockwave also moves downstream and is strengthened. This could explain why the amplitude minimisation process results in a thicker aerofoil.

5.1.3 Flutter point

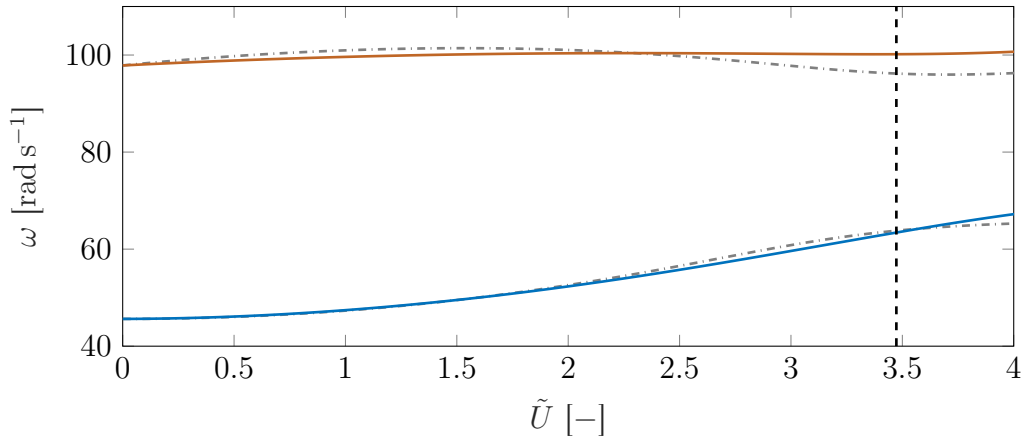
As previously explained, the aerofoil's movement amplitude was eliminated at the end of the optimisation procedure. This implies that the flutter point lies at a higher freestream velocity than the one studied. In order to confirm this, the flutter process described in Sec. 2.8 is applied to the new aerofoil. The deformed mesh obtained for the steady calculations is used for the flutter calculations.

At the end of the last unsteady simulation, a mean pitch of $\alpha_0 = 0.0024\,\text{rad} \simeq 0.14°$ is recovered. While this mean pitch depends on the dynamic pressure, for simplicity it is kept constant during the flutter calculation.

A freestream velocity sweep at a constant Mach $\text{M}_\infty = 0.8$ is performed. The resulting damping and frequency curves are shown in Fig. 5.5(a) and Fig. 5.5(b), respectively. Both figures show the pitch-dominant mode in orange and the plunge-dominant mode in blue. The flutter point appears as a dashed black line, showing both the flutter speed and the $\zeta = 0$ line. The damping and frequency curves calculated for the original aerofoil are shown as dash-dotted grey lines.



(a) Damping



(b) Frequency

Figure 5.5: Results of the flutter analysis for the minimum amplitude aerofoil

The flutter point for the optimised aerofoil lies at a reduced airspeed of $\tilde{U}_F = 3.473$, compared to the original aerofoil's flutter airspeed of $\tilde{U}_F = 2.853$. This represents an increase of approximately 22% of the flutter airspeed. Since the case's reduced airspeed is slightly lower, at $\tilde{U} = 3.465$, there is no flutter.

As was the case in the original aerofoil, the plunge-dominant mode is the one to flutter. The damping of the pitch-dominant mode is lower for the optimised aerofoil than for the original aerofoil for all freestream velocities. The plunge-dominant mode, on the other hand, behaves in a similar fashion for the two aerofoils for $\tilde{U} < 2$. At higher airspeeds, the slope of the plunge-dominant damping of the optimised aerofoil is shallower than for the original aerofoil, which leads to the

increased flutter speed. As seen in Fig. 5.5, the frequencies of the modes are farther apart compared to the original at high values of the reduced airspeed. This is mostly due to the absence of a reduction in the frequency of the pitch-dominant mode.

5.2 Inviscid mean drag objective function minimisation

Since the unconstrained amplitude minimisation led to an increased steady drag coefficient, using the mean drag coefficient in the objective function could lead to an aerofoil in which this increase was not as marked. Furthermore, as seen in Sec. 4.5.2, the gradients of the inviscid mean drag coefficient are highly correlated with the gradients of the amplitude. For the present test case, the objective function is defined as

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & J(\boldsymbol{\xi}) = \frac{\bar{c}_d}{\bar{c}_{d,o}}, \\ \text{subject to} \quad & \xi_i \in \left[-1.0 \times 10^{-3}, 1.0 \times 10^{-3}\right] \quad \forall i \in [0, 23], \end{aligned} \tag{5.3}$$

where $\bar{c}_d$ is the mean drag coefficient for the unsteady problem and $\bar{c}_{d,o}$ is the mean drag coefficient for the original design. Thus, the value of the original objective function is $J_o = 1$. Since the Euler equations are used to model the flow, only the inviscid component of the drag coefficient is taken into account. As in the unconstrained minimisation of amplitude case, the L-BFGS-B algorithm was used [141, 142].

5.2.1 Optimisation process

The path the optimisation followed is shown in Fig. 5.6. The value of the inviscid drag objective function, J, is shown in blue circles. At the end of the optimisation process, there was an 84% reduction of the average drag coefficient. The final amplitude was reduced more than an order of magnitude, to $0.0064\,\text{rad} \simeq 0.18°$.

The camber line and thickness distribution of the final optimised aerofoil are shown in dashed orange in Figs. 5.7(a) and 5.7(b), respectively. The NACA 64A010 aerofoil is plotted in solid blue. The optimised aerofoil is very similar to the one obtained by minimising the amplitude, being somewhat thicker than the original (10.5% of the chord) and having a slight, reflex camber (maximum of 0.1% of the chord). Compared to the aerofoil described in Fig. 5.3, the present test case's camber had its crossing point farther downstream, around 58% of the chord. After this point, the negative camber is minimal.

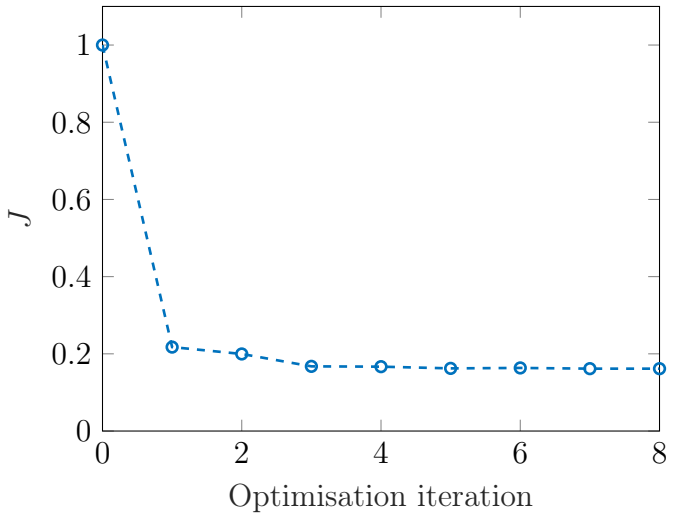


Figure 5.6: Evolution of the mean drag objective function, J, in blue circles

5.2.2 Steady results

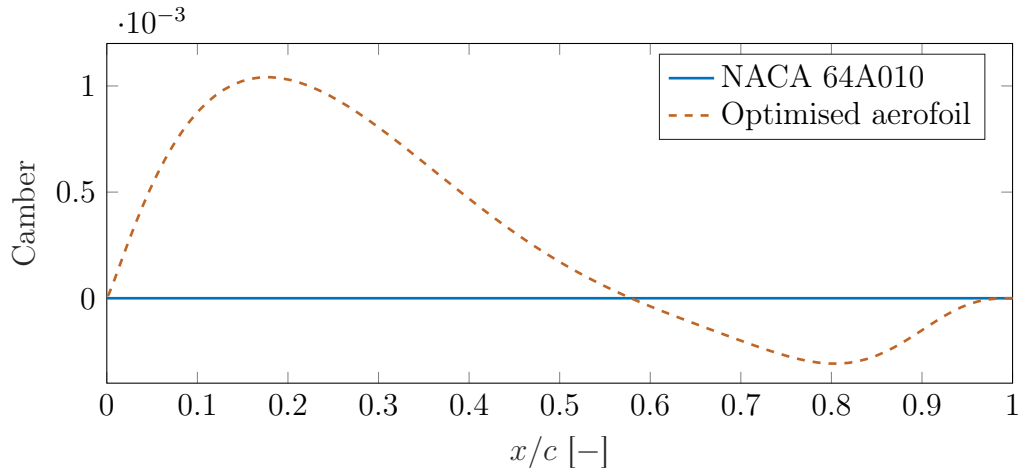
Figure 5.8 shows the steady pressure distribution around the original and optimised aerofoils at $\alpha = 0$. The same process described in Sec. 5.1.2, with steady CFD simulations, was applied. The C_p distribution around the original aerofoil is plotted in blue and around the optimised aerofoil in orange. The upper side for the optimised aerofoil is represented as a dashed line and the lower side as a dotted line. The shock, which is located around 57% of the chord in both sides, is stronger than the original one but weaker than in the unconstrained minimisation of amplitude. The asymmetry is significantly less marked.

The reduced asymmetry leads to a steady lift coefficient that is closer to zero, $c_l = 0.0027$, for $\alpha = 0$. The corresponding steady drag coefficient is $c_d = 0.00431$, which is lower than that obtained in Sec. 5.1 but still $\sim 60\%$ larger than the original value.

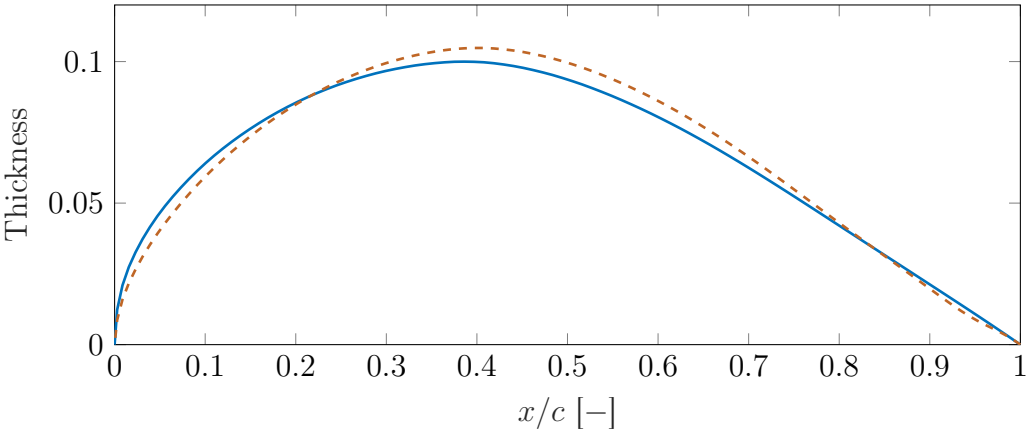
5.2.3 Flutter point

The setup described in Sec. 5.1.3 is used in order to obtain the flutter point of the optimised aerofoil. Since in this case the final pitching amplitude is not 0, a steady FSI simulation is performed in order to obtain the steady pitch angle. The steady pitch is smaller than that of the amplitude-minimised aerofoil. Its value is $\alpha = 0.0005\,\text{rad} \simeq 0.03°$.

The flutter point is at a reduced airspeed of $\tilde{U}_F = 3.432$. It is slightly lower



(a) Camber line



(b) Thickness distribution

Figure 5.7: Comparison of the original NACA 64A010 aerofoil and the minimum mean drag aerofoil

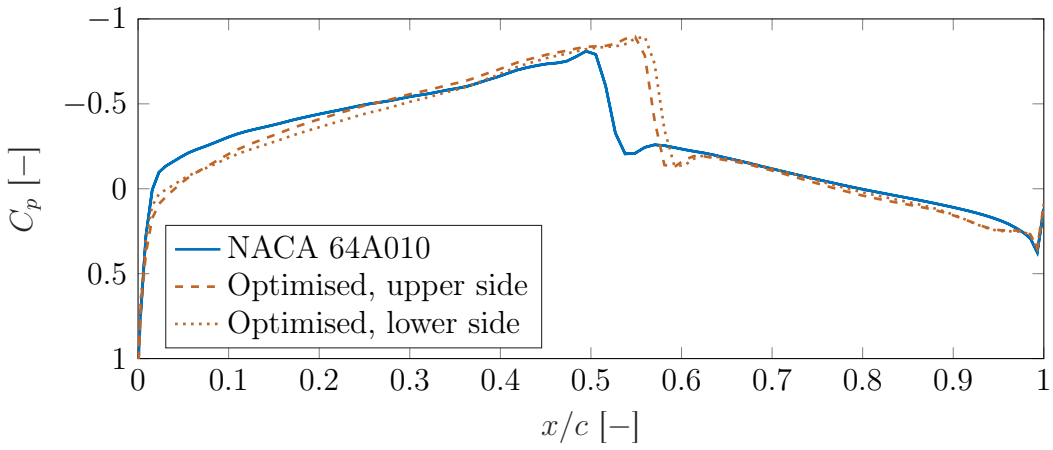


Figure 5.8: Comparison of the steady pressure coefficient distribution around the original and minimum mean drag aerofoils at $\alpha = 0$

than the airspeed used for the computation, $\tilde{U} = 3.465$. This is coherent with the behaviour observed during the optimisation, with the final design having a low, but non-zero pitching amplitude. The flutter point shows an increase of approximately 20% with respect to the unoptimised aerofoil.

5.3 Drag-constrained LCO amplitude minimisation

A mean-drag objective function minimisation has been presented in the previous section. However, there was a smaller, but still significant increase in the steady drag coefficient of the aerofoil at $\alpha = 0$. One way to prevent this issue is by means of a drag-constrained optimisation process.

The optimisation problem is defined as

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & J(\boldsymbol{\xi}) = \frac{|\alpha_1|^2}{|\alpha_{1,o}|^2}, \\ \text{subject to} \quad & \xi_i \in \left[-1.0 \times 10^{-3}, 1.0 \times 10^{-3}\right] \quad \forall i \in [0, 23], \\ & K(\boldsymbol{\xi}) = \left.\frac{c_d}{c_{d,o}}\right|_{\alpha=0} \leq 1.1, \end{aligned} \tag{5.4}$$

where c_d is the steady drag coefficient and $c_{d,o}$ is the steady drag coefficient of the original aerofoil. The constraint allows a 10% increase in the zero-angle-of-attack steady drag coefficient of the aerofoil. These steady problems are solved using the inviscid Euler equations as well, so the drag coefficient does not take into account

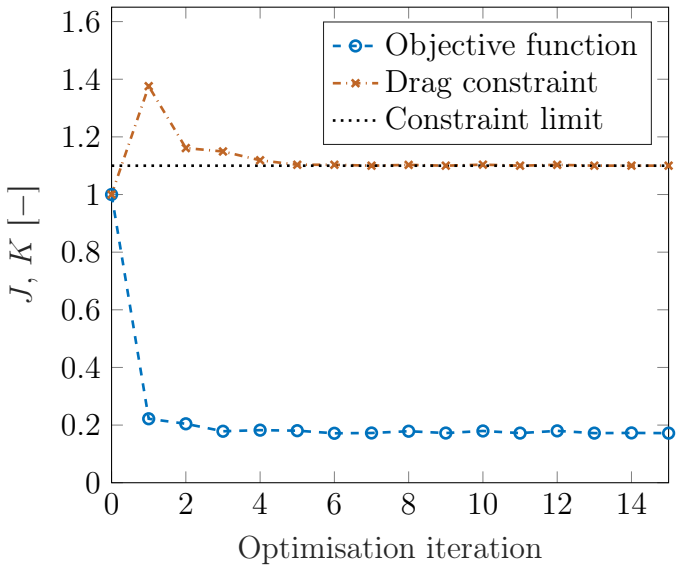


Figure 5.9: Evolution of the amplitude objective function and the steady drag constraint during the optimisation process

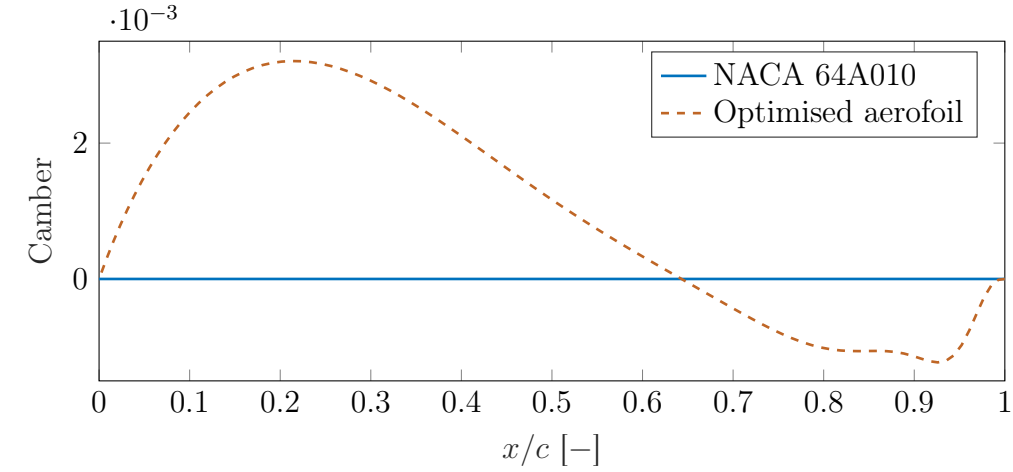
viscous effects. For this constrained optimisation, the Sequential Least Squares Programming (SLSQP) algorithm for constrained optimisation implemented in SciPy[2] was used [143]. Since for this test case the c_d constraint is steady, it is implemented using SU2's already-existing design capabilities. SU2 uses the steady discrete adjoint method in order to calculate the gradients with respect to the shape design variables. The constraint is a purely fluid one, so there is no coupling with a structural solver.

