Aerodynamic Shape Optimization using a Time-Spectral Approach for Limit Cycle Oscillation Prediction

by

Sicheng He

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2021

Doctoral Committee:

Professor Joaquim R. R. A. Martins, Chair
Professor Carlos E. S. Cesnik
Professor Bogdan I. Epureanu
Professor Krzysztof J. Fidkowski

Sicheng He

hschsc@umich.edu

ORCID iD: 0000-0003-1307-4909

© Sicheng He 2021

To my parents

Acknowledgments

I would like to thank Prof. Martins for his support through the years of my Ph.D. study. He has pointed me towards a fruitful direction of research. His passion for numerical methods and aircraft design is contagious. I have also benefited from the quick responses and feedbacks from him on both research and other aspects of life. I consider myself very lucky to have Prof. Martins as my advisor.

I would also like to acknowledge my committee members, Profs. Cesnik and Fidkowski from the aerospace engineering department and Prof. Epureanu from the mechanical engineering department for agreeing to review this thesis and the efforts they have made to accommodate my schedule. I took one of my first courses on structural mechanics and also the aeroelasticity course from Prof. Cesnik. In the aeroelasticity class for the first time, I learned about the phenomenon called flutter that I did not know by that time would be my Ph.D. thesis topic. After taking Prof. Fidkowski's aerodynamic class as an undergraduate, I made my mind to pursue a Ph.D. in this field. He was also kind enough to take me over several independent studies that gave me a chance to work on the discontinuous Galerkin method based code *xflow*. This makes the experience of learning our CFD code *ADflow* much smoother.

During my Ph.D. study, I also had a great opportunity to collaborate with Profs. Terlaky, and Zuluaga from Lehigh University on a side project about structural optimization. I have learned many things about linear algebra and convex optimization from them. I also want to thank them for the efforts they have put into editing my first structural optimization paper which I believe improve the paper's quality a lot. I am also thankful to Mohammad Shahabsafa, Ali Mohammad Nezhad, Weiming Lei, and Ramin Fakhimi for their help.

Through my Ph.D., I collaborated the most with Eirikur Jonsson. I learned many things from him about programming and visualization through this collaboration. Besides, whenever I had any issue with my computer or the cluster, Eirikur was always there to help. Daning Huang, Gaetan Kenway, John Hwang, and Charles (Sandy) Mader helped me to get on my research. The Friday frisbee games with them are missed a lot. I also enjoyed the time that I spent with the lab visiting students Jichao Li, Yayun Shi, Xiaolong He, and Song Chen. They were very hard working and always held a very positive attitude about life and research even during difficult times

(a.k.a your adjoint gradient does not match with the finite difference derivative). Talking with them always reminded me about my home town that I did not have a chance to visit for the last six years. I would like to thank Hang Li from the University of Tennessee for many useful talks about flutter. I thank Yinqian Liao for the many questions she has answered about visualization towards the very end of my Ph.D. study. I am grateful to Ping He, Peter Lyu, Devina Sanjaya, and Shaowu Pan for giving so many great suggestions on research and beyond. I also benefited from the interactions with Anil Yidrim, Ben Brelje, David Burdette, Gustavo Halila, John Jasa, Josh Anibal, Marco Mangano, Mohamed Bouhlel, Nick Bons, Neil Wu, Sabet Seraj, Shamsheer Chauhan, Shugo Kaneko, and Xiaosong Du. Chenyu Yi, Qian Ma, Xianjun Pei, Jifa Mei, Liren Yang, Bowen Li, and Sid Srivastava have been very supportive all through the years.

Finally, I would like to thank my parents for their support through my Ph.D. study. Without their support, I would never accomplish this goal.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF APPENDICES

APPENDIX

LIST OF ACRONYMS

AD algorithmic differentiation

AGARD Advisory Group for Aerospace Research and Development

ALE arbitrary Lagrangian Eulerian

ANK approximate Newton–Krylov

AOA angle-of-attack

BWB blended wing body

CFD computational fluid dynamics

CFL Courant—Friedrichs—Lewy

CNK coupled Newton–Krylov

CSD computational structural dynamics

CS complex-step

DADI diagonalized alternating direction implicit iterations

DFT discrete Fourier transform

DOF degree-of-freedom

FD finite difference

FFD free-form deformation

FEM finite element method

FFT fast Fourier transform

FGMRES flexible generalized minimal residual

FAD forward algorithmic differentiation

GCL geometric conservation law

GMRES generalized minimal residual

GS Gauss–Seidel

LCO limit cycle oscillation

LHS left hand side

MDO multidisciplinary design optimization

MC Monte Carlo

NLFD nonlinear frequency domain

PDE partial differential equation

RAD reverse algorithmic differentiation

RANS Reynolds-averaged Navier–Stokes

RHS right hand side

SA Spalart–Allmaras

SPS saddle point system

TSD transonic small disturbance

uCRM undeflected Common Research Model

URANS unsteady Reynolds-averaged Navier–Stokes

XDSM extended design structure matrix

LIST OF SYMBOLS

Latin Symbols

$\mathbf{A}^n$	a vector containing all values of the area in different time-instances of a cell
A_i	area of a cell at the i^{th} time-instance
$\boldsymbol{\mathcal{A}}_{\text{TS}}$	time-spectral CFD residual
a	nondimensional location of airfoil elastic axis
$\mathbf{B}_1$	a matrix dependent on V_f, $\mathbf{f}^n$, $\mathbf{M}^n$, $d\mathbf{D}_{\mathbf{Q}}/d\omega$, and $\mathbf{u}^n$
$\mathbf{B}_2$	a matrix dependent on $\partial\lvert\alpha_{\text{1st mode}}\rvert/\partial\mathbf{u}^n$ and $\partial\mathbf{u}^n\partial\phi/\partial\mathbf{u}^n$
b	semi chord length
$C_{l,i}$	lift coefficient for i^{th} time-instance
$C_{m,i}$	moment coefficient for i^{th} time-instance
C_p	coefficient of pressure
$\mathbf{D}$	spectral differentiation operator
$\mathbf{D}_{\mathbf{Q}}$	second-order time differentiation operator for a two degree-of-freedom system
$\mathbf{D}_{t,t}$	second order spectral differentiation operator
$\mathbb{E}$	set of all edges of a cell
e_{rel}	finite-difference parameter
$\mathbf{f}$	aerodynamic load
$\mathbf{f}_{\boldsymbol{\mathcal{A}}}^n$	aerodynamic load on aerodynamic surface mesh nodes for all time-instances
$\bar{\mathbf{f}}_i$	i^{th} time-instance aerodynamic load, equal to $\left[-C_{l,i}, 2C_{m,i}\right]^\top$
$\bar{\mathbf{f}}_{r,i}$	dimensionless aerodynamic for the i^{th} structural mode
$\mathbf{f}_r$	generalized external loads
$\bar{\mathbf{f}}$	dimensionless aerodynamic load
$\bar{\mathbf{f}}^n$	dimensionless aerodynamic load for all time-instances
$\boldsymbol{\mathcal{G}}$	function that transfers structural displacements to aerodynamic surface mesh
$\boldsymbol{\mathcal{G}}'$	functions that transfers aerodynamic surface loads to structural nodes
h	plunging coordinate
$\mathbf{I}$	function of interest
I_α	airfoil moment of inertia
$\mathbf{J}$	time-spectral aeroelastic equation Jacobian
$\mathbf{K}$	stiffness matrix
$\mathbf{K}^n$	stiffness matrix for all time-instances