5.3.1 Optimisation process

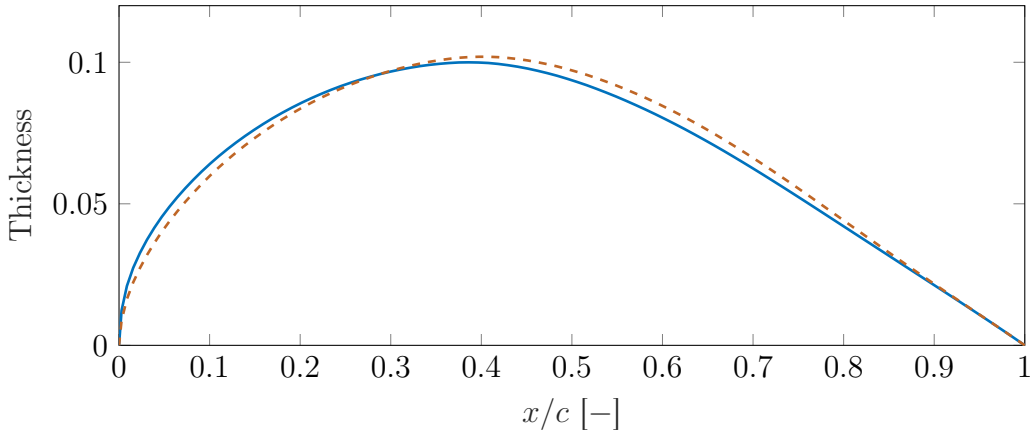
The optimisation path is shown in Fig. 5.9. The objective function, J, is shown as blue circles, while the constraint, K, appears as orange crosses. The maximum value of the constraint appears as a dotted black line. The first iteration greatly reduces the value of the objective function. However, the steady drag constraint is violated. At the end, after 15 iterations, the final design meeting the constraint is obtained. The amplitude of the pitching motion is reduced to $\alpha_1 = 0.0305\,\text{rad} \simeq 1.75°$. This represents a $\simeq 60\%$ reduction in amplitude.

The final optimised aerofoil's camber line and thickness distribution are shown in Fig. 5.10. The asymmetry in the camber line is much more pronounced than

[2] `https://scipy.org/`



(a) Camber line



(b) Thickness distribution

Figure 5.10: Comparison of the original NACA 64A010 aerofoil and the steady drag-constrained optimised aerofoil

in the unconstrained optimisation cases, with a more significant reflex camber line (0.3% of the chord). The crossing point is at 64% of the chord, farther downstream. After this point, there are some oscillations which may be caused by the reduced density of design variables close to the trailing edge. The aerofoil is thinner than the previous two unconstrained optimisation cases, but still thicker than the original one at 10.2% thickness. Unlike those cases, the thickness close to the trailing edge is not reduced.

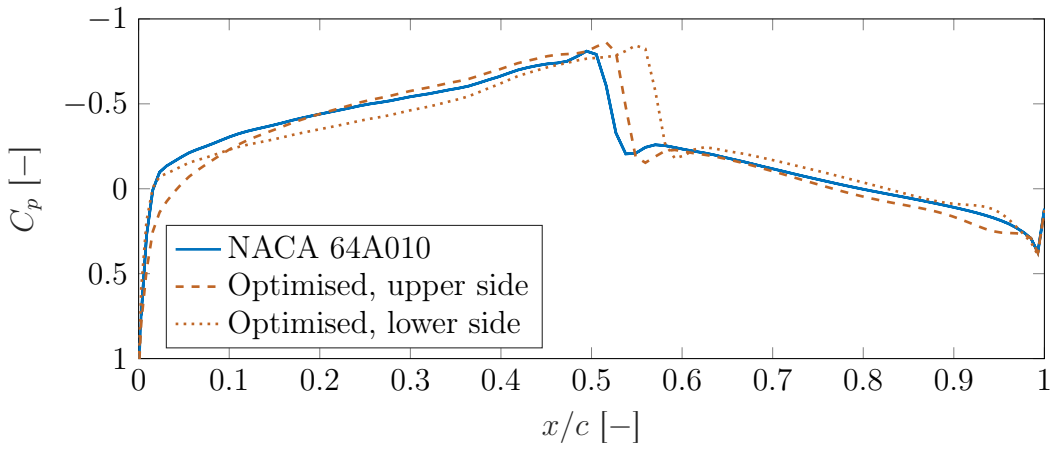


Figure 5.11: Comparison of the steady pressure coefficient distribution around the original NACA 64A010 and the steady drag-constrained aerofoil at $\alpha = 0$

5.3.2 Steady results

The steady pressure distribution around the optimised aerofoil is shown in Fig. 5.11 in dashed orange for the upper side and in dotted orange for the lower side. The original C_p distribution appears as a solid blue line. As in the two unconstrained cases, the pressure around the leading edge is increased, while the shock moves downstream and strengthens. However, the extent of the shock's movement is much lower in this case. Since the fluid model used, Euler, does not include viscous behaviour, the drag force is computed from the pressure distribution.

As expected from the significant camber line, the asymmetry of the pressure distribution is quite pronounced. The shock on the upper side is at 54% of the chord, while on the lower side it is at 57%. The load coefficients at $\alpha = 0$ are $c_l = -0.0252$ and $c_d = 0.00297$. Interestingly, the lift coefficient is negative despite the mostly-positive camber line. The camber line's maximum is at 21% of the chord, which is upstream of the maximum thickness. This positive maximum leads to a higher deflection on the upper side, which causes the shock wave to occur earlier on that side. Since on the lower side the shock wave appears farther downstream, the total lift is negative. The inviscid drag coefficient meets the constraint, while the lift coefficient is larger in magnitude than the ones obtained by unconstrained optimisation.

5.3.3 Flutter point

The steady pitch induced by the asymmetric pressure distribution at the imposed freestream conditions is $\alpha = 0.0037\,\text{rad} \simeq 0.21°$. This value is used for the flutter calculation.

For the drag-constrained optimised aerofoil, flutter occurs at $\tilde{U}_F = 3.169$ instead of $\tilde{U}_F = 2.853$. The increase is of approximately 11%, around half the one observed in the unconstrained optimisation cases. This is coherent with the smaller reduction in amplitude observed.

5.4 Power dissipation maximisation

The optimisation test cases shown previously concerned minimising the amplitude of a limit-cycle oscillation. However, LCOs are not always undesirable. One example of an engineering problem in which they are needed are energy-harvesting applications. They use the self-excited motion of a body in order to generate electricity for remote operations.

An optimisation problem inspired by such applications is treated in the present section. Since the method has not been applied to a code capable of solving the piezoelectric or inductive equations used for energy harvesting, the plunging damper is used instead. Another difference is that most energy harvesting applications occur at very low Mach numbers, where the flow is subcritical and compressibility effects are absent. The setup used, however, is in the transonic regime. It is meant to illustrate the use of the harmonic balance adjoint method in a test case in which eliminating the limit-cycle oscillation is not desirable.

The objective of the test case is to maximise the power dissipated by the plunging damper while not exceeding a maximum LCO amplitude. Thus, the expression of the optimisation problem is

$$\begin{aligned} \max_{\boldsymbol{\xi}} \quad & J(\boldsymbol{\xi}) = \frac{1}{T}\int_0^T c_h \cdot \dot{h}^2 \,\mathrm{d}t = \frac{1}{2\cdot N_h + 1}\sum_{i=0}^{2\cdot N_h} c_h \cdot \dot{h}^2, \\ \text{subject to} \quad & \xi_i \in \left[-1.0\times 10^{-3}, 1.0\times 10^{-3}\right] \quad \forall i \in [0, 23], \\ & \xi_{24} = \zeta_h \in [0, 0.2]\,, \\ & K(\boldsymbol{\xi}) = \frac{|\alpha_1|^2}{|\alpha_{1,o}|^2} \leq 1, \end{aligned} \tag{5.5}$$

where T is the period of the LCO, c_h is the damping of the plunge degree of freedom, $\dot{h}$ is the plunge velocity and N_h is the number of harmonics. The constraint is the

pitching amplitude of the original test case, $\alpha_{1,o} = 0.0735\,\text{rad} \simeq 4.21°$. This constraint limits indirectly the maximum power dissipation. Besides the Hicks-Henne bumps used in the other optimisation problems, the plunge damping is included as a design variable. The plunge damping is bounded between 0 and a value such that $\zeta_h = c_h/(2\sqrt{k_h \cdot m}) = 0.2$. From Eq. (5.5) it can be seen that if $c_h = 0$, the derivative with respect to the shape variables is 0. In order to allow the optimiser to modify the shape of the aerofoil in the first iteration, the starting value is such that the corresponding wind-off uncoupled damping coefficient is $\zeta_h = 0.01$.

The power can be turned non-dimensional for ease of comparison. One possible scaling is

$$\bar{P} = \frac{J}{m\omega_h^3 b^2}, \tag{5.6}$$

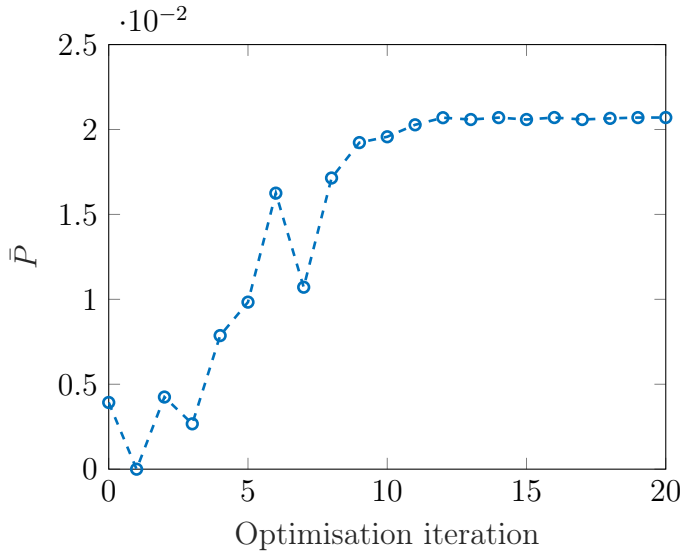
where $\bar{P}$ is the non-dimensional average power, J is the power dissipation objective function defined in Eq. (5.5), ω_h is the uncoupled wind-off natural frequency of the plunge mode and b is the half-chord [144].

In this case, one harmonic balance adjoint simulation is used for the energy dissipation and another for the amplitude constraint. The direct harmonic balance simulation is re-used for the objective function and the constraint in order to reduce the computational cost. As in Sec. 5.3, the Sequential Least Squares Programming (SLSQP) algorithm was used.

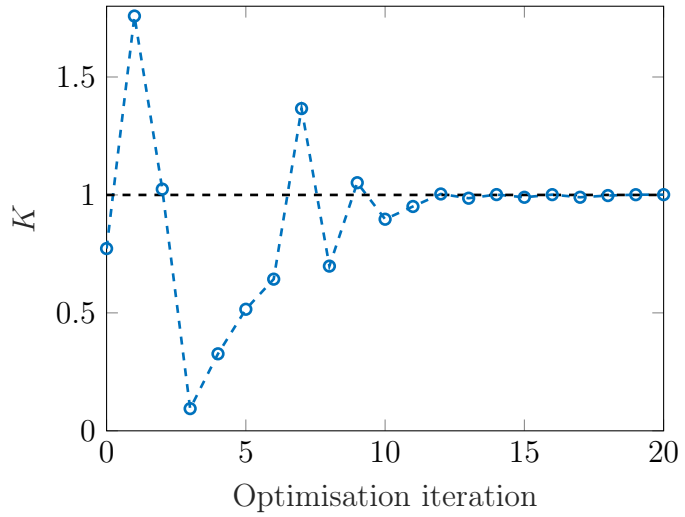
5.4.1 Optimisation process

The evolution of the energy dissipation objective function appears in Fig. 5.12(a) and the corresponding value of the amplitude constraint is in Fig. 5.12(b). Since the damping is non-zero, the first design is not exactly equal to the original test case. As shown in Table 4.1, the pitching amplitude decreases with increasing c_h. Therefore, the first design has a lower amplitude than the original, zero-damping, one.

The evolution of damping is shown in Fig. 5.13(b). In the second design iteration, the optimiser reduces the damping to 0. This results in an elimination of the energy dissipation and an increase in pitching amplitude that violates the constraint. Then, the damping starts to increase. In iteration 3, it reaches the upper bound, $\zeta_h = 0.2$. The pitching amplitude is significantly reduced and so is the dissipated power. Afterwards, it oscillates around the values that will lead to the optimised solution. After iteration number 13, the changes in the magnitudes of interest are marginal. The optimised solution has a damping ratio of $\zeta_h = 0.074$ that leads to a non-dimensional dissipation in the plunging damper of $\bar{P} = 0.0207$.



(a) Non-dimensional power dissipation



(b) Amplitude constraint

Figure 5.12: Evolution of the non-dimensional objective function, $\bar{P}$, and amplitude constraint, K during the optimisation process

At constant plunging amplitude, the power generated is proportional to the movement frequency squared. Consequently, this frequency increases during the optimisation process from the original value of $63.6\,\mathrm{rad\,s^{-1}}$ to $68.2\,\mathrm{rad\,s^{-1}}$. The evolution of the frequency is shown in Fig. 5.13(a). Comparing it to Fig. 5.13(b) and to Fig. 5.12(b), it can be seen that high-frequency designs also have a higher damping ratio and a reduced amplitude. The optimiser finds a balance between these three magnitudes in order to maximise the dissipated power.

The final design is shown in dashed orange in Fig. 5.14. It is compared to the original aerofoil, in solid blue. Unlike the aerofoils that minimise the amplitude, the present case results in a thinner aerofoil. Its maximum thickness is of 9.1%. It also has a slight reflex camber, comparable to the one in the drag minimisation shown in Sec. 5.2 (maximum of 0.1%, but negative). Contrary to the other optimisation cases, it is first negative and then positive.

5.4.2 Steady results

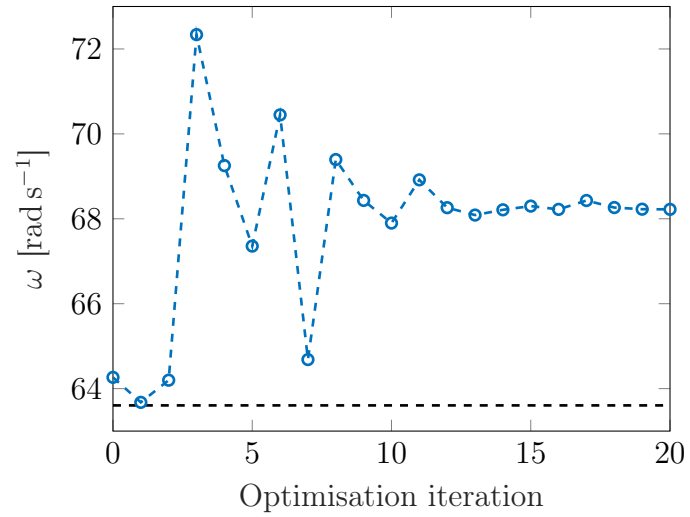
The steady pressure coefficient distribution for the optimised (in orange) and original (in blue) aerofoils is plotted in Fig. 5.15. Since the optimised aerofoil is asymmetric, its upper side is represented as a dashed line and its lower side is a dotted line. The pressure distributions around both aerofoils were calculated using a steady CFD computation at $\alpha = 0$.

Compared to the other optimised aerofoils, the asymmetry of the pressure distribution is smaller, with only slight differences close to the leading edge. Furthermore, the strength of the shock is significantly reduced as it moves upstream, to approximately 47% of the chord. This results in a lower value of the drag coefficient, $c_d = 0.00258$, compared to the original $c_{d,o} = 0.00270$. As a result of the reduced asymmetry, the lift coefficient is negligible.