$\mathbf{K}_r$	reduced stiffness matrix
$\mathbf{K}_r^n$	reduced stiffness matrix for all time-instances
$\mathbf{k}$	a unit vector perpendicular to the airfoil plane
$\mathbf{M}$	mass matrix
$\mathbf{M}^n$	mass matrix for all time-instances
$\mathbf{M}_r$	reduced mass matrix
$\mathbf{M}_r^n$	reduced mass matrix for all time-instances
M	Mach number
m	airfoil mass
m_0	wing mass
N_{CFD}	number of CFD state variables
N_{CSD}	number of CSD state variables
$N_{\mathbf{x}}$	number of the design variables
n	number of time-instances
$\mathbf{P}$	preconditioner for coupled system
$\mathbf{P}_{\text{CFD}}$	block preconditioner for CFD
$\mathbf{P}_{\text{motion, CSD}}$	block preconditioner for motion and CSD component
p_∞	mainstream static pressure
$\mathbf{Q}$	permutation matrix
$\mathbf{q}$	coupled time-spectral aeroelastic equation state variables
q_∞	mainstream dynamic pressure
$q^{(k)}$	state variable for kth Newton iteration
$\mathbf{R}$	spatial CFD residual
$\boldsymbol{\mathcal{R}}$	coupled time-spectral aeroelastic residual
$\boldsymbol{\mathcal{R}}_m$	prescribed motion magnitude residual from $\boldsymbol{\mathcal{R}}$
$\boldsymbol{\mathcal{R}}_p$	prescribed motion phase residual from $\boldsymbol{\mathcal{R}}$
$\mathbf{r}$	linear equation residual for Newton updates
r_α	radius of gyration
S_{ref}	reference area for the wing
$\boldsymbol{\mathcal{S}}_{\text{TS}}$	time-spectral CSD residual
S_α	static unbalance, equal to mbx_α
T	minimum time period, equal to $2\pi/\omega$
t	time
U_f	freestream speed
U_∞	mainstream flow speed
$\mathbf{u}$	CSD state variable
$\mathbf{u}_i$	state variable for i^{th} time-instance
$\mathbf{u}^n$	CFD state variable for all time-instances
u_{min}	one of finite-difference parameters for matrix-free matrix-vector product
V_0	wing volume
$\mathbf{V}_i$	nodal velocity of the i^{th} node

V_f	flutter (LCO) velocity index, equal to $U_f/b\omega_\alpha\sqrt{\mu}$
$V_{f,0}$	initial state flutter (LCO) velocity index
$\mathbf{v}_{i,j}$	edge vector from point i to point j
V_x, V_y, V_z	flow velocities relative to mesh in the x, y, z directions, respectively
v_x, v_y, v_z	flow velocities in the x, y, z directions, respectively
$v_{x,g}, v_{y,g}, v_{z,g}$	mesh velocities in the x, y, z directions, respectively
$\boldsymbol{\mathcal{W}}$	aerodynamic surface mesh to volume mesh transfer module
$\mathbf{X}_{S,0}$	surface mesh for the jig shape
$\mathbf{X}_S^n$	surface mesh for all time-instances
$\mathbf{X}_V^n$	volume mesh for all time-instances
$\mathbf{x}_g^n$	mesh x coordinates for all time-instances
x_g	mesh x coordinate
x_α	static unbalance
y^+	non-dimensional distance of the first mesh layer from a wall

Greek Symbols

α	pitching coordinate
α_m	mean angle of attack
α_1	pitching motion first harmonic mode
$\Delta\mathbf{q}$	increment unit Newton step without preconditioner
$\Delta\mathbf{q}_{V_f,\omega,\text{CSD}}$	part of $\Delta\mathbf{q}$ corresponding to V_f, ω and CSD state variables
$\Delta\mathbf{q}_1$	first two rows of $mbq_{V_f,\omega,\text{CSD}}$, corresponding to V_f and ω
$\Delta\mathbf{q}_2$	3rd row and beyond of $mbq_{V_f,\omega,\text{CSD}}$, corresponding to the CSD state variables
$\Delta\mathbf{y}$	increment unit Newton step with preconditioner
$\Delta\mathbf{y}_{\text{CFD}}$	CFD state variable part of Δy
$\Delta\mathbf{y}_{V_f,\omega,\text{CSD}}$	V_f, ω and CSD state variable part of Δy
$\epsilon_{0,j}$	prescribed motion magnitude for the j^{th} mode
ζ	CFD state variables
ζ^n	CFD state variables for all time-instances
$\boldsymbol{\eta}_j^i$	structural displacement of j^{th} mode
$\bar{\boldsymbol{\eta}}_j^i$	dimensionless structural displacement of j^{th} mode
θ	Newton step size parameter
$\theta_{1\text{stharmonic,j}}$	the phase angle of 1st harmonic component of the j^{th} mode
$\boldsymbol{\Lambda}$	matrix composed of eigenvalues of the generalized eigenvalue problem
μ	airfoil (wing) mass ratio
$\boldsymbol{\Phi}$	the matrix made up of $\boldsymbol{\phi}_j$
$\breve{\boldsymbol{\Phi}}$	polynomial fitted mode shapes
$\hat{\boldsymbol{\Phi}}$	reduced set of the mode shapes
$\tilde{\boldsymbol{\Phi}}$	truncated set of the mode shapes
ϕ	pitching motion for first harmonic mode phase

ϕ_0	prescribed pitching motion for first harmonic mode phase
$\boldsymbol{\phi}_j$	the j^{th} structural mode shape
$\boldsymbol{\psi}$	the adjoint vector
$\boldsymbol{\Omega}$	the matrix made up of ω
$\boldsymbol{\Omega}^n$	$\boldsymbol{\Omega}$ for all time-instances
ω	flutter (LCO) frequency, equal to $2\pi/T$
ω_j	natural frequency of the j^{th} mode
ω_h	plunging natural frequency
ω_α	scaling frequency (pitching natural frequency for the airfoil and the 2nd structural mode natural frequency for the wing)
ω_0	initial state flutter (LCO) frequency

ABSTRACT

In aircraft design, limit cycle oscillation (LCO) is an important phenomenon that we need to consider. Future aircraft are likely to have more flexible wings making them more susceptible to LCO. To avoid this tendency, we can conduct an multidisciplinary design optimization (MDO) to maximize the LCO onset speed by changing the aerodynamic shape of a wing.

One challenge is that we need to simulate LCO efficiently using a high-fidelity computational fluid dynamics (CFD) model. Previous harmonic-balance-based LCO prediction methods either have low linear convergence rates or require expensive Newton steps to achieve quadratic convergence. To address this, we propose a preconditioned, Jacobian-free, coupled Newton–Krylov (CNK) method for the time-spectral aeroelastic equations. By solving the coupled system directly, the method reduces the computational cost of each Newton step, making quadratic convergence affordable. We demonstrate the capability of the CNK solver by verifying the results against a time-accurate solver and by comparing them to other harmonic-balance-based results reported in the literature. We observe that the proposed method is more efficient than the time-accurate method in LCO response simulations.

Another challenge is that we need to compute the LCO speed derivative to a large number of design variables. We base our work on previous research in the literature, which uses a segregated adjoint formulation. We use the coupled adjoint approach, which is a monolithic way to compute the gradient. The coupled adjoint is cheaper to compute compared to the segregated adjoint. We verify the adjoint sensitivity computation with the finite difference method, where we achieve 10^{-6} accuracy for most design variables for the wing test case. We conduct an aerodynamic shape optimization of a wing, and the LCO speed increases by 118%.

Finally, to extend the adjoint method to an aerostructural optimization problem, we propose two formulations based on reverse algorithmic differentiation (RAD) to reduce the computational cost to one single computation.

To conclude, we developed computational methods to make aerodynamic shape optimization for LCO suppression practical for wing cases, and the RAD formulae for the mode shapes and natural frequencies are likely to be useful for future aerostructural optimization.

CHAPTER 1

Introduction

In this chapter, we present the motivation for our research in Section 1.1. The context for the different parts of this thesis is explained in Section 1.2. Finally, we show our contributions and present the organization of the thesis in Section 1.3.

1.1 Motivation

The wings of next-generation aircraft are trending towards the higher aspect ratio, more flexible designs, making them more prone to flutter. Since flutter is a certification-critical phenomenon, it is important to predict this aeroelastic phenomenon accurately as early as possible in the design process. Ignoring flutter may lead to overly-flexible wings that cause problems when certifying the aircraft. Flutter issues are often identified only at the final design or flight testing stages, at which point design changes are extremely costly. Therefore, accurate flutter prediction methods reduce the risk and can lead to significant cost savings.