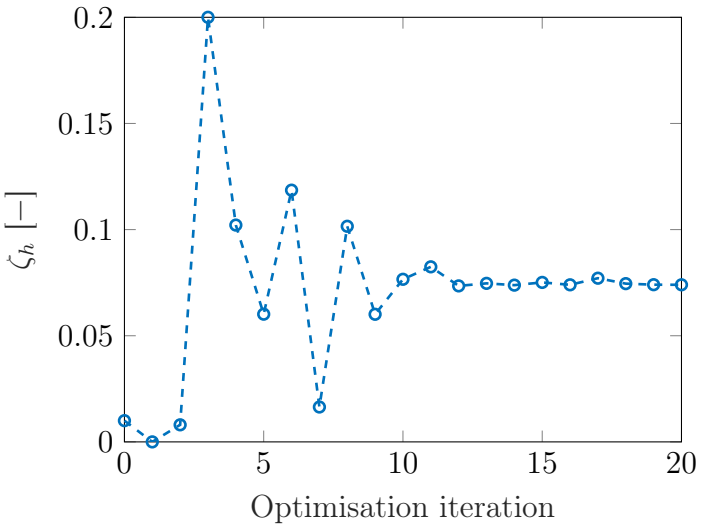
5.5 Summary

The results of optimisations using the FSI harmonic balance method presented in Chapter 2 and the corresponding adjoint method described in Chapter 4 have been shown in the present chapter. A summary table with some magnitudes of interest of the five studied aerofoils is presented in Table 5.1. The table compares the steady drag and lift coefficients at $\alpha = 0$ and the pitching amplitude and mean drag coefficients of the unsteady results.

The optimisation process has been demonstrated for unconstrained and constrained cases, including combining steady and unsteady calculations. The results for those cases minimising the amplitude follow similar trends, even when including

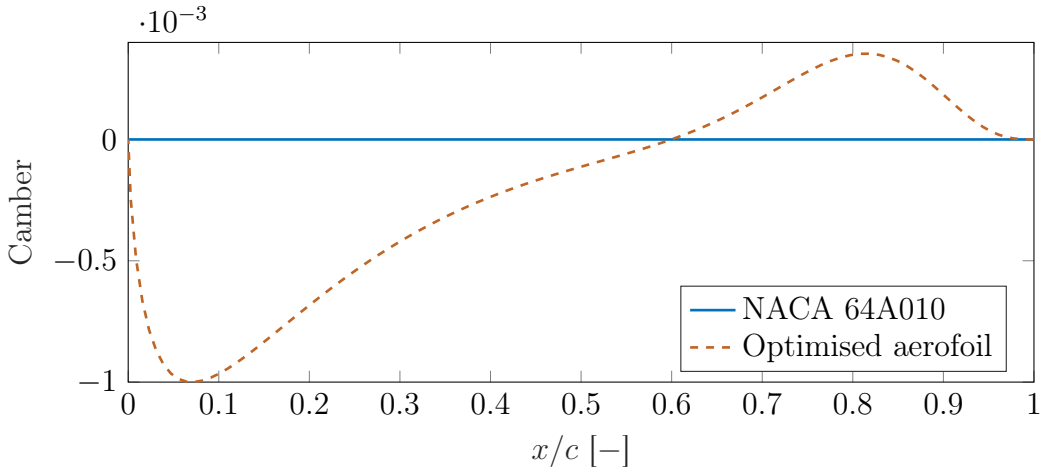


(a) Movement frequency

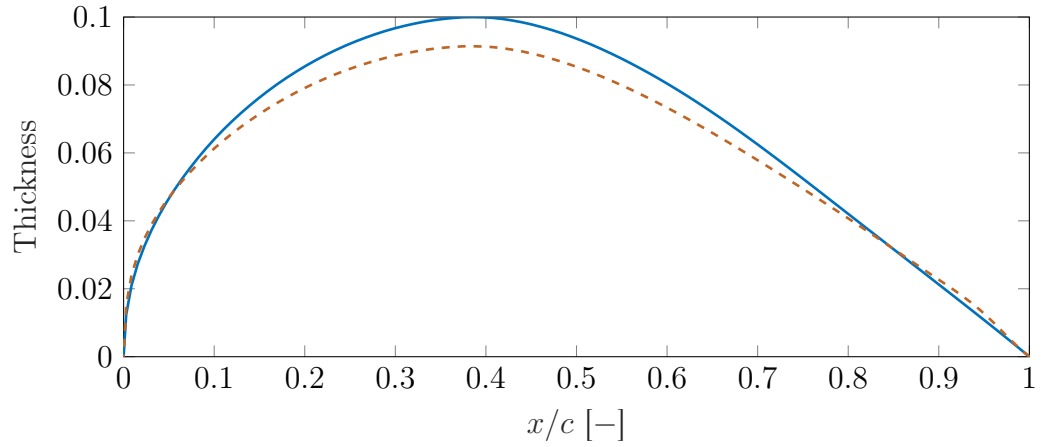


(b) Damping coefficient

Figure 5.13: Evolution of some magnitudes of interest during the constrained power dissipation maximisation



(a) Camber line



(b) Thickness distribution

Figure 5.14: Comparison of the original NACA 64A010 aerofoil and the maximum power dissipation aerofoil

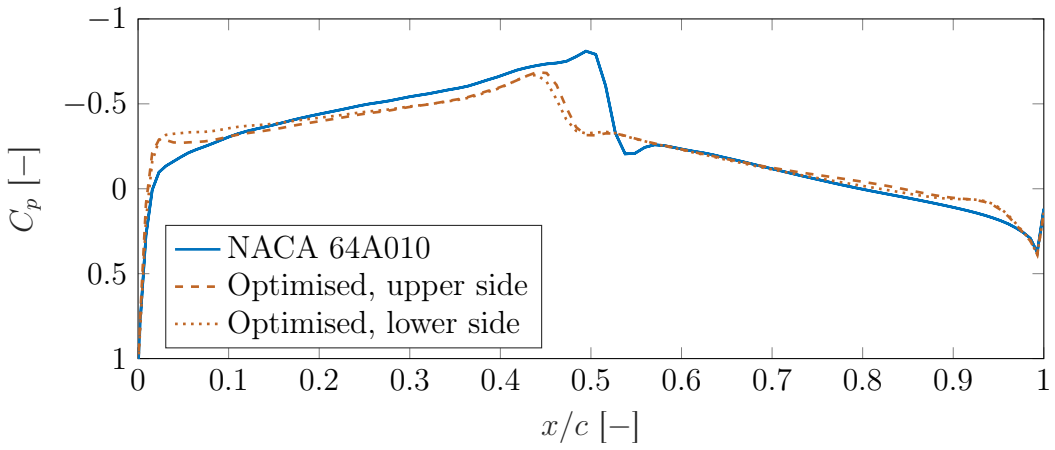


Figure 5.15: Comparison of the steady pressure coefficient distribution around the original NACA 64A010 aerofoil and the maximum power dissipation aerofoil at $\alpha = 0$

a strict steady drag constraint. In general, thicker aerofoils lead to lower amplitudes and higher flutter points, but they increase the strength of the shock and, thus, the steady drag.

Because the harmonic balance solution introduces a non-physical asymmetry, the resulting aerofoils are not symmetric. In the future, a larger number of harmonics could be used in order to check both the impact of the number of harmonics on the optimised solution and whether the asymmetric aerofoils result in a better design. This is more important for the power dissipation maximisation case, since it requires a limit-cycle oscillation with a relatively high amplitude.

In all the optimisation cases it can be seen that small modifications to the shape of the aerofoil can lead to very important changes to the predicted amplitude of LCOs. However, it must be noted that the fluid model is inviscid, which can limit the applicability of the results. Finally, the case maximising the energy dissipation has shown that the method can be applied to more complex objective functions.

Obj. function	Constraint	Steady results ($\alpha = 0$)		Unsteady results	
		c_d [−]	c_l [−]	$\lvert\alpha_1\rvert$ [rad]	$\bar{c}_d$ [−]
Original		0.00270	0	0.07357	0.02699
Pitching amp.	None	0.00516	−0.01497	0	0.00511
Mean drag	None	0.00431	0.00274	0.00637	0.00436
Pitching amp.	Steady drag	0.00297	−0.02521	0.03050	0.00759
Power diss.	Pitching amp.	0.00258	−0.00044	0.07353	0.02170

Table 5.1: Comparison of the optimised aerofoils

Chapter 6

Conclusions and future work

6.1 Conclusions

The main objective of the present work is to accelerate the gradient-based optimisation of time-periodic fluid-structure interaction (FSI) problems. To that effect a harmonic balance (HB) adjoint technique for FSI using a partitioned approach has been developed, implemented and verified.

In order to apply the method, two basic building blocks are needed: a direct solver and an adjoint solver. The direct solver provides the physical solution and the adjoint solver calculates the gradients.

The first building block consists in a partitioned FSI harmonic balance approach for problems with unknown frequency. It includes a new frequency iteration method for partitioned solvers, inspired by monolithic approaches. The technique combines phase-fixing and frequency iteration, removing one structural unknown from the problem. Unlike other frequency iteration approaches, this new method does not require the inclusion of an additional frequency equation.

The complete FSI harmonic balance approach was verified on a typical nonlinear aeroelastic problem with unknown frequency. It consists in a pitch-plunge aerofoil undergoing a limit-cycle oscillation (LCO) in the transonic flow regime. The inviscid Euler equations are used to model the flow field. The proposed harmonic balance approach was successfully verified by comparing its predictions to those of time-marching simulations.

Several test cases with reduced velocities beyond the flutter point have been used to obtain the bifurcation diagram. Two cases are studied in more detail: one with lower pitching amplitude and another one with higher pitching amplitude. The present approach is one order of magnitude faster than a time-marching simulation

for the low-amplitude test case studied. However, there are some differences between the predicted limit-cycle oscillation amplitudes using the HB method with one harmonic and the time marching approach. Namely, the LCO amplitude is overpredicted and a non-physical break of symmetry occurs.

The differences are more important in the higher-amplitude case. For this case, the effect of the phase and the number of harmonics on the harmonic balance solution has been studied. If a sine waveform is used for the pitch instead of a cosine waveform, the break of symmetry is reduced. The discrepancy between harmonic balance and time-marching results can also be reduced by increasing the number of harmonics. While simulations with more harmonics lead to a closer match with the results of the time-marching approach, they result in an increased computational cost. Despite this, all harmonic balance simulations were faster than the time-marching calculations.

One further advantage of the harmonic balance method is that it only requires mesh convergence. In contrast, time-marching requires both mesh- and time-convergence. That is, the end time of the simulation must be long enough for all the transients to have decayed in order to accurately determine the oscillation amplitude. It is not possible to select in advance the time necessary to reach the steady-state solution.

The second building block of the present optimisation methodology consists in the derivation of the adjoint equations for the aeroelastic harmonic balance method. First, a partitioned approach for steady FSI adjoint calculations was implemented and verified. It works with both matching meshes and conservative radial-basis function interpolation methods. A 2D beam in crossflow was used to verify the calculations. Two different structural solvers were employed: a geometrically non-linear FEM-based code and a beam method. The drag coefficient was selected as an objective function. The gradients obtained matched those calculated using finite differences for structural and fluid parameters using both codes.

The final step was to extend the adjoint calculation to unsteady problems. The frequency iteration technique introduces one adjoint equation, while phase-fixing removes another. These adjoint equations were implemented in the pitch-plunge solver used to model the 2D transonic LCO. Furthermore, Hicks-Henne bumps were introduced to the structural solver as shape design variables.

The high-amplitude transonic limit-cycle oscillation test case was used to verify the adjoint harmonic balance method. Two unsteady objective functions were defined: the amplitude of the pitching motion and the mean drag coefficient.

The harmonic balance adjoint technique has several advantages. One is that it only requires saving $2 \cdot N_h + 1$ solutions for N_h harmonics. Whereas for the time-

marching direct problem 20000 time iterations were saved, for the one-harmonic solution only 3 time instances had to be stored. Furthermore, the calculation of the objective functions and their derivatives is very simple to implement, since both objective functions are outputs of the harmonic balance solver. There is no need to define a window or to try to estimate a maximum in order to calculate the mean drag or the pitching amplitude.

For the case studied here, each harmonic balance adjoint calculation is faster than the harmonic balance direct calculation. In total, six structural, one fluid and 24 shape design parameters were used for the comparison. With two objective functions and 31 design variables, the proposed adjoint approach is approximately 30 times faster than central finite differences. The resulting values of the gradients were compared to those evaluated using finite differences and they show a very good match for both objective functions. Interestingly, the mean drag decreases with the Mach number. In this case it is probably because of the reduced motion amplitude.

Finally, the complete adjoint harmonic balance approach has been applied to the optimisation of four test cases. Two of them were unconstrained and two were constrained. The unconstrained cases reflected the objective functions studied for the verification of the adjoint approach. They were pitching amplitude and mean drag minimisation.

The first case eliminated the limit-cycle oscillation, leading to a flutter point slightly beyond the freestream conditions of the problem. However, it significantly increased the steady drag coefficient of the aerofoil. The second case reduced the amplitude of the LCO, since in this case, higher amplitudes lead to higher values of the mean drag.

The two constrained cases demonstrated different aspects of the method. The first one consisted in a steady drag-constrained amplitude minimisation. For that case, a constraint using a fluid solver was implemented. This shows that the proposed methodology can take into account different kinds of solvers. It succeeded in reducing the amplitude, though by a lesser amount than the unconstrained problems.

The second constrained case attempted to maximise power dissipation on a plunge damper while keeping pitch amplitude below its original value. The objective function was inspired by energy harvesting applications. Both the power dissipation and the pitch constraint were calculated using the proposed harmonic balance technique.

In general, thicker aerofoils led to a reduction of the LCO amplitude. This reduction was accompanied by an increase in the flutter speed of the three minimisation cases. Furthermore, since the direct solver introduces a break in the symmetry of

the problem the optimiser modified the camber line of the aerofoil. This led to a reflex camber line in all four cases.

In conclusion, the combined harmonic balance adjoint technique proposed in this work is significantly faster than time-marching approaches or finite differences for the studied cases. It can be used to optimise different objective functions and can be combined with other approaches. While the computational cost is still high, the method represents an important step towards cheap, higher-fidelity LCO calculations for optimisation.

6.2 Challenges

The present project's objective was to reduce the computational cost of gradient calculation of limit-cycle oscillation problems by combining two complex approaches: the harmonic balance and the adjoint methods. As such, it required understanding the techniques, developing and combining them, implementing the combined approaches and verifying their results. In several occasions, this proved quite challenging.

For the harmonic balance part of the project, two main difficulties arose: the change in volume of cells with time and the introduction of a frequency iteration technique. The pre-existing time-domain harmonic balance method in SU2 assumed that the volume of cells was constant. This simplified the calculations of the source term by taking the volume out of them. However, the proposed approach deforms the mesh, which enables the use of more general structural solvers. Therefore, SU2 had to be modified in order to allow for varying cell volume, as part of this PhD thesis as shown in Sec. A.1.2.

The development of the new frequency iteration technique was not originally part of the project. It had to be introduced because the FSI solution diverged when using the L^2 norm of the residual approach detailed in Sec. 2.5.1 for the case shown in Sec. 2.8.

With respect to the adjoint, there was one main challenge. The rigid body motion integrator's adjoint solver uses an explicit formulation that does not require iteration. However, it also means that the Jacobian has to be calculated exactly [36]. Therefore, the dependence of the moment on the pitching angle had to be included, as shown in Sec. 4.3.2. The lack of agreement in gradients between finite differences and the adjoint method resulting from the inexact Jacobian used delayed the project, compounded by convergence issues on the grid being used at the time limiting the available precision for finite differences.

6.3 Suggestions for further work

6.3.1 Reducing the computational cost

One of the issues seen in the direct solution in Sec. 2.8 is that the lack of time resolution in flows with discontinuities, such as shocks, leads to unexpected behaviour. In particular, the harmonic balance method resulted in a non-physical break of symmetry and a change in the estimation of the loads for the case studied here. While using more harmonics significantly improves the solution, the computational cost is also increased. Two possible ways to reduce the computational cost of more accurate solutions are suggested here: introducing a frequency-domain harmonic balance solver and introducing the derivative of loads with respect to the frequency in the frequency iteration technique.

Frequency-domain harmonic balance fluid solver

There are two main approaches for frequency-domain harmonic balance CFD solvers: the convolution method and Fourier transform-based techniques. The former approach with one harmonic was used by Ning and He in order to calculate the flow around an oscillating turbine blade [145]. However, it was reported by Hall et al. that for the Euler equations the convolution method requires $\mathcal{O}(N_h^3)$ operations per iteration, where N_h is the number of harmonics. This makes using the approach unfeasible for moderate numbers of harmonics. Furthermore, some terms in turbulence models are too complex to obtain analytically in the frequency domain [52].

The nonlinear terms of the equations could also be estimated using a Fast Fourier-Transform (FFT). The main advantage with respect to the convolution technique is that the FFT requires fewer operations, especially if the number of time instances is a power of 2 [85]. The FFT-based approach was applied by McMullen et al. to both the Navier-Stokes and Reynolds-averaged Navier-Stokes equations and a rigid mesh [146].