LCO is another important phenomenon encountered in wing design, which can be subcritical or supercritical [57]. LCO is a more general concept than flutter where the latter can be taken as an LCO with an infinitesimal motion magnitude. Thus, besides a more detailed elaboration of flutter and LCO in Section 1.2, in the rest of the thesis, we will treat flutter as a special case of LCO and avoid mentioning flutter if possible. Exceptions includes *flutter boundary*, and *flutter point*

(speed).

LCO only appears in nonlinear dynamic systems, where the nonlinearity could come from the aerodynamic model. Both shock wave motions and flow separations could result in a nonlinear relation between the aerodynamic force with respect to structural displacement. It could also come from the structural model. Possible structural model nonlinearity includes a control surface freeplay, geometric nonlinearity, and other factors [24]. In this thesis, we only consider the aerodynamic model nonlinearity. To resolve the shock wave motion and the flow separation, we employ time-spectral methods with Euler and Reynolds-averaged Navier–Stokes (RANS) equations.

Time-spectral method is an efficient tool used for simulating periodic flows [39, 44, 102]. The time-spectral method was developed for turbomachinery applications and later used to model external flows. By keeping several uniformly distributed "snapshots" in time, the time-spectral method reconstructs the periodic flow field using Fourier transform. Earlier studies have shown that the time-spectral method is one order more efficient in simulating periodic flow compared with the general-purpose time-accurate method, such as unsteady Reynolds-averaged Navier–Stokes (URANS) [101]. This is because the latter has to resolve a transient response before the final periodic flow field is captured, and the former method directly models the final periodic flow field. Another reason is that the time-spectral method requires much fewer time instances per period compared with the time-accurate method. The approach is an ideal time discretization scheme for LCO modeling because LCO is periodic in time. For more discussion on spectral methods, we refer the readers to the book by Boyd [16].

In LCO simulation, the frequency and flow speed to trigger such responses are not known and need to be figured out. This is different from a compressor turbine [44], a helicopter blade [22] or a rotor [47] where the frequency and boundary conditions are known *a priori* . To address the issue, Thomas et al. [137] proposed a set of equations to model LCO. The set of equations include aerodynamic equations, structural dynamic equations, and prescribed motion magnitude and phase equations. We denote the set of equations as the time-spectral aeroelastic equation to distinguish it

from the steady-state aerostrucural equation [62]. By solving this time-spectral aeroelastic equation, the LCO speed and motion frequency are found. They proposed using the Newton method to solve this set of equations. The Newton solver interfaced directly with the computational structural dynamics (CSD) solver, and the aerodynamic load was evaluated by solving CFD equations with the current structural displacement. Because of this segregated formulation, the method required multiple flow solutions for each Newton step. A total of $\mathcal{O}(N_{\text{CSD}} \times n \times N_{\text{Newton}})$ CFD evaluations for each solution is required, where N_{CSD} is CSD degree-of-freedom (DOF)s, n is number of time instances, and N_{Newton} is number of Newton steps. To address the high computational cost issue, we propose an alternative CNK method which only requires the CFD residual to be driven to zero twice: once at a warm start stage and the other during the time-spectral aeroelastic equations solution.

When designing an aircraft, the aircraft shall never demonstrate any LCO behavior within its flight envelop. That is to say, we want to make sure that the LCO speed be high enough to avoid any catastrophe. When high fidelity tools are used for design optimization, the only viable optimization methods are gradient-based optimization approaches due to fewer function evaluations required compared with other gradient-free methods such as genetic algorithms. For the derivative computation, there are several methods available: finite difference (FD), complex-step (CS), direct and adjoint methods [95]. The former three methods are preferred for cases with more functions of interest than design variables. The adjoint method, however, is preferred for the opposite case where there are more design variables than functions of interest. For aerodynamic shape optimization, there are usually a handful of functions of interest and hundreds or even thousands of design variables. Thus, an adjoint method shall be used. A segregated adjoint formulation was proposed by Thomas and Dowell [135]. Similar to the solution method proposed by Thomas et al. [137], the structural equations are directly resolved, and the aerodynamic equations are enforced implicitly. Using this formulation, the derivative of the generalized aerodynamic load at each time-instance for each structural DOF needed to be computed resulting in $N_{\text{CSD}} \times n$ aerodynamic adjoint to be

solved for each coupled adjoint computation (The direct method is equally applicable here since the Jacobian here is a square matrix). To address this high computational cost, we develop a coupled adjoint formulation where the adjoint of the coupled system is solved together once. The linear system is only slightly bigger than the original CFD adjoint equations by $\mathcal{O}(N_{\text{CSD}} \times n)$ making the resulting coupled adjoint much more computationally efficient.

In this thesis, we consider optimizing the LCO speed index by changing the aerodynamic shape. Ultimately, we want to take the structural design variables into the picture for an aerostructural optimization. To achieve that goal, we could at first construct a finite element model based on input structural design variables. And then, we conduct a modal analysis to construct the mode shapes of the structure. After this, we can conduct an LCO analysis based on the methods proposed in this thesis. For the derivative computation, the same adjoint method proposed in the thesis could be used. However, different from the aerodynamic shape optimization, here we must compute the derivative of the structural modes with respect to the design variables. In the literature, there are forward formulations that are able to solve the problem in $\mathcal{O}(r \times n_{\mathbf{x}})$ iterations, where r is the number of modes and $n_{\mathbf{x}}$ is the number of structural variables. And there is also the adjoint approach that can be used to compute the gradient accurately by solving r adjoint equations. We propose two methods to compute the derivatives. The first method approximate the mode shape derivative by conducting $\mathcal{O}(r)$ matrix-vector product computations. The second method adds correctional terms to the first formulation to make it more accurate, but this also adds a computational cost of solving additional r elastic equations. We show that these equations can be solved at a low cost.

1.2 Background

1.2.1 Flutter and LCO

Flutter is a dynamic aeroelastic instability that causes divergent harmonic vibrations. The flutter speed corresponds to the minimum airspeed where the structure enters into periodic oscillation [13].

Flutter in the subsonic regime can be modeled with linear aerodynamics using a doublet-lattice method (DLM), for example. Most commercial airliners operate in the transonic regime, which is more challenging to model because it involves highly nonlinear aerodynamics. In particular, there is a significant reduction in the flutter speed in the transonic regime, called the *transonic dip*, as shown in Fig. 1.1. The transonic dip is mainly due to compressibility effects. Low fidelity models tend to underestimate this dip, leading to overprediction of the flutter speed. In this work, we model the flow with the Euler equations and RANS equations so that we can capture the transonic dip.

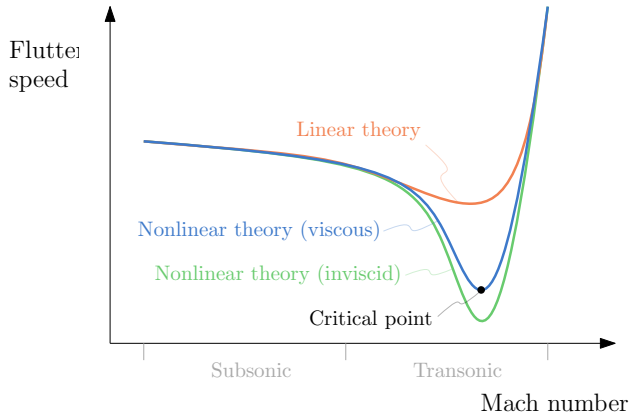


Figure 1.1: The transonic dip is best captured using a nonlinear viscous aerodynamic model [57].

LCO is a periodic motion of the structure due to interactions between the fluid and the structure. LCO can be supercritical or subcritical. These two types of responses are visualized in Fig. 1.2, where the dashed lines represent unstable responses, and the solid lines represent stable ones.

The arrows indicate the paths taken by subcritical and supercritical LCO responses with varying airspeed.

The supercritical LCO is a smooth and benign response, for which the structure stays steady below the flutter speed. As the airspeed increases above the flutter point, the structure enters into a periodic motion. If the airspeed decreases, the motion reduces in magnitude and returns to a stable state, following the same trajectory as when the speed increases, but in the reverse direction.