Derivative of loads with respect to frequency

In order to improve the convergence of the L^2 residual norm technique, Yao and Marques introduced the derivative of loads with respect to frequency. This approach could be adapted in order to work with the combined technique proposed in Sec. 2.5.3.

6.3.2 Impact of shape changes on interpolation matrices

Since the matrices used for interpolation depend on the position of the boundary nodes for non-matching meshes and aerodynamic shape optimisation modifies the external shape of a body, it would be interesting to study the effect of the interpolation-related terms on the gradients. The present study used matching meshes, so this effect was absent.

6.4 Further applications of the coupled adjoint harmonic balance method

6.4.1 Multipoint constraints

The optimisation test cases in Chapter 5 have centred around the limit-cycle oscillation of an aerofoil at a specific Mach number. However, LCOs may appear at a range of conditions that must be studied. The method described in the present work could be applied to implement multipoint LCO amplitude constraints.

6.4.2 Extension to three dimensions

Many problems of engineering interest, such as the limit-cycle oscillations of wings, are three-dimensional. One example is the Goland$^+$ wing, which was introduced by Eastep and Olsen [147]. If a store is added to this wing, it undergoes LCOs in the transonic regime [61, 148, 149].

The harmonic balance coupling described in Sec. 2.6.1 for the direct problem and in Sec. 4.3.3 for the adjoint problem can be applied in 3D cases without any modifications. However, such an application would also require a three-dimensional structural solver. A finite element method simulation that models the complete structure could be used. However, the frequency iteration technique proposed in Sec. 2.5.3 requires fixing the phase of one state and a reduced number of structural states simplifies the choice. Many aeroelastic problems are formulated by means of linear modal models or nonlinear beam approaches, which already feature a small number of states.

6.4.3 Nonlinear structural solver

Besides aerodynamic nonlinearities, structural nonlinear behaviour may also lead to limit-cycle oscillations. For the two-degree-of-freedom model described in Appendix C, the springs can be modified to include behaviour such as freeplay, hysteresis or hardening/softening. Using a structural solver with cubic stiffness is a

useful way of modelling hardening and softening of the spring. A nonlinear mass matrix could also be implemented in the code.

6.4.4 Extension of multi-physics coupling

There has been considerable interest in using aeroelastic movement for energy harvesting. Several aeroelastic phenomena, such as vortex-induced vibrations, galloping or limit-cycle oscillations have been studied. Such studies are usually conducted at very low Mach numbers, in the incompressible regime [150]. In order to extend the present method to energy-harvesting applications, piezoelectric [151, 152] and inductive [144] coupling terms would have to be added.

6.4.5 Surrogate model of the harmonic balance approach

Surrogate models are a family of reduced-order approaches that approximate the response of complex systems based on known responses. Some surrogate models use derivatives in order to improve the prediction of the desired magnitudes [153, 154]. If the number of outputs is much smaller than the number of inputs, adjoint methods can be used in order to reduce the computational cost of applying these techniques. The harmonic balance method, both direct and adjoint, could be used to provide the known values for the surrogate model at a reduced cost.

Bibliography

[1] A. R. Collar. 'The Expanding Domain of Aeroelasticity'. In: *The Journal of the Royal Aeronautical Society* 50.428 (1946), pp. 613–636. DOI: 10.1017/s0368393100120358 (cit. on pp. 1, 2).

[2] D. S. Woolston, H. L. Runyan, and T. A. Byrdsong. *Some effects of system nonlinearities in the problem of aircraft flutter*. Tech. rep. 3539. National Advisory Committee for Aeronautics, 1955. URL: https://ntrs.nasa.gov/citations/19930084263 (cit. on pp. 3, 4).

[3] S. F. Shen. 'An Approximate Analysis of Nonlinear Flutter Problems'. In: *Journal of the Aerospace Sciences* 26.1 (1959), pp. 25–32. ISSN: 1936-9999. DOI: 10.2514/8.7914 (cit. on p. 4).

[4] N. M. Krylov and N. N. Bogoliubov. *Introduction to Non-Linear Mechanics. (AM-11) (Annals of Mathematics Studies)*. A free translation by S. Lefschetz. Princeton University Press, 1949, p. 106. ISBN: 9780691079851 (cit. on p. 4).

[5] D. A. Masters, N. J. Taylor, T. Rendall, C. B. Allen, and D. J. Poole. 'Review of Aerofoil Parameterisation Methods for Aerodynamic Shape Optimisation'. In: *53rd AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 2015. DOI: 10.2514/6.2015-0761 (cit. on pp. 6, 114–116).

[6] K. Ekici and K. C. Hall. 'Harmonic Balance Analysis of Limit Cycle Oscillations in Turbomachinery'. In: *AIAA Journal* 49.7 (2011), pp. 1478–1487. DOI: 10.2514/1.j050858 (cit. on pp. 7, 30, 31).

[7] R. Prasad, S. Choi, and M. Patil. 'Aerodynamic shape optimization using a time spectral coupled adjoint for nonlinear aeroelastic problems'. In: *Aerospace Science and Technology* (2022), p. 107495. DOI: 10.1016/j.ast.2022.107495 (cit. on pp. 7, 33, 43, 106, 107).

[8] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. 'SU2: An Open-Source Suite for Multiphysics Simulation and Design'. In: *AIAA Journal* 54.3 (2016), pp. 828–846. DOI: 10.2514/1.j053813 (cit. on pp. 7, 38, 87, 88, 118, 177).

[9] A. Rubino et al. 'Adjoint-based fluid dynamic design optimization in quasi-periodic unsteady flow problems using a harmonic balance method'. In: *Journal of Computational Physics* 372 (2018), pp. 220–235. DOI: 10.1016/j.jcp.2018.06.023 (cit. on pp. 7, 25, 38, 177, 181).

[10] A. Crovato et al. 'Effect of Levels of Fidelity on Steady Aerodynamic and Static Aeroelastic Computations'. In: *Aerospace* 7.4 (2020), p. 42. DOI: 10.3390/aerospace7040042 (cit. on p. 12).

[11] M. J. de C. Henshaw et al. 'Non-linear aeroelastic prediction for aircraft applications'. In: *Progress in Aerospace Sciences* 43.4-6 (2007), pp. 65–137. DOI: 10.1016/j.paerosci.2007.05.002 (cit. on p. 12).

[12] E. Albano and W. P. Rodden. 'A Doublet-Lattice Method for Calculating Lift Distributions on Oscillating Surfaces in Subsonic Flows'. In: *AIAA Journal* 7.2 (1969), pp. 279–285. DOI: 10.2514/3.5086 (cit. on p. 12).

[13] M. Blair. *A Compilation of the Mathematics Leading to the Doublet Lattice Method*. Tech. rep. Defense Technical Information Center, 1992. DOI: 10.21236/ada256304. URL: https://apps.dtic.mil/sti/citations/ADA256304 (cit. on p. 12).

[14] J. Katz and A. Plotkin. *Low-Speed Aerodynamics*. Cambridge University Press, 2001. DOI: 10.1017/cbo9780511810329 (cit. on p. 12).

[15] L. Morino. *A general theory of unsteady compressible potential aerodynamics*. Contractor Report NASA CR-2464. NASA, 1974 (cit. on p. 12).

[16] L. Morino, L. Chen, and E. O. Suciu. 'Steady and Oscillatory Subsonic and Supersonic Aerodynamics around Complex Configurations'. In: *AIAA Journal* 13.3 (1975), pp. 368–374 (cit. on p. 12).

[17] C. S. Prasad and G. Dimitriadis. 'Double wake vortex lattice modeling of horizontal axis wind turbines'. In: *Proceedings of the 15th International Forum on Aeroelasticity and Structural Dynamics, IFASD*. 180. 2011. URL: http://hdl.handle.net/2268/94931 (cit. on p. 12).

[18] J. Murua, R. Palacios, and J. M. R. Graham. 'Applications of the unsteady vortex-lattice method in aircraft aeroelasticity and flight dynamics'. In: *Progress in Aerospace Sciences* 55 (2012), pp. 46–72. DOI: 10.1016/j.paerosci.2012.06.001 (cit. on p. 12).

[19] J. Gardiner et al. 'Simulation of bird wing flapping using the unsteady vortex lattice method'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics, IFASD 2013*. 2013 (cit. on p. 12).

[20] I. Castro-Fernández and R. Cavallaro. 'A harmonic balance aeroelastic solver with frequency-domain aerodynamics applied to nonlinear flags flutter'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics 2022*. 2022 (cit. on pp. 12, 25).

[21] M. Sánchez Martínez and G. Dimitriadis. 'An improved subsonic unsteady source and doublet panel method'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics 2022*. 2022. URL: https://hdl.handle.net/2268/292462 (cit. on p. 12).

[22] M. Sánchez Martínez and G. Dimitriadis. 'Subsonic source and doublet panel methods'. In: *Journal of Fluids and Structures* 113 (2022), p. 103624. DOI: 10.1016/j.jfluidstructs.2022.103624 (cit. on p. 12).

[23] M. G. Farmer, P. W. Hanson, and E. C. Wynne. 'Comparison of Supercritical and Conventional Wing Flutter Characteristics'. In: *17th Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 1976. DOI: 10.2514/6.1976-1560 (cit. on p. 13).

[24] A. V. Balakrishnan. 'Transonic Small Disturbance Potential Equation'. In: *AIAA Journal* 42.6 (2004), pp. 1081–1088. DOI: 10.2514/1.5101 (cit. on p. 13).

[25] T. L. Holst. 'Transonic flow computations using nonlinear potential methods'. In: *Progress in Aerospace Sciences* 36.1 (2000), pp. 1–61. DOI: 10.1016/s0376-0421(99)00010-x (cit. on p. 14).

[26] S. Samant et al. 'TRANAIR - A computer code for transonic analyses of arbitrary configurations'. In: *25th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 1987. DOI: 10.2514/6.1987-34 (cit. on p. 14).

[27] A. Crovato et al. 'A Full Potential Static Aeroelastic Solver for Preliminary Aircraft Design'. In: *Proceedings of the 18th International Forum on Aeroelasticity and Structural Dynamics (IFASD 2019)*. 2019 (cit. on p. 14).

[28] A. Crovato et al. 'Fast full potential based aerostructural optimization calculations for preliminary aircraft design'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics 2022*. 2022 (cit. on p. 14).

[29] Z. Zhang, F. Liu, and D. Schuster. 'An Efficient Euler Method on Non-Moving Cartesian Grids with Boundary-Layer Correction for Wing Flutter Simulations'. In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2006. DOI: 10.2514/6.2006-884 (cit. on p. 16).

[30] P. C. Chen, Z. Zhang, A. Sengupta, and D. D. Liu. 'Overset Euler/Boundary-Layer Solver with Panel-Based Aerodynamic Modeling for Aeroelastic Applications'. In: *Journal of Aircraft* 46.6 (2009), pp. 2054–2068. DOI: 10.2514/1.43434 (cit. on p. 16).

[31] J. W. Edwards. 'Transonic Shock Oscillations and Wing Flutter Calculated with an Interactive Boundary Layer Coupling Method'. In: *EUROMECH-Colloquium: Simulation of Fluid-Structure Interaction in Aeronautics*. 1996 (cit. on p. 16).

[32] G. De Nayer, M. Breuer, and J. N. Wood. 'Numerical investigations on the dynamic behavior of a 2-DOF airfoil in the transitional Re number regime based on fully coupled simulations relying on an eddy-resolving technique'. In: *International Journal of Heat and Fluid Flow* 85 (2020), p. 108631. DOI: 10.1016/j.ijheatfluidflow.2020.108631 (cit. on p. 16).

[33] T. B. Goonan, R. P. R. Cardoso, and O. B. Adetoro. 'Numerical investigation into the effects of turbulence modelling on the aeroelastic analysis of flexible wings'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics 2022*. 2022 (cit. on p. 16).

[34] S. Piperno, C. Farhat, and B. Larrouturou. 'Partitioned procedures for the transient solution of coupled aeroelastic problems Part I: Model problem, theory and two-dimensional application'. In: *Computer Methods in Applied Mechanics and Engineering* 124.1-2 (1995), pp. 79–112. DOI: 10.1016/0045-7825(95)92707-9 (cit. on p. 16).

[35] C. Farhat, M. Lesoinne, and P. L. Tallec. 'Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity'. In: *Computer Methods in Applied Mechanics and Engineering* 157.1-2 (1998), pp. 95–114. DOI: 10.1016/s0045-7825(97)00216-8 (cit. on p. 16).

[36] R. Sanchez Fernandez. 'A coupled adjoint method for optimal design in fluid-structure interaction problems with large displacements'. PhD thesis. Department of Aeronautics, 2018. URL: http://hdl.handle.net/10044/1/58882 (cit. on pp. 16, 38, 87, 154).

[37] D. Thomas. 'Efficient and flexible implementation of an interfacing Python-based tool for numerical simulations of fluid-structure interaction problems'. English. PhD thesis. ULiège - Université de Liège, 2021. URL: https://hdl.handle.net/2268/252830 (cit. on pp. 16, 19, 37, 87).

[38] B. M. Irons and R. C. Tuck. 'A version of the Aitken accelerator for computer iteration'. In: *International Journal for Numerical Methods in Engineering* 1.3 (1969), pp. 275–277. DOI: 10.1002/nme.1620010306 (cit. on p. 18).

[39] U. Küttler and W. A. Wall. 'Fixed-point fluid–structure interaction solvers with dynamic relaxation'. In: *Computational Mechanics* 43.1 (2008), pp. 61–72. DOI: 10.1007/s00466-008-0255-5 (cit. on p. 18).

[40] J. Degroote, K.-J. Bathe, and J. Vierendeels. 'Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction'. In: *Computers & Structures* 87.11-12 (2009), pp. 793–801. DOI: 10.1016/j.compstruc.2008.11.013 (cit. on p. 18).

[41] A. Beckert and H. Wendland. 'Multivariate interpolation for fluid-structure-interaction problems using radial basis functions'. In: *Aerospace Science and Technology* 5.2 (2001), pp. 125–134. DOI: 10.1016/s1270-9638(00)01087-7 (cit. on pp. 18, 20, 22).

[42] A. de Boer, A. H. van Zuijlen, and H. Bijl. 'Radial Basis Functions for Interface Interpolation and Mesh Deformation'. In: *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, 2009, pp. 143–178. DOI: 10.1007/978-3-642-03344-5_6 (cit. on pp. 19, 21).

[43] H. Wendland. 'Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree'. In: *Advances in Computational Mathematics* 4.1 (1995), pp. 389–396. DOI: 10.1007/bf02123482 (cit. on pp. 20, 22, 93).

[44] J. Duchon. 'Splines minimizing rotation-invariant semi-norms in Sobolev spaces'. In: *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1977, pp. 85–100. ISBN: 9783540374961. DOI: 10.1007/bfb0086566 (cit. on p. 21).

[45] A. de Boer, A. van Zuijlen, and H. Bijl. 'Review of coupling methods for non-matching meshes'. In: *Computer Methods in Applied Mechanics and Engineering* 196.8 (2007), pp. 1515–1525. DOI: 10.1016/j.cma.2006.03.017 (cit. on p. 21).

[46] J. M. Verdon. *The unsteady flow in the far field of an isolated blade row*. Tech. rep. United Technologies Research Center, 1987 (cit. on p. 23).

[47] W. S. Clark and K. C. Hall. 'A Numerical Model of the Onset of Stall Flutter in Cascades'. In: *Volume 5: Manufacturing Materials and Metallurgy; Ceramics; Structures and Dynamics; Controls, Diagnostics and Instrumentation; Education; IGTI Scholar Award*. American Society of Mechanical Engineers, 1995. DOI: 10.1115/95-gt-377 (cit. on pp. 23–25).

[48] W. S. Clark and K. C. Hall. 'A time-linearized Navier-Stokes analysis of stall flutter'. In: *Journal of Turbomachinery* 122.3 (1999), pp. 467–476. DOI: 10.1115/1.1303073 (cit. on pp. 24, 25).

[49] G. Dufour, F. Sicot, G. Puigt, C. Liauzun, and A. Dugeai. 'Contrasting the Harmonic Balance and Linearized Methods for Oscillating-Flap Simulations'. In: *AIAA Journal* 48.4 (2010), pp. 788–797. DOI: 10.2514/1.43401 (cit. on pp. 24, 25).

[50] M. Widhalm, R. Dwight, R. Thormann, and A. Hübner. 'Efficient computation of dynamic stability data with a linearized frequency domain solver'. In: *ECCOMAS CFD 2010*. 2010. URL: https://elib.dlr.de/64919 (cit. on p. 24).

[51] A. D. Ronch, A. J. McCracken, K. J. Badcock, M. Widhalm, and M. S. Campobasso. 'Linear Frequency Domain and Harmonic Balance Predictions of Dynamic Derivatives'. In: *Journal of Aircraft* 50.3 (2013), pp. 694–707. DOI: 10.2514/1.c031674 (cit. on p. 24).