The subcritical LCO features a sudden jump to a finite magnitude LCO as the airspeed increases. This class of LCO exhibits a hysteresis when varying the airspeed. Once the LCO is encountered, it continues to a much lower airspeed than the value that originally triggers the onset of the LCO, before jumping back to a steady and non-oscillatory state. The red curves in Fig. 1.2 show the path of this type of LCO. The sharp jump in the oscillation magnitude of subcritical LCO may be destructive, and therefore we want to make sure that designs do not exhibit this behavior. This requires a model that can capture both the subcritical and supercritical response branches. Capturing the unstable branch from subcritical LCO can be particularly challenging.

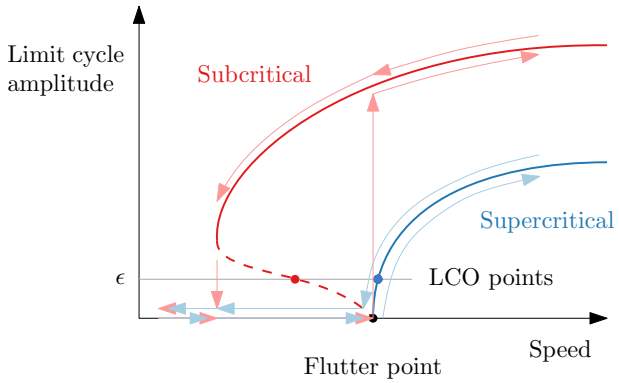


Figure 1.2: LCO response curves and flutter point.

1.2.2 LCO solution methods

In the research community, the standard method for predicting a wing's flutter boundary is to analyze the wing with a time-accurate CFD solver coupled to a CSD solver, as proposed by Liu et al. [81]. However, these methods incur a high computational cost, since hundreds or thousands of time steps are required to simulate the LCO response. Furthermore, to accurately locate the flutter point, several flow conditions need to be tested. Opgenoord et al. [108] attempted to reduce the cost of this type of analysis by constructing a low-fidelity aerodynamic model based on time-accurate CFD data. The low-fidelity model has around 5% error compared with CFD results for airfoil or wing test cases in the transonic regime. Opgenoord et al. [108] considered one airfoil shape factor variable–the thickness ratio for the low-fidelity model. To make the model useful for designers, a much bigger design space needs to be sampled. Ghadami et al. [34], Riso et al. [115] developed a data-driven method for flutter boundary prediction. At first, several time-accurate solutions were sampled under different flow speeds in the pre-flutter domain. The corresponding damping ratios were collected. Then, the damping ratio was approximated as a function of flow speeds. Finally, the flutter speed that corresponded with a zero damping ratio was found. Several pre-flutter sample points should be enough to capture the flutter point. However, the pre-flutter points seemed to be very close to the flutter point [115]. For a new configuration, it may be difficult to decide where to put the sampling points.

Another shortcoming of the time-accurate methods is that they cannot predict the unstable branch from the subcritical response. This is because any small perturbation steers it towards stable branches with either a higher limit cycle amplitude or a fixed point.

As in many periodic problems, much of the computational time consumed in the time-accurate simulation is spent resolving the decay of the initial transients in the unsteady problem [102]. Fortunately, in problems where the periodic steady-state solution is of primary interest, time-periodic simulation methods, such as the nonlinear frequency domain (NLFD) method [102], the harmonic-balance method [44], and the time-spectral method [39, 44] can all be used to accelerate the so-

lution process. Recently, the scalability of the time-spectral method was improved by Ramezanian et al. [114]. The computational time of the proposed method was demonstrated to scale with $n \log_2 n$ for even numbers of time instances and with $2n \log_3 n$ for odd numbers. This was achieved by using the fast Fourier transform (FFT) to replace the original discrete Fourier transform (DFT).

In the present work, we use the original time-spectral method. The idea is to capture a time-periodic variable with snapshots from different time instances using DFT, which allows the time derivatives to be evaluated using spectral differentiation. Formulating the problem completely in the time domain makes it easier to implement in an existing steady solver.

Work has also been done to extend these spectral methods to solve aeroelastic problems. For example, Kachra and Nadarajah [58] extended the NLFD method to simulate an airfoil aeroelastic response in a loosely coupled manner. Tardif and Nadarajah [129] and Benoit and Nadarajah [9] proposed the geometric conservation law (GCL) for time-spectral CFD, where they solve structural and aerodynamic equations separately, and exchange the interface data for every few cycles. Mavriplis and Yang [100] proposed a GCL scheme in which the swept volumes are computed by discretizing the trajectories of the faces between the two time-levels into a large number of discrete steps and computing the volume swept between each step using a two-point integration rule in time. Choi and Datta [22] performed a time-spectral aeroelastic simulation of a three-dimensional helicopter rotor. Mundis and Mavriplis [104] decomposed the flow into a periodic and a polynomial motion in time. With the polynomial components, this method captured transient aeroelastic responses accurately.

There has also been significant effort put into the prediction of LCO using spectral methods. Thomas et al. [137] proposed a set of equations to capture the LCO. The solution of the equations satisfied aerodynamics, structural dynamics, prescribed motion magnitude, and prescribed motion phase constraints. They proposed using the Newton method to solve this set of equations. The Newton solver interfaced directly with the CSD solver and the aerodynamic load was evaluated by solving CFD equations with the current structural displacement. Later, they presented a detailed

parametric study of an airfoil case based on this method [65], studied wing LCO [136], proposed an aeroelastic adjoint method [135], and presented a numerically stabilized solution method [132]. Because a segregated formulation was used, the method required multiple flow solutions for each Newton step. The method needed $N_{\text{CSD}} \times n \times N_{\text{Newton}}$ CFD evaluations for each solution, where N_{CSD} is CSD DOF, n is number of time instances, and N_{Newton} is number of Newton steps.

In the present work, we reduce the computational cost of each Newton step by letting the Newton solver interface with CSD and CFD solvers simultaneously. Thus, we update both the structural displacements and the CFD solution simultaneously, allowing the Newton iterations to be completed with residual evaluations rather than full solutions.

Other spectral methods developed to analyze LCO are categorized based on whether the prescribed motion magnitude and phase constraints are considered. Among the methods that consider the prescribed motion, Thomas and Dowell [131] proposed a fixed-point iteration approach. With the proposed method, the CFD equation is no longer treated as a segregated module, and the method's computational cost is no longer proportional to the number of structural DOF. To address the issue with the expensive Newton step, a nonlinear Gauss–Seidel (GS) type method named as "one-shot" method was proposed [70–73]. The name "one-shot" indicates that the CFD residual is reduced to zero only at the end. Gong and Zhang [38] solved the same set of equations for the flutter point prediction. However, for LCO, Gong and Zhang [38] proposed a different treatment, where the LCO speed was prescribed, and the motion was solved. All the methods mentioned above have a linear convergence rate. In contrast, our proposed method exhibits a quadratic convergence rate, making it more efficient compared with other approaches.

For the methods that do *not* consider the prescribed motion, Prasad et al. [113] presented a procedure to update LCO speed and frequency. Yao and Marques [146] proposed a procedure that adapted the LCO frequency but not LCO speed based on earlier work by Ekici and Hall [29]. Tardif and Nadarajah [129] proposed a LCO onset condition that within the M consecutive aeroelastic iterations, the LCO magnitude should not change. They constrained lift coefficient magnitude and

phase, and solved for LCO speed and frequency.

In this work, we propose to use a CNK method to solve the time-spectral aeroelastic equations that include the prescribed motion constraint. In theory, it has a quadratic convergence rate, which we have observed in practice. By resolving all the state variables at once, we reduce the number of CFD solutions. During the nonlinear solution, each Newton step is solved with the Jacobian-free Krylov solver. Thus, the Jacobian of the time-spectral aeroelastic equations is never formed explicitly, which used much less memory. We propose direct, Schur, and saddle point system (SPS) preconditioners for the Krylov solver. This approach has been used to predict wing aeroelastic response with an inviscid model as well [50].

1.2.3 LCO derivative computation methods

For partial differential equation (PDE) constrained optimization problems, the function evaluation can be very expensive. The only promising method to conduct such optimization is the gradient-based optimization method as demonstrated by Lyu et al. [88] because the gradient-based method requires much fewer function evaluations compared with the gradient-free method. The adjoint method was developed to evaluate function derivative with a large number of design variables [54, 59, 99].

In the perspective of dynamical systems