[52] K. C. Hall, J. P. Thomas, and W. S. Clark. 'Computation of Unsteady Nonlinear Flows in Cascades Using a Harmonic Balance Technique'. In: *AIAA Journal* 40.5 (2002), pp. 879–886. DOI: 10.2514/2.1754 (cit. on pp. 25, 155).

[53] F. Sicot, A. Gomar, G. Dufour, and A. Dugeai. 'Time-Domain Harmonic Balance Method for Turbomachinery Aeroelasticity'. In: *AIAA Journal* 52.1 (2014), pp. 62–71. DOI: 10.2514/1.j051848 (cit. on p. 25).

[54] J. P. Thomas, E. H. Dowell, and K. C. Hall. 'Modeling Viscous Transonic Limit Cycle Oscillation Behavior Using a Harmonic Balance Approach'. In: *Journal of Aircraft* 41.6 (2004), pp. 1266–1274. DOI: 10.2514/1.9839 (cit. on p. 25).

[55] T. A. Fitzgibbon, M. A. Woodgate, G. N. Barakos, and R. H. Markiewicz. 'Rotor-Blade Planform Design Based on an Overset Harmonic-Balance-Adjoint Optimization Framework'. In: *AIAA Journal* 59.9 (2021), pp. 3431–3447. DOI: 10.2514/1.j060175 (cit. on p. 25).

[56] G. Dimitriadis. *Introduction to Nonlinear Aeroelasticity*. John Wiley & Sons, Ltd, 2017. DOI: 10.1002/9781118756478 (cit. on pp. 25, 32, 48).

[57] G. Dimitriadis. 'Continuation of Higher-Order Harmonic Balance Solutions for Nonlinear Aeroelastic Systems'. In: *Journal of Aircraft* 45.2 (2008), pp. 523–537. DOI: 10.2514/1.30472 (cit. on p. 27).

[58] T. M. Cameron and J. H. Griffin. 'An Alternating Frequency/Time Domain Method for Calculating the Steady-State Response of Nonlinear Dynamic Systems'. In: *Journal of Applied Mechanics* 56.1 (1989), pp. 149–154. DOI: 10.1115/1.3176036 (cit. on p. 28).

[59] H. Li and K. Ekici. 'Revisiting the One-shot method for modeling limit cycle oscillations: Extension to two-degree-of-freedom systems'. In: *Aerospace Science and Technology* 69 (2017), pp. 686–699. DOI: 10.1016/j.ast.2017.07.037 (cit. on pp. 31, 34, 43, 55, 57, 65, 67).

[60] W. Yao and S. Marques. 'Prediction of Transonic Limit-Cycle Oscillations Using an Aeroelastic Harmonic Balance Method'. In: *AIAA Journal* 53.7 (2015), pp. 2040–2051. DOI: 10.2514/1.j053565 (cit. on pp. 31, 43, 155).

[61] W. Yao and S. Marques. 'A harmonic balance method for nonlinear fluid structure interaction problems'. In: *Computers & Structures* 201 (2018), pp. 26–36. DOI: 10.1016/j.compstruc.2018.02.003 (cit. on pp. 31, 156).

[62] N. Simiriotis and R. Palacios. 'A comparison of direct and sampling-based flutter solution methods using CFD'. In: *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics 2022*. 2022 (cit. on pp. 32, 33, 43, 55, 57, 65).

[63] J. P. Thomas, E. H. Dowell, and K. C. Hall. 'Nonlinear Inviscid Aerodynamic Effects on Transonic Divergence, Flutter, and Limit-Cycle Oscillations'. In: *AIAA Journal* 40.4 (2002), pp. 638–646. DOI: 10.2514/2.1720 (cit. on pp. 33, 43).

[64] D. Thomas et al. 'CUPyDO - an integrated Python environment for coupled fluid-structure simulations'. In: *Advances in Engineering Software* 128 (2019), pp. 69–85. DOI: 10.1016/j.advengsoft.2018.05.007 (cit. on pp. 37, 44, 87, 93).

[65] M. L. Cerquaglia, D. Thomas, R. Boman, V. Terrapon, and J.-P. Ponthot. 'A fully partitioned Lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects'. In: *Computer Methods in Applied Mechanics and Engineering* (2019). ISSN: 0045-7825. DOI: 10.1016/j.cma.2019.01.021 (cit. on p. 37).

[66] M. L. Cerquaglia. 'Development of a fully-partitioned PFEM-FEM approach for fluid-structure interaction problems characterized by free surfaces, large solid deformations, and strong added-mass effects'. English. PhD thesis. ULiège - Université de Liège, 2019. URL: https://hdl.handle.net/2268/233166 (cit. on p. 37).

[67] D. M. Beazley. 'SWIG: an easy to use tool for integrating scripting languages with C and C++.' In: *Fourth Annual USENIX Tcl/Tk Workshop*. Monterey, CA: USENIX Association, 1996. URL: https://www.usenix.org/conference/fourth-annual-usenix-tcltk-workshop/swig-easy-use-tool-integrating-scripting-languages-c (cit. on p. 37).

[68] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. 'Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries'. In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset, and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202. DOI: 10.1007/978-1-4612-1986-6_8 (cit. on p. 37).

[69] L. Dalcín, R. Paz, M. Storti, and J. D'Elía. 'MPI for Python: Performance improvements and MPI-2 extensions'. In: *Journal of Parallel and Distributed Computing* 68.5 (2008), pp. 655–662. DOI: 10.1016/j.jpdc.2007.09.005 (cit. on p. 37).

[70] L. D. Dalcín, R. R. Paz, P. A. Kler, and A. Cosimo. 'Parallel distributed computing using Python'. In: *Advances in Water Resources* 34.9 (2011), pp. 1124–1139. DOI: 10.1016/j.advwatres.2011.04.013 (cit. on p. 37).

[71] R. Sanchez et al. 'Coupled adjoint-based sensitivities in large-displacement fluid-structure interaction using algorithmic differentiation'. In: *International Journal for Numerical Methods in Engineering* 113.7 (2017), pp. 1081–1107. DOI: 10.1002/nme.5700 (cit. on pp. 38, 87, 88).

[72] B. Y. Zhou et al. 'A Discrete Adjoint Framework for Unsteady Aerodynamic and Aeroacoustic Optimization'. In: *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2015. DOI: 10.2514/6.2015-3355 (cit. on pp. 38, 105).

[73] F. Blanc, F.-X. Roux, and J.-C. Jouhaud. 'Harmonic-Balance-Based Code-Coupling Algorithm for Aeroelastic Systems Subjected to Forced Excitation'. In: *AIAA Journal* 48.11 (2010), pp. 2472–2481. DOI: 10.2514/1.45444 (cit. on pp. 38, 39).

[74] F. R. Menter. *Improved two-equation k-omega turbulence models for aerodynamic flows*. Tech. rep. NASA Ames Research Center, Moffett Field, CA, United States, 1992 (cit. on p. 39).

[75] F. R. Menter. 'Two-equation eddy-viscosity turbulence models for engineering applications'. In: *AIAA Journal* 32.8 (1994), pp. 1598–1605. DOI: 10.2514/3.12149 (cit. on p. 39).

[76] W. Zhang, B. Wang, Z. Ye, and J. Quan. 'Efficient Method for Limit Cycle Flutter Analysis Based on Nonlinear Aerodynamic Reduced-Order Models'. In: *AIAA Journal* 50.5 (2012), pp. 1019–1028. DOI: 10.2514/1.j050581 (cit. on p. 43).

[77] C. Geuzaine and J.-F. Remacle. 'Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities'. In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331. DOI: 10.1002/nme.2579 (cit. on p. 45).

[78] H. Güner. 'Unsteady aerodynamic modeling methodology based on Dynamic Mode Interpolation (DMI) for transonic flutter calculations'. English. PhD thesis. ULiège - Université de Liège, 2020. URL: https://hdl.handle.net/2268/245578 (cit. on pp. 46–48).

[79] C. A. K. Irwin and P. R. Guyett. *The subcritical response and flutter of a swept-wing model*. Tech. rep. 3497. Aeronautical Research Council Reports and Memoranda, 1965 (cit. on p. 47).

[80] A. Jocelyn Lawrence and P. Jackson. *Comparison of different methods of assessing the free oscillatory characteristics of aeroelastic systems*. Tech. rep. 1084. Aeronautical Research Council Current Papers, 1968 (cit. on p. 47).

[81] H. J. Hassig. 'An approximate true damping solution of the flutter equation by determinant iteration.' In: *Journal of Aircraft* 8.11 (1971), pp. 885–889. DOI: 10.2514/3.44311 (cit. on p. 47).

[82] G. Dimitriadis, N. Giannelis, and G. Vio. 'A modal frequency-domain generalised force matrix for the unsteady Vortex Lattice method'. In: *Journal of Fluids and Structures* 76 (2018), pp. 216–228. DOI: 10.1016/j.jfluidstructs.2017.10.010 (cit. on p. 48).

[83] P. C. Chen. 'Damping Perturbation Method for Flutter Solution: The g-Method'. In: *AIAA Journal* 38.9 (2000), pp. 1519–1524. DOI: 10.2514/2.1171 (cit. on p. 48).

[84] H. Güner, D. Thomas, G. Dimitriadis, and V. Terrapon. 'Unsteady aerodynamic modeling methodology based on dynamic mode interpolation for transonic flutter calculations'. In: *Journal of Fluids and Structures* 84 (2019), pp. 218–232. DOI: 10.1016/j.jfluidstructs.2018.11.002 (cit. on p. 48).

[85] F. H. Ling and X. X. Wu. 'Fast Galerkin method and its application to determine periodic solutions of non-linear oscillators'. In: *International Journal of Non-Linear Mechanics* 22.2 (1987), pp. 89–98. ISSN: 0020-7462. DOI: 10.1016/0020-7462(87)90012-6 (cit. on pp. 69, 155).

[86] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. Wiley, 2003. DOI: 10.1002/0471671746 (cit. on p. 72).

[87] Z. Lyu, Z. Xu, and J. R. R. A. Martins. 'Benchmarking optimization algorithms for wing aerodynamic design optimization'. In: *Proceedings of the 8th International Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China*. Vol. 11. 2014 (cit. on pp. 72, 73).

[88] H. H. Khodaparast et al. 'Efficient Worst Case "1-Cosine" Gust Loads Prediction'. In: *Journal of Aeroelasticity and Structural Dynamics* 2.3 (2012), pp. 33–54. ISSN: 1974-5117. DOI: 10.3293/asdj.2012.17 (cit. on p. 73).

[89] K. M. Passino. 'Biomimicry of bacterial foraging for distributed optimization and control'. In: *IEEE Control Systems Magazine* 22.3 (2002), pp. 52–67. DOI: 10.1109/MCS.2002.1004010 (cit. on p. 73).

[90] J. Nocedal and S. J. Wright. *Numerical Optimization*. Ed. by T. V. Mikosch, S. I. Resnick, and S. M. Robinson. 2nd ed. Springer New York, 2006. ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5 (cit. on pp. 73, 77, 81).

[91] E. M. Anderson, F. H. Bhuiyan, D. J. Mavriplis, and R. S. Fertig. 'Adjoint-Based High-Fidelity Aeroelastic Optimization of Wind Turbine Blade for Load Stress Minimization'. In: *2018 Wind Energy Symposium*. American Institute of Aeronautics and Astronautics, 2018. DOI: 10.2514/6.2018-1241 (cit. on p. 73).

[92] Z. Zhang, P. C. Chen, S. Yang, Z. Wang, and Q. Wang. 'Unsteady Aerostructure Coupled Adjoint Method for Flutter Suppression'. In: *AIAA Journal* 53.8 (2015), pp. 2121–2129. DOI: 10.2514/1.j053495 (cit. on pp. 73, 104, 115).

[93] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. 'The complex-step derivative approximation'. In: *ACM Transactions on Mathematical Software* 29.3 (2003), pp. 245–262. DOI: 10.1145/838250.838251 (cit. on pp. 76, 80).

[94] J. Fike and J. Alonso. 'The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations'. In: *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2011. DOI: 10.2514/6.2011-886 (cit. on p. 76).

[95] O. Pironneau. 'On optimum design in fluid mechanics'. In: *Journal of Fluid Mechanics* 64.01 (1974), p. 97. DOI: 10.1017/s0022112074002023 (cit. on p. 76).

[96] A. Jameson. 'Aerodynamic Design via Control Theory'. In: *Journal of Scientific Computing* (1989), pp. 377–401. DOI: 10.1007/978-3-642-83733-3_14 (cit. on pp. 76, 115).

[97] M. B. Giles and N. A. Pierce. 'An Introduction to the Adjoint Approach to Design'. In: *Flow, Turbulence and Combustion* 65.3 (2000), pp. 393–415. ISSN: 1573-1987. DOI: 10.1023/A:1011430410075 (cit. on p. 76).

[98] S. Nadarajah and A. Jameson. 'A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization'. In: *38th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2000. DOI: 10.2514/6.2000-667 (cit. on p. 79).

[99] A. Bueno-Orovio, C. Castro, F. Palacios, and E. Zuazua. 'Continuous Adjoint Approach for the Spalart-Allmaras Model in Aerodynamic Optimization'. In: *AIAA Journal* 50.3 (2012), pp. 631–646. DOI: 10.2514/1.j051307 (cit. on p. 79).

[100] T. D. Economon, F. Palacios, and J. J. Alonso. 'An Unsteady Continuous Adjoint Approach for Aerodynamic Design on Dynamic Meshes'. In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2014. DOI: 10.2514/6.2014-2300 (cit. on pp. 79, 115, 118).

[101] T. D. Economon. 'Optimal Shape Design Using an Unsteady Continuous Adjoint Approach'. PhD thesis. Department of Aeronautics and Astronautics, Stanford University, 2014. URL: http://purl.stanford.edu/tg620qm8104 (cit. on pp. 79, 119).

[102] P. Spalart and S. Allmaras. 'A one-equation turbulence model for aerodynamic flows'. In: *30th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 1992. DOI: 10.2514/6.1992-439 (cit. on p. 79).

[103] J. R. R. A. Martins and J. T. Hwang. 'Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models'. In: *AIAA Journal* 51.11 (2013), pp. 2582–2599. DOI: 10.2514/1.j052184 (cit. on p. 80).

[104] M. Sagebaum, T. Albring, and N. R. Gauger. 'Expression templates for primal value taping in the reverse mode of algorithmic differentiation'. In: *Optimization Methods and Software* 33.4-6 (2018), pp. 1207–1231. DOI: 10.1080/10556788.2018.1471140 (cit. on p. 80).

[105] M. Sagebaum, T. Albring, and N. R. Gauger. 'High-Performance Derivative Computations using CoDiPack'. In: *ACM Transactions on Mathematical Software* 45.4 (2019), pp. 1–26. DOI: 10.1145/3356900 (cit. on pp. 80, 87, 91).

[106] N. M. K. Poon and J. R. R. A. Martins. 'An adaptive approach to constraint aggregation using adjoint sensitivity analysis'. In: *Structural and Multidisciplinary Optimization* 34.1 (2006), pp. 61–73. DOI: 10.1007/s00158-006-0061-7 (cit. on pp. 80, 82).

[107] A. Lambe, G. Kennedy, and J. Martins. 'Multidisciplinary Design Optimization of an Aircraft Wing via a Matrix-Free Approach'. In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2014. DOI: 10.2514/6.2014-2429 (cit. on p. 81).

[108] P. Jansen and R. Perez. 'Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach'. In: *Computers & Structures* 89.13-14 (2011), pp. 1352–1366. DOI: 10.1016/j.compstruc.2011.03.011 (cit. on p. 81).

[109] G. Kreisselmeier and R. Steinhauser. 'Systematic Control Design by Optimizing a Vector Performance Index'. In: *IFAC Proceedings Volumes* 12.7 (1979), pp. 113–117. DOI: 10.1016/s1474-6670(17)65584-8 (cit. on pp. 81, 82).

[110] C. Raspanti, J. Bandoni, and L. Biegler. 'New strategies for flexibility analysis and design under uncertainty'. In: *Computers & Chemical Engineering* 24.9-10 (2000), pp. 2193–2209. DOI: 10.1016/s0098-1354(00)00591-3 (cit. on p. 82).

[111] T. A. Albring, M. Sagebaum, and N. R. Gauger. 'Efficient Aerodynamic Design using the Discrete Adjoint Method in SU2'. In: *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2016. DOI: 10.2514/6.2016-3518 (cit. on p. 87).

[112] R. Bombardieri, R. Sanchez, R. Cavallaro, and N. R. Gauger. 'Towards an open-source framework for aero-structural design and optimization within the SU2 suite'. In: *EUROGEN 2019 Proceedings*. 2019 (cit. on p. 91).

[113] S. Nadarajah and A. Jameson. 'Optimal control of unsteady flows using a time accurate method'. In: *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, 2002. DOI: 10.2514/6.2002-5436 (cit. on pp. 100, 103).

[114] M. Rumpfkeil and D. Zingg. 'A General Framework for the Optimal Control of Unsteady Flows with Applications'. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2007. DOI: 10.2514/6.2007-1128 (cit. on p. 102).

[115] A. Griewank. 'Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation'. In: *Optimization Methods and Software* 1.1 (1992), pp. 35–54. DOI: 10.1080/10556789208805505 (cit. on p. 103).

[116] A. Griewank and A. Walther. 'Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation'. In: *ACM Transactions on Mathematical Software* 26.1 (2000), pp. 19–45. DOI: 10.1145/347837.347846 (cit. on p. 103).

[117] Q. Wang, P. Moin, and G. Iaccarino. 'Minimal Repetition Dynamic Checkpointing Algorithm for Unsteady Adjoint Calculation'. In: *SIAM Journal on Scientific Computing* 31.4 (2009), pp. 2549–2567. DOI: 10.1137/080727890 (cit. on p. 103).

[118] D. I. Papadimitriou. 'A two-step back–forth algorithm for the solution of the unsteady adjoint equations'. In: *International Journal of Computational Fluid Dynamics* 30.3 (2016), pp. 272–284. DOI: 10.1080/10618562.2016.1194978 (cit. on p. 103).

[119] K. Mani and D. J. Mavriplis. 'Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes'. In: *AIAA Journal* 46.6 (2008), pp. 1351–1364. DOI: 10.2514/1.29924 (cit. on pp. 104, 115).

[120] K. E. Jacobson. 'Adjoint-based Aeroelastic Optimization with High-fidelity Time-accurate Analysis'. PhD thesis. School of Aerospace Engineering, Georgia Institute of Technology, 2019. URL: http://hdl.handle.net/1853/61189 (cit. on pp. 104, 105, 118).

[121] K. Jacobson, J. F. Kiviaho, M. J. Smith, and G. Kennedy. 'An Aeroelastic Coupling Framework for Time-accurate Analysis and Optimization'. In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2018. DOI: 10.2514/6.2018-0100 (cit. on pp. 104, 118).

[122] E. J. Nielsen and B. Diskin. 'Discrete Adjoint-Based Design for Unsteady Turbulent Flows on Dynamic Overset Unstructured Grids'. In: *AIAA Journal* 51.6 (2013), pp. 1355–1373. DOI: 10.2514/1.j051859 (cit. on p. 105).

[123] D. N. Srinath and S. Mittal. 'An adjoint method for shape optimization in unsteady viscous flows'. In: *Journal of Computational Physics* 229.6 (2009), pp. 1994–2008. DOI: 10.1016/j.jcp.2009.11.019 (cit. on pp. 105, 106, 117).

[124] J. F. Kiviaho, K. Jacobson, M. J. Smith, and G. Kennedy. 'Application of a Time-Accurate Aeroelastic Coupling Framework to Flutter-Constrained Design Optimization'. In: *2018 Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2018. DOI: 10.2514/6.2018-2932 (cit. on p. 106).

[125] A. Mishra, K. Mani, D. Mavriplis, and J. Sitaraman. 'Time dependent adjoint-based optimization for coupled fluid–structure problems'. In: *Journal of Computational Physics* 292 (2015), pp. 253–271. DOI: 10.1016/j.jcp.2015.03.010 (cit. on p. 106).

[126] S. Choi, K. Lee, M. M. Potsdam, and J. J. Alonso. 'Helicopter Rotor Design Using a Time-Spectral and Adjoint-Based Method'. In: *Journal of Aircraft* 51.2 (2014), pp. 412–423. DOI: 10.2514/1.c031975 (cit. on pp. 106, 115).

[127] E. N. Jacobs, K. E. Ward, and R. M. Pinkerton. *The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel*. Tech. rep. 460. National Advisory Committee for Aeronautics, 1933. URL: https://ntrs.nasa.gov/citations/19930091108 (cit. on p. 114).

[128] E. N. Jacobs and R. M. Pinkerton. *Tests in the variable-density wind tunnel of related airfoils having the maximum camber unusually far forward*. Tech. rep. National Advisory Committee for Aeronautics, 1935. URL: https://ntrs.nasa.gov/citations/19930091610 (cit. on p. 114).

[129] H. Sobieczky. 'Parametric Airfoils and Wings'. In: *Recent Development of Aerodynamic Design Methodologies*. Ed. by K. Fujii and G. S. Dulikravich. Vol. 65. Notes on Numerical Fluid Mechanics. Vieweg+Teubner Verlag, 1999, pp. 71–87. ISBN: 9783322899521. DOI: 10.1007/978-3-322-89952-1_4 (cit. on p. 114).

[130] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. 'Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation'. In: *34th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 1996. DOI: 10.2514/6.1996-94 (cit. on pp. 114–116).

[131] R. M. Hicks and P. A. Henne. 'Wing Design by Numerical Optimization'. In: *Journal of Aircraft* 15.7 (1978), pp. 407–412. DOI: 10.2514/3.58379 (cit. on p. 115).

[132] P. Castonguay and S. Nadarajah. 'Effect of Shape Parameterization on Aerodynamic Shape Optimization'. In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2007. DOI: 10.2514/6.2007-59 (cit. on p. 115).

[133] J. Reuther and A. Jameson. *A comparison of design variables for control theory based airfoil optimization*. Tech. rep. Research Institute for Advanced Computer Science, 1995 (cit. on p. 116).

[134] J. Lepine, F. Guibault, J.-Y. Trepanier, and F. Pepin. 'Optimized Nonuniform Rational B-Spline Geometrical Representation for Aerodynamic Design of Wings'. In: *AIAA Journal* 39.11 (2001), pp. 2033–2041. ISSN: 1533-385X. DOI: 10.2514/2.1206 (cit. on pp. 116, 117).

[135] T. W. Sederberg and S. R. Parry. 'Free-form deformation of solid geometric models'. In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86*. ACM Press, 1986. DOI: 10.1145/15922.15903 (cit. on p. 117).

[136] H. J. Lamousin and N. N. Waggenspack. 'NURBS-based free-form deformations'. In: *IEEE Computer Graphics and Applications* 14.6 (1994), pp. 59–65. ISSN: 0272-1716. DOI: 10.1109/38.329096 (cit. on pp. 117, 118).

[137] J. Samareh. 'Aerodynamic Shape Optimization Based on Free-Form Deformation'. In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2004. DOI: 10.2514/6.2004-4630 (cit. on p. 118).

[138] J. Samareh. 'Multidisciplinary aerodynamic-structural shape optimization using deformation (MASSOUD)'. In: *8th Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, 2000. DOI: 10.2514/6.2000-4911 (cit. on p. 118).

[139] J. A. Samareh. 'Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization'. In: *AIAA Journal* 39.5 (2001), pp. 877–884. DOI: 10.2514/2.1391 (cit. on p. 118).

[140] J. F. Kiviaho, K. Jacobson, M. J. Smith, and G. Kennedy. 'A Robust and Flexible Coupling Framework for Aeroelastic Analysis and Optimization'. In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, 2017. DOI: 10.2514/6.2017-4144 (cit. on p. 118).

[141] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. 'A Limited Memory Algorithm for Bound Constrained Optimization'. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: 10.1137/0916069 (cit. on pp. 131, 136).

[142] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. 'Algorithm 778: L-BFGS-B'. In: *ACM Transactions on Mathematical Software* 23.4 (1997), pp. 550–560. DOI: 10.1145/279232.279236 (cit. on pp. 131, 136).

[143] D. Kraft. *A software package for sequential quadratic programming*. Tech. rep. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1988 (cit. on p. 140).

[144] J. A. C. Dias, C. D. Marqui, and A. Erturk. 'Hybrid piezoelectric-inductive flow energy harvesting and dimensionless electroaeroelastic analysis for scaling'. In: *Applied Physics Letters* 102.4 (2013), p. 044101. DOI: 10.1063/1.4789433 (cit. on pp. 144, 157).

[145] W. Ning and L. He. 'Computation of Unsteady Flows Around Oscillating Blades Using Linear and Nonlinear Harmonic Euler Methods'. In: *Volume 4: Manufacturing Materials and Metallurgy; Ceramics; Structures and Dynamics; Controls, Diagnostics and Instrumentation; Education; IGTI Scholar Award*. GT1997. American Society of Mechanical Engineers, 1997. DOI: 10.1115/97-gt-229 (cit. on p. 155).

[146] M. McMullen, A. Jameson, and J. Alonso. 'Demonstration of Nonlinear Frequency Domain Methods'. In: *AIAA Journal* 44.7 (2006), pp. 1428–1435. DOI: 10.2514/1.15127 (cit. on p. 155).

[147] F. E. Eastep and J. J. Olsen. 'Transonic Flutter Analysis of a Rectangular Wing with Conventional Airfoil Sections'. In: *AIAA Journal* 18.10 (1980), pp. 1159–1164. ISSN: 1533-385X. DOI: 10.2514/3.50866 (cit. on p. 156).

[148] P. S. Beran, N. S. Khot, F. E. Eastep, R. D. Snyder, and J. V. Zweber. 'The Dependence of Store-Induced Limit-Cycle Oscillation Predictions on Modelling Fidelity'. In: *RTO Applied Vehicle Technology Panel Symposium*. 2002. URL: https://apps.dtic.mil/sti/citations/tr/ADP014183 (cit. on p. 156).

[149] P. S. Beran, T. W. Straganac, K. Kim, and C. Nichkawde. 'Studies of store-induced limit-cycle oscillations using a model with full system nonlinearities'. In: *Nonlinear Dynamics* 37.4 (2004), pp. 323–339. ISSN: 0924-090X. DOI: 10.1023/b:nody.0000045544.96418.bf (cit. on p. 156).

[150] A. Abdelkefi. 'Aeroelastic energy harvesting: A review'. In: *International Journal of Engineering Science* 100 (2016), pp. 112–135. DOI: 10.1016/j.ijengsci.2015.10.006 (cit. on p. 157).

[151] A. Erturk, W. G. R. Vieira, C. D. Marqui, and D. J. Inman. 'On the energy harvesting potential of piezoaeroelastic systems'. In: *Applied Physics Letters* 96.18 (2010), p. 184103. DOI: 10.1063/1.3427405 (cit. on p. 157).

[152] J.-S. Bae and D. J. Inman. 'Aeroelastic characteristics of linear and nonlinear piezo-aeroelastic energy harvester'. In: *Journal of Intelligent Material Systems and Structures* 25.4 (2013), pp. 401–416. DOI: 10.1177/1045389x13498312 (cit. on p. 157).

[153] M. A. Bouhlel and J. R. R. A. Martins. 'Gradient-enhanced kriging for high-dimensional problems'. In: *Engineering with Computers* 35.1 (2018), pp. 157–173. ISSN: 1435-5663. DOI: 10.1007/s00366-018-0590-x (cit. on p. 157).

[154] M. A. Bouhlel et al. 'A Python surrogate modeling framework with derivatives'. In: *Advances in Engineering Software* 135 (2019), p. 102662. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2019.03.005 (cit. on p. 157).

Appendix A

Implementation of the harmonic balance fluid-structure interaction algorithm in CUPyDO

The extension of SU2, CUPyDO and the rigid body motion integrator in order to implement frequency-domain techniques and frequency iteration represented an important part of this project. This extension required a significant amount of implementation work and presented several challenges. Some of these challenges were not apparent at first, only becoming visible after a naïve approach failed. One objective of the implementation task was to affect as little as possible the already-existing code and to attempt to preserve its behaviour where modification was needed. In the present appendix, the work performed in for this thesis order to enable the calculation of the partitioned fluid-structure interaction solution using a harmonic balance with frequency iteration technique is presented.

A.1 Extension of SU2

The SU2 CFD suite [8] already implements the time-domain harmonic balance (TDHB) method [9]. However, the applications for which it had been used did not require mesh deformation or programmatically modifying the frequency. Hence, the software had to be modified in order to be able to do so. There are three main components that were added for the present thesis: the extension of the Python application programming interface (API) to work with the TDHB, mesh deformation for the TDHB and extension of the API to modify the frequency.

A.1.1 Time-domain harmonic balance API

The SU2 API was first developed to interact with the base single-zone fluid driver, `CSinglezoneDriver`. This driver is in charge of running the simulation. However, time-domain harmonic balance calculations use a different driver class, `CHBDriver`. Furthermore, the TDHB driver for N_h harmonics uses $2 \cdot N_h + 1$ different solvers, stored in a container. This container includes four indices: zone (for monolithic multi-physics problems), time instance, mesh (for multi-grid solutions) and physics solver. If not using the TDHB method, the value of the time instance index is 0, so many functions use that as a default. Two important routines for fluid-structure interaction calculations are those that impose boundary displacements and those that extract boundary forces. These two API functions were modified for this project by adding an optional parameter defining which instance was being targeted, so that the correct solver was selected. Otherwise, the value of this optional parameter was set to 0 in order to maintain the old behaviour.

A.1.2 Mesh deformation

Adapting the TDHB fluid driver to take into account mesh deformation required two main modifications: adapting the calculation of source terms according to the procedure described in Sec. 2.4.3 and calculating the velocity of each grid point. Since the TDHB approach in SU2 was designed for non-deforming meshes, the volume of each cell was assumed to be fixed. Therefore, the source term was calculated as described in Algorithm 1. The correct algorithm for meshes with variable cell volumes as implemented for this project is instead described in Algorithm 2. In those algorithms, N_h is the number of harmonics; $N_i = 2 \cdot N_h + 1$ is the number of time instances; $nNode$ is the total number of nodes; $nVar$ is the number of state variables; $Source$ is the matrix with the values of the TDHB source terms; D represents matrix $\mathbf{D}$, the TDHB time derivative matrix introduced in Eq. (2.58); U is the matrix of state variables and V is the matrix with the control volumes of each node in each time instance. The iterating variables are i for the nodes, j for the state variables, k for the receiving time instance and l for the donor time instance. In Algorithm 2, an extra variable with the product of the volume and the state variables is introduced, $UVol$. Note that in both cases there is a large number of operations in the innermost loop.

While the difference between the two procedures is subtle, using Algorithm 1 instead of Algorithm 2 results in different flow fields. In particular, for the test case shown in Sec. 2.7, Algorithm 1 produced a significantly asymmetric flow field. This was unexpected for a low-amplitude test case. Furthermore, increasing the number of harmonics did not lead to convergence towards the time-marching solution, unlike what happens when using Algorithm 2.

Algorithm 1 Original SU2 algorithm for TDHB source term calculation

$i \leftarrow 0$, $N_i \leftarrow 2 \cdot N_h + 1$
while $i < nNode$ **do**
  $j \leftarrow 0$
  while $j < nVar$ **do**
    $k \leftarrow 0$
    while $k < N_i$ **do**
      Reset source term: $Source[i][j][k] \leftarrow 0$
      $l \leftarrow 0$
      while $l < N_i$ **do**
        Add contribution of lth instance on kth instance:
        $Source[i][j][k] \leftarrow Source[i][j][k] + D[k][l] \times U[i][j][l]$
        $l \leftarrow l + 1$
      end while
      Multiply source by control volume:
      $Source[i][j][k] \leftarrow Source[i][j][k] \times V[i][0]$
      $k \leftarrow k + 1$
    end while
    $j \leftarrow j + 1$
  end while
  $i \leftarrow i + 1$
end while

Regarding the calculation of mesh velocities, they were implemented for this thesis as part of the Python API in SU2. To do so, the mesh update API functions were modified in the case of harmonic balance calculations. They follow the expression introduced in Eq. (2.63).

A.1.3 Frequency API function

The TDHB algorithm used in SU2 is quite general, but it did not allow for programmatically modifying the frequency before being extended for this project. It uses as parameters to define the TDHB matrices the period and the frequencies studied. Therefore, a new API function was coded that allows external programs to change the frequencies and period. Its structure follows Algorithm 3. There, ω_{ext} is the imposed frequency, N_h is the number of harmonics studied, $\boldsymbol{\omega}$ is the $2 \cdot N_h + 1$-length vector with all frequencies studied and i iterates over the harmonics.

Algorithm 3 assumes that there is exactly one base frequency. Therefore, it cannot be applied to cases where there are several base frequencies. For one such

Algorithm 2 Modified algorithm for TDHB source term calculation

$i \leftarrow 0$, $N_i \leftarrow 2 \cdot N_h + 1$
while $i < nNode$ **do**
  $j \leftarrow 0$
  while $j < nVar$ **do**
    $k \leftarrow 0$
    while $k < N_i$ **do**
      Multiply U by control volume:
      $UVol[k] \leftarrow U[i][j][k] \times V[i][k]$
      $k \leftarrow k + 1$
    end while
    $k \leftarrow 0$
    while $k < N_i$ **do**
      Reset source term: $Source[i][j][k] \leftarrow 0$
      $l \leftarrow 0$
      while $l < N_i$ **do**
        Add contribution of lth instance on kth instance:
        $Source[i][j][k] \leftarrow Source[i][j][k] + D[k][l] \times UVol[l]$
        $l \leftarrow l + 1$
      end while
      $k \leftarrow k + 1$
    end while
    $j \leftarrow j + 1$
  end while
  $i \leftarrow i + 1$
end while

Algorithm 3 Frequency updating in SU2

Set the value of the base frequency: $\omega \leftarrow \omega_{ext}$
Set the zeroth harmonic's frequency: $\boldsymbol{\omega}[0] \leftarrow 0$
Set the time period: $T \leftarrow \frac{2\pi}{\omega}$
$i \leftarrow 1$
while $i \leq N_h$ **do**
  Set positive frequency: $\boldsymbol{\omega}[i] \leftarrow i \times \omega$
  Set negative frequency: $\boldsymbol{\omega}[i + N_h] \leftarrow -i \times \omega$
  $i \leftarrow i + 1$
end while
Update HB matrices: `Compute_HBOperator()`

example see the original TDHB work in SU2 by Rubino et al., with two different frequencies in a turbomachinery application [9].

A.2 Extension of CUPyDO

CUPyDO is an open-source Python code that couples fluid and structural solvers to solve fluid-structure interaction problems. It has also been used for conjugate heat transfer applications. Its architecture relies on the use of classes, called interfaces, that are adapted to each code. They handle the initialisation of the solvers and allow the transfer of data to and from the fluid and structural solvers. This results in a very flexible architecture: extending CUPyDO to incorporate a new solver is simple, only a new interface has to be added. Furthermore, it is not very invasive on the codes. The API functions to extract the data and impose the Dirichlet or Neumann boundary conditions have to be provided, but the fluid or structural solver does not have to be modified otherwise.

CUPyDO implements both Block-Gauss-Seidel and quasi-Newton algorithms. It also handles data transfer and interpolation. In order to implement the harmonic balance method, CUPyDO had to be extended in two main ways: including the time instances and transferring the change in base frequency of the problem.

A.2.1 Inclusion of several time instances

The donor and receiver codes have to use the same structure. That is, both codes have to operate in either the frequency or the time domain. Otherwise, CUPyDO is agnostic as to what kinds of data are being transferred. It can, in principle, transfer both time-domain and frequency-domain data with no further modification, but in this thesis it is only used in the time domain.

For steady calculations, the size of the matrices being transferred is $n_n \times n_{dim}$, where n_n is the number of nodes at each boundary and n_{dim} is the number of dimensions (either 2 or 3). For the harmonic balance, the size is $n_n \times n_{dim} \cdot N_i$, where N_i is the number of time instances (if the solutions were in the frequency domain, the harmonics would be used instead). This simplifies the interpolation and transfer of data in the harmonic balance approach. Interpolation of data from one boundary to the other is local to each time instance.

There are other functions that handle the extraction of data and the imposition of boundary conditions for each solver. Originally, CUPyDO used vectors of length n_n for each of the x, y and z components of the loads and displacements. The functions were modified for both SU2 and the rigid body motion integrator to handle multiple time instances, with the input being changed to $n_n \times N_i$ matrices.

For parallel solvers using OpenMPI data transfer to and from multiple processors also has to be considered. In SU2, the adjudication of nodes to processors is independent of the time instance, so the process is unaffected.

A.2.2 Transfer of change in frequency

A function that obtains the change in frequency, $\Delta\omega$, from the structural solver and another one that transfers it to the fluid solver has to be included in each interface. It was implemented for the rigid body motion structural solver and for the SU2 fluid solver. Underrelaxation can be applied to $\Delta\omega$ in order to more easily converge. Currently, the value of the underrelaxation parameter is equal for the frequency and the displacements.

A.3 Extension of the rigid body motion integrator

The mathematics of the frequency-domain technique that was finally chosen are explained in more detail in Appendix C. In the present section, the choice of using this approach instead of the time-domain harmonic balance is justified first. Then, how the method was implemented in this project is presented.

A.3.1 Justification of the frequency-domain technique

The unsteady equation of the motion of an undamped, one degree of freedom rigid body with mass m and a structural restoring force provided by a linear spring with stiffness k is

$$\ddot{x} + \frac{k}{m}x = \frac{F(t)}{m}, \tag{A.1}$$

where x is the displacement, $\ddot{x}$ is the acceleration and F is the external force. There is a second-order time derivative.

The time-domain harmonic balance Jacobian is $\partial\mathbf{R}^*/\partial\mathbf{x}^*$, with $\mathbf{R}^*$ being the residual and $\mathbf{x}^*$ the value of the state variables at each time instance. Assume that the problem, $\mathbf{R}$, is linear on x. If the $2 \cdot N_h + 1$ time instances are solved separately, the matrix used on the left-hand side is $\partial\mathbf{R}^*/\partial\mathbf{x}^* = k/m\mathbf{I}$, with $\mathbf{I}$ being the identity matrix. The acceleration appears as a source term on the right-hand side of each time instance. Transform the matrix to the frequency domain

$$\frac{\partial\hat{\mathbf{R}}}{\partial\hat{\mathbf{x}}} = \frac{\partial\hat{\mathbf{R}}}{\partial\mathbf{R}^*}\frac{\partial\mathbf{R}^*}{\partial\mathbf{x}^*}\frac{\partial\mathbf{x}^*}{\partial\hat{\mathbf{x}}} = \frac{k}{m}\mathbf{E}^{-1}\mathbf{I}\mathbf{E} = \frac{k}{m}\mathbf{I}, \tag{A.2}$$

where $\hat{\mathbf{R}}$ is the frequency-domain residual, $\hat{\mathbf{x}}$ is the vector of frequency-domain state variables, $\mathbf{E}$ is the discrete Fourier transform matrix introduced in Eq. (2.53) and $\mathbf{E}^{-1}$ is its inverse. The time-domain and frequency-domain matrices are equal.

For the structural problem in Eq. (A.1), the Jacobian of the frequency-domain equations with a base frequency of ω and using N_h harmonics is

$$\frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{x}}} = \begin{bmatrix} \frac{k}{m} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{k}{m} - \omega^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{k}{m} - \omega^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{k}{m} - (N_h\omega)^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{k}{m} - (N_h\omega)^2 \end{bmatrix}. \tag{A.3}$$

The matrix is diagonal, but the sign of the terms will change if the frequency of any harmonic is larger than the natural frequency of the system. If $N_h\omega > \sqrt{k/m}$, a structural TDHB code which solves each instance separately using the matrix in Eq. (A.2) will diverge. The time instances have to be solved all at once, unlike for the fluid solver.

The main advantage of the time-domain harmonic balance method is that the steady solver can be re-used. If the time instances cannot be solved independently, the steady solver has to be modified, regardless. Therefore, considering that implementing the frequency-iteration technique was easier in the frequency domain, it was decided to use a frequency-domain approach with a state-space model to reduce the differential equation from second order to first order. This transformation was also tested for the time-domain harmonic balance method with separate time instances during this project. Like the original second-order TDHB approach, it did not converge.

A.3.2 Implementation of the frequency-domain solver

The structure of the frequency-domain solver was adapted from that of the already-existing solvers for this thesis. Before any iterations, the restart files with the initial guesses for the displacements are read. Unlike for time-marching solvers, this step is required for the frequency-domain approach. Furthermore, the discrete Fourier transform ($\mathbf{E}$, $\mathbf{E}^{-1}$) and frequency-domain time-derivative matrices ($1/\omega\mathbf{A}$) are calculated at this point, since they can be re-used.

The calculation includes a few steps. First, a matrix of size $4 \cdot (2 \cdot N_h + 1) \times 4 \cdot (2 \cdot N_h + 1)$ is created first for the two-degree-of-freedom aerofoil. It is the $\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}$ term in Eq. (C.6), which is used in order to calculate the residual of the problem,

$$\hat{\mathbf{R}} = \hat{\mathbf{F}} - \left(\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}\right)\hat{\mathbf{x}}_0, \tag{A.4}$$

where $\hat{\mathbf{M}}$ is the frequency-domain mass matrix, $\hat{\mathbf{K}}$ is the frequency-domain stiffness matrix, $\hat{\mathbf{F}}$ are the forces in the frequency domain and $\hat{\mathbf{x}}_0$ is the current vector of state variables.

Then, the Jacobian of the problem is calculated, based on matrix $\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}$. The fixed degree of freedom's column is skipped, since its value is 0. Then, the last column of the Jacobian is filled with the partial derivative of the residual with respect to the base frequency, which is calculated using Eq. (C.12) in Appendix C. After that, the system of equations is solved. The change in the solution is

$$\begin{bmatrix} \Delta\hat{\mathbf{x}} \\ \Delta\omega \end{bmatrix} = \begin{bmatrix} \dfrac{\partial\hat{\mathbf{R}}}{\partial\hat{\mathbf{x}}} & \dfrac{\partial\hat{\mathbf{R}}}{\partial\omega} \end{bmatrix}^{-1} \hat{\mathbf{R}}. \tag{A.5}$$

The new values of the state variables are calculated by adding the results vector, on the left in Eq. (A.5), to the state vector at the start. Then, the solution is transferred from the frequency domain to the time domain so that the boundary displacements can be obtained at each of the $2 \cdot N_h + 1$ time instances.

Appendix B

Implementation of the adjoint harmonic balance fluid-structure interaction algorithm in CUPyDO

Within the present appendix the implementation work that was required for the use of the coupled adjoint method in CUPyDO is detailed. Except in those cases where it is explicitly noted otherwise, the implementation was performed for the present work. First, the extensions of the steady adjoint codes are introduced. Then, the changes required in order to use the harmonic balance adjoint method are described.

B.1 Steady adjoint

Three components were required in order to have steady adjoint calculations for fluid-structure interaction (FSI) problems: the fluid solver (SU2), the coupler (CUPyDO) and the structural solver (SU2 and pyBeam). The solvers were chosen because they already had adjoint and FSI capabilities, so they would require less extensive modifications.

B.1.1 Changes to SU2

The work required in order to transfer the steady adjoint data to and from SU2 had already been performed before the project. It was already included in the SU2 Python interface. SU2 has Python functions that allow external codes to impose adjoint boundary conditions and to extract the gradients required for adjoint coupling.

B.1.2 Extension of CUPyDO

CUPyDO relies on solver-specific classes, called interfaces, to extract and transfer the data that it needs. This is a highly flexible and non-intrusive approach, which makes extending the coupler to work with new solvers easy. These interfaces inherit from a base class that has all the methods used by the code to transfer the information around.

For this project, new base classes had to be written to implement the adjoint method. These base classes were defined for generic fluid and structural adjoint solvers. New interfaces were written as well for the structural solvers and the adjoint version of SU2. They also inherited from the original classes, in order to set up the physical boundary conditions of the problem, that is, displacements and forces.

Adjoint fluid-structure interaction algorithm

CUPyDO already included a coupling algorithm with constant underrelaxation factor for direct fluid-structure interaction problems. An adjoint FSI algorithm with constant underrelaxation factor was implemented for this project taking advantage of the pre-existing building blocks. First, the case has to be initialised. The solution obtained by the direct solver is loaded from disk and the boundary conditions (loads and displacements) are set for the adjoint solver. The fluid mesh is deformed based on the corresponding boundary displacements. After initialisation of the problem, only the fluid and structural adjoint gradients are interpolated and transferred between the solvers for each adjoint FSI iteration.

Extraction of adjoint information

Two new functions to extract and impose the adjoint boundary conditions described in Sec. 3.6 were added to the adjoint interfaces. The adjoint boundary conditions are gradients of the single-field augmented objective function introduced in Eq. (3.41). They behave like the functions to extract and impose boundary forces and displacements, but for the adjoint problem.

Adjoint gradient interpolation and transfer

It is shown in Sec. 3.6.1 that if using conservative interpolation methods, the direct interpolation matrices can be re-used for the adjoint problem. In CUPyDO, that means that the interpolation functions defined for the direct approach can be reused as well: there is no need to write a specific interpolation approach for the adjoint gradients at the fluid/solid boundary. For consistent approaches, however, the transposed matrices have to be used instead. The transfer of the adjoint data is modelled on the one used for direct solutions.

Value of objective functions and gradients

In order to perform the optimisations in Chapter 5, the values of the objective functions, constraints and the corresponding gradients needed to be provided to the optimiser. Therefore, a way to extract their values from the fluid and structural solvers was included in CUPyDO for this project.

The objective function extractors were implemented during this thesis for SU2 and the rigid body motion integrator. The corresponding objective functions have to be defined in the configuration files. In this work, the SU2 fluid objective function is set to the drag coefficient multiplied by a very small (1×10^{-12}) number if the chosen objective function is structural. For the rigid body motion integrator, a "fluid" objective function that avoids this issue by setting $J_s = 0$ is included.

The extraction of the gradients was only implemented for the rigid body motion integrator used for the harmonic balance test cases. In that case, an array with the shape design variables is included. In the other codes, the gradients are extracted from output files in the working directory. They were used in order to verify the adjoint approach. The deformation due to the shape design variables is imposed on the rigid body motion integrator as well. It is transferred to the fluid solver like any other boundary displacement.

B.1.3 Structural adjoint solvers

Two different structural solvers were used for steady problems: the structural solver included in the SU2 computational fluid dynamics suite and a beam solver. They were chosen to demonstrate different aspects of the coupling technique: the beam solver required using non-matching meshes interpolation while SU2 allowed for more complex structural modelling. In both cases, the work to extract adjoint gradient and impose adjoint boundary conditions had already been performed.

However, for SU2's structural solver a function to identify the boundary with the fluid was added for this thesis. This work was on the Python interface, since there were internal tools to mark this boundary already.

B.2 Harmonic balance adjoint

The harmonic balance adjoint method is an efficient alternative to compute gradients of time-periodic FSI problems. In order to implement the technique, the three codes used (SU2 for the fluid, CUPyDO as the coupler and the rigid body motion integrator for the structure) were modified.

B.2.1 Implementation of harmonic balance fluid adjoint solver in SU2

SU2 implements the adjoint method by means of algorithmic differentiation (AD) using CoDiPack. While the standard fluid driver was already differentiated, the harmonic balance-based one was not. The harmonic balance solver uses the time-domain harmonic balance technique. It uses $2 \cdot N_h + 1$ time instances, where N_h is the number of harmonics studied.

The harmonic balance adjoint was implemented for this project by creating a modified version of the already-existing `CDiscAdjSinglezoneDriver` class in order to write a harmonic balance equivalent. There are three changes that had to be included: the number of time instances, the time-domain harmonic balance source term and the frequency gradient. Contrary to the original adjoint solver, which only had one time instance, the harmonic balance adjoint process iterates over the $2 \cdot N_h + 1$ time instances of the direct solution. The source term calculation is recorded by the AD code so that its effect on the solution can be taken into account. The source term is calculated using the method in Algorithm 2. The calculation of matrix $\mathbf{D}$, the TDHB time derivative matrix, was recorded as well. It depends on the base frequency, so this recording is required in order to calculate the frequency gradient.

Frequency gradient calculation

As explained in Sec. 4.3.3, for the proposed approach the fluid solver has to calculate the gradients with respect to the frequency. It uses the AD methods in order to perform this calculation. The base frequency is registered as an input of the problem. Then, the frequency-setter function described in Algorithm 3 is modified in order to let CoDiPack record the calculations. This allows the AD code to calculate the value of the gradient of the objective function with respect to the externally-imposed frequency. Finally, this value is exposed to the Python interface in order to transfer it to the structural solver.

B.2.2 Additional requirements in CUPyDO

Time instances for the adjoint

The extensions implemented for the FSI harmonic balance calculations also applied to adjoint harmonic balance calculations. The sizes of the adjoint data matrices were extended to work with multiple time instances. Therefore, no extra work was required beyond that needed for the direct harmonic balance.

Frequency gradients

A frequency gradient extractor was introduced for this project in SU2's interface. Then, the frequency gradient was applied with underrelaxation to the structural solver, the rigid body motion integrator.

B.2.3 Rigid body motion integrator

The adjoint equations for the rigid body motion integrator are explained in Sec. 4.3.2, with some further details in Appendix D. The influence of the pitch on the moment, described in Eq. (4.46), has to be included in the adjoint equations. Otherwise, the gradients will be wrong. For the case studied in Sec. 4.5, when not including the term in Eq. (4.46) the signs of the gradients were correct. However, their magnitudes were overpredicted by approximately 25%.

Adjoint frequency-domain solver

There are two main differences between the direct and the adjoint solvers: the introduction of the transposed discrete Fourier transform matrices and the influence of the influence of the pitch on the moment. Both changes are related, since the transpose of the inverse Fourier transform matrix is used to calculate the second term. Otherwise, the calculation of the Jacobian is identical to the one described in Sec. A.3.2. The right-hand side is calculated by adding the fluid terms and the dependence of the objective function on the state variables. Then, the system is solved the new values of the adjoint variables.

Implementation of shape design variables

The Hicks-Henne bumps were implemented during this project in the structural solver. They require two parameters: whether they are on the upper or lower side of the aerofoil and the position of the peak. The definition of parameters is managed at the Python level in order to enable easier changes to the distribution of the peaks. Furthermore, including it in the rigid body motion integrator's configuration file would require writing a new parser for only this option. However, the number of design variables is required in the configuration file in order to allocate three design-related vectors: amplitude, position and side of the bumps.

The boundary is deformed based on the value of the design variables, scaled by the chord. The base shape, which is read from an SU2 mesh file, is modified before the displacements are applied, so that the latter are calculated over the deformed shape. This order of operations is important for the pitch degree of freedom. In order

to avoid issues at the leading edge, nodes with a very small chordwise coordinate are ignored. This matches the SU2 definition of the Hicks-Henne bumps.

The deformation is transferred to the fluid solver together with the boundary displacements for each time instance. This greatly simplifies the shape design process: it takes advantage of the mesh deformation step and avoids a costly remeshing procedure. While in this case the resulting mesh was good enough, for more complex setups a remeshing step might be needed.

Structural gradients

The expression of the gradients can be found in Appendix D. The calculations are performed after obtaining the adjoint variables for every FSI adjoint iteration. They are implemented as functions, which can then be called by the structural solver itself in order to output the gradients or by the optimiser to be able to perform the optimisation.

Appendix C

Derivation of a frequency-domain rigid body motion integrator

C.1 Derivation of the equations

Start from the standard dimensional aeroelastic system of equations for a two-degree-of-freedom pitch and plunge aerofoil

$$\begin{cases} S_\alpha \cdot \ddot{\alpha} + m \cdot \ddot{h} + c_h \cdot \dot{h} + k_h \cdot h & = -L \\ S_\alpha \cdot \ddot{h} + I_\alpha \cdot \ddot{\alpha} + c_\alpha \cdot \dot{\alpha} + k_\alpha \cdot \alpha & = M \end{cases}, \tag{C.1}$$

where $S_\alpha = m \cdot (x_f - x_{CG})$ is the dimensional static imbalance, m is the mass of the aerofoil, c_h is the plunge damping, k_h is the plunge stiffness, I_α is the moment of inertia around the flexural axis, c_α is the pitch damping, k_α is the pitch stiffness, h and α are the plunge and pitch displacements according to the convention defined in Fig. 2.9. Rewrite Eq. (C.1) in a matrix form,

$$\begin{bmatrix} m & S_\alpha \\ S_\alpha & I_\alpha \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \ddot{h} \end{bmatrix} + \begin{bmatrix} c_h & 0 \\ 0 & c_\alpha \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{h} \end{bmatrix} + \begin{bmatrix} k_h & 0 \\ 0 & k_\alpha \end{bmatrix} \begin{bmatrix} \alpha \\ h \end{bmatrix} - \begin{bmatrix} -L \\ M \end{bmatrix} = 0. \tag{C.2}$$

Unlike Eq. (2.38), this equation has a second derivative term. By writing it as a state-space model,

$$\mathbf{x} = \begin{bmatrix} \dot{\alpha} \\ \dot{h} \\ \alpha \\ h \end{bmatrix}; \quad \begin{bmatrix} m & S_\alpha & 0 & 0 \\ S_\alpha & I_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} c_h & 0 & k_h & 0 \\ 0 & c_\alpha & 0 & k_\alpha \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \mathbf{x} - \begin{bmatrix} -L \\ M \\ 0 \\ 0 \end{bmatrix} = 0, \tag{C.3}$$

the order of the equations of motion is reduced to 1. However, this system of equations has twice the number of degrees of freedom. It has a mass and a structural

restoring force matrix,

$$\mathbf{M} = \begin{bmatrix} m & S_\alpha & 0 & 0 \\ S_\alpha & I_\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{K} = \begin{bmatrix} c_h & 0 & k_h & 0 \\ 0 & c_\alpha & 0 & k_\alpha \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}. \tag{C.4}$$

Note that the structural restoring force matrix includes both stiffness and damping contributions.

Extend the mass and structural restoring force matrices to take into account the $2 \cdot N_h + 1$ time instances,

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{M} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{M} \end{bmatrix} \qquad \mathbf{K}^* = \begin{bmatrix} \mathbf{K} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{K} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{K} \end{bmatrix}, \tag{C.5}$$

where $\mathbf{0}$ is an $m \times m$ zero matrix. Since the problem is linear, the frequency-domain matrices are equal to the time-domain matrices: $\hat{\mathbf{M}} = \mathbf{M}^*$, $\hat{\mathbf{K}} = \mathbf{K}^*$. In this case, the number of degrees of freedom is $m = 4$. Substitute the frequency-domain matrices and operate

$$\left(\hat{\mathbf{M}}\mathbf{A} + \hat{\mathbf{K}}\right)\hat{\mathbf{x}} - \mathbf{E}\begin{bmatrix} -L \\ M \\ 0 \\ 0 \end{bmatrix} = 0, \tag{C.6}$$

where $\hat{\mathbf{x}}$ is the vector of displacements and velocity amplitudes, $\mathbf{A}$ is the frequency-domain time derivative matrix in Eq. (2.55) and $\mathbf{E}$ is the Fourier transform operator described in Eq. (2.53). This is the frequency-domain equation solved by the code.

C.2 Application of displacements

In the present implementation, the displacements are applied and transferred in the time domain. Therefore, the first step is transforming the frequency-domain displacements, $\hat{\mathbf{x}}$, to the time domain

$$\mathbf{x} = \mathbf{E}^{-1}\hat{\mathbf{x}}. \tag{C.7}$$

The time-domain displacement vector's definition appears in Eq. (C.3).

The new coordinates of node n at the boundary are given by a rotation and a displacement on the y axis. The expression of these coordinates is

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}\begin{bmatrix} x_{n,0} - x_{c,0} \\ y_{n,0} - y_{c,0} \end{bmatrix} + \begin{bmatrix} x_{c,0} \\ y_{c,0} \end{bmatrix} + \begin{bmatrix} 0 \\ -h \end{bmatrix}, \tag{C.8}$$

where $x_{n,0}$ and $y_{n,0}$ are the original coordinates of node n and $x_{c,0}$ and $y_{c,0}$ are the original coordinates of the centre of rotation.

C.3 Calculation of loads

The solid solver requires two kinds of loads: the lift and the moment around the centre of rotation. Both can be obtained from the x and y components of the forces applied in each boundary node, F_x and F_y.

$$L = \sum_{n=0}^{N_n} F_y^n \tag{C.9}$$

$$M = \sum_{n=0}^{N_n} F_y^n \cdot (x_n - x_c) + F_x^n \cdot (y_n - y_c)\,, \tag{C.10}$$

where N_n is the number of nodes at the boundary, F^n is the force applied on node n, x_n and y_n are the x and y coordinates of node n and x_c and y_c are those of the centre of rotation.

The position of the nodes appears in the calculation of the moment, as shown in Eq. (C.10). As described in the previous section, the position of the nodes depends on the value of the pitching and plunging motions. Therefore, there is a dependence of the value of the loads on the displacements.

C.4 Phase-fixing and frequency-iteration procedure

A combined technique for phase-fixing and frequency-iteration was presented in Sec. 2.5.3. This technique has been applied to the rigid-body motion solver. In order to do so, the derivative of the structural equations with respect to the motion frequency, $\frac{\partial \mathcal{S}}{\partial \omega}$, has to be obtained.

Differentiating the left-hand side of Eq. (C.6)

$$\frac{\partial \mathcal{S}}{\partial \omega} = \frac{\partial}{\partial \omega}\left(\hat{\mathbf{M}}\mathbf{A}\hat{\mathbf{x}}\right) + \frac{\partial}{\partial \omega}\left(\hat{\mathbf{K}}\hat{\mathbf{x}}\right). \tag{C.11}$$

Of these terms, only the time-derivative matrix, $\mathbf{A}$, depends on the frequency. From Eq. (2.55), it is a linear function of ω. Therefore, the derivative of the structural equations is

$$\frac{\partial \mathcal{S}}{\partial \omega} = \frac{1}{\omega}\left(\hat{\mathbf{M}}\mathbf{A}\hat{\mathbf{x}}\right). \tag{C.12}$$

If all terms in vector $\hat{\mathbf{x}}$ except for the zeroth harmonic are zero, this derivative vector is zero. This occurs if the time-domain solution is constant in time. If $\frac{\partial \mathcal{S}}{\partial \omega}$ is zero, the matrix is rank-deficient and the problem cannot be solved. Thus, the structural solution needs an initial guess of the amplitudes of motion in order to provide a solution.

Appendix D

Gradient calculation for an adjoint frequency-domain rigid body motion integrator

The solver must provide the derivative with respect to the loads at the boundaries. It takes as an input the derivative with respect to the displacements of each of the boundary nodes.

D.1 Calculation of source terms

Source terms appear on the right-hand side of the adjoint equation in Eq. (4.41). They are the partial derivative of the objective function with respect to the movement amplitude. These partial derivatives are only non-zero for structural objective functions. In the present work, two such objective functions have been implemented: the pitching amplitude and the plunge damper power dissipation.

D.1.1 Pitching amplitude

Because of the use of a frequency-domain method, the pitching amplitude's derivatives are easy to obtain. Since the sine component is set to 0, assuming that the cosine component is positive the frequency-domain derivative of the amplitude is

$$\frac{\partial \left|\hat{\alpha}_1\right|}{\partial \hat{\alpha}} = \frac{\partial \hat{\alpha}_{1,c}}{\partial \hat{\alpha}^*} = \left[0 \; 1 \; 0 \; \cdots \; 0\right]^T . \tag{D.1}$$

This term does not need to be written in the time domain. For Sec. 4.5.1 the square of the pitching amplitude was used instead. This way, the sign of the cosine

component is not taken into account. The corresponding vector of derivatives is

$$\frac{\partial \left|\hat{\alpha}_1\right|^2}{\partial \hat{\alpha}} = \left[0 \; 2\hat{\alpha}_{1,c} \; 0 \; \cdots \; 0\right]^T . \tag{D.2}$$

D.1.2 Power dissipation

The power dissipation on the plunge damper is

$$J = \frac{1}{2 \cdot N_h + 1} \sum_{i=0}^{2 \cdot N_h} c_h \cdot \dot{h}^2. \tag{D.3}$$

Differentiate it with respect to the time derivative of the plunge degree of freedom in the time domain:

$$\frac{\partial J}{\partial \dot{\mathbf{h}}} = \frac{2 \cdot c_h}{2 \cdot N_h + 1} \cdot \dot{\mathbf{h}}^T. \tag{D.4}$$

This gradient can then be transformed to the frequency domain

$$\frac{\partial J}{\partial \dot{\hat{\mathbf{h}}}} = \frac{2 \cdot c_h}{2 \cdot N_h + 1} \cdot \left(\mathbf{E}^{-1} \dot{\hat{\mathbf{h}}} \mathbf{E}\right)^T . \tag{D.5}$$

The source term obtained in this manner is added to the right-hand side of the equations.

D.2 Application of shape design variables

Any shape deformation has to be applied before the displacements. This deformed shape is the one that then moves in the pitching and plunging degrees of freedom.

For the present thesis, the Hicks-Henne bumps were implemented. The generic expression of these bumps is

$$y(x) = y_0(x) + \sum_{i=0}^{N} \xi_i \cdot f_i(x) \tag{D.6}$$

$$f_i(x) = \begin{cases} 0, & x = 0 \\ \left[\sin\left(\pi \cdot x^{\frac{\log 0.5}{\log t_1}}\right)\right]^{t_2}, & x \in \,]0, 1] \end{cases} , \tag{D.7}$$

where x and y are the chordwise and perpendicular position, ξ_i is the ith bump's amplitude, t_1 is a parameter that controls where the maximum of the bump is reached and t_2 controls the width of the bump. In order to match the original definition of the bumps as well as that used by SU2, t_2 was set to 3. The shape is not modified along the chordwise axis.

D.3 Calculation of sensitivities from adjoint variables

D.3.1 Gradient with respect to design variables

Structural design variables

Recall the expression of the gradient of an objective function, J, with respect to a design variable, ξ, using the adjoint method

$$\frac{\mathrm{d}J}{\mathrm{d}\xi} = \frac{\partial J}{\partial \xi} - \hat{\lambda}^T \frac{\partial \hat{\mathbf{R}}}{\partial \xi}, \tag{D.8}$$

where $\hat{\lambda}$ are the frequency-domain adjoint variables and $\hat{\mathbf{R}}$ is the frequency-domain residual. The derivatives with respect to some structural parameters are

$$\frac{\mathrm{d}J}{\mathrm{d}k_h} = \frac{\partial J}{\partial k_h} - \hat{\lambda}_h^T \hat{\mathbf{h}} \tag{D.9}$$

$$\frac{\mathrm{d}J}{\mathrm{d}k_\alpha} = \frac{\partial J}{\partial k_\alpha} - \hat{\lambda}_\alpha^T \hat{\alpha} \tag{D.10}$$

$$\frac{\mathrm{d}J}{\mathrm{d}c_h} = \frac{\partial J}{\partial c_h} - \hat{\lambda}_h^T \mathbf{A} \hat{\mathbf{h}} \tag{D.11}$$

$$\frac{\mathrm{d}J}{\mathrm{d}c_\alpha} = \frac{\partial J}{\partial c_\alpha} - \hat{\lambda}_\alpha^T \mathbf{A} \hat{\alpha} \tag{D.12}$$

$$\frac{\mathrm{d}J}{\mathrm{d}m} = \frac{\partial J}{\partial m} - \hat{\lambda}_h^T \mathbf{A}\mathbf{A} \hat{\mathbf{h}} + (x_f - x_{CG}) \cdot \left(\frac{\partial J}{\partial S_\alpha} - \hat{\lambda}_h^T \mathbf{A}\mathbf{A} \hat{\alpha} - \hat{\lambda}_\alpha^T \mathbf{A}\mathbf{A} \hat{\mathbf{h}} \right) \tag{D.13}$$

$$\frac{\mathrm{d}J}{\mathrm{d}I_\alpha} = \frac{\partial J}{\partial I_\alpha} - \hat{\lambda}_\alpha^T \mathbf{A}\mathbf{A} \hat{\alpha}, \tag{D.14}$$

where $\hat{\mathbf{h}}$ is the vector of amplitudes of the plunge degree of freedom and $\hat{\alpha}$ is that of the pitch degree of freedom. Note that the partial derivatives of the residual with respect to the structural parameters are the degrees of freedom and their time derivatives. Since the degrees of freedom are computed in order to obtain the direct solution, the gradients simplify to a scalar product of two already-computed vectors. Therefore, they are very cheap to calculate.

Shape design variables

Assume that the objective function is only a function of the mesh and fluid variables. The fluid components of the gradients have to be rotated according to the pitching angle. The moment, on the other hand, depends on the location of the boundary

as per Eq. (C.10). Expressing the values on the time domain, the gradients with respect to the nth mesh point's coordinates are

$$\frac{\mathrm{d}J}{\mathrm{d}x_n} = \left.\frac{\mathrm{d}J}{\mathrm{d}x_n}\right|_{\mathcal{F}} \cdot \cos\alpha - \left.\frac{\mathrm{d}J}{\mathrm{d}y_n}\right|_{\mathcal{F}} \cdot \sin\alpha - \lambda_\alpha \cdot \left(F_y^n \cdot \cos\alpha - F_x^n \cdot \sin\alpha\right), \tag{D.15}$$

$$\frac{\mathrm{d}J}{\mathrm{d}y_n} = \left.\frac{\mathrm{d}J}{\mathrm{d}x_n}\right|_{\mathcal{F}} \cdot \sin\alpha + \left.\frac{\mathrm{d}J}{\mathrm{d}y_n}\right|_{\mathcal{F}} \cdot \cos\alpha - \lambda_\alpha \cdot \left(F_y^n \cdot \sin\alpha + F_x^n \cdot \cos\alpha\right), \tag{D.16}$$

where $\left.\frac{\mathrm{d}J}{\mathrm{d}x_n}\right|_{\mathcal{F}}$ and $\left.\frac{\mathrm{d}J}{\mathrm{d}y_n}\right|_{\mathcal{F}}$ are the components of the gradient provided by the fluid solver. These gradients are calculated for each time instance. The final result is the sum over the time instances. The time-domain adjoint variables are calculated according to Eq. (4.50).

D.3.2 Partial derivative of power dissipation with respect to structural parameters

The only objective function that depends directly on a structural parameter is the power dissipation on the plunge damper. Since it is directly proportional to the plunging velocity squared, the derivative is

$$\frac{\partial J}{\partial c_h} = \frac{\dot{h}^2}{2 \cdot N_h + 1}. \tag{D.17}$$

This value is calculated and added over every time instance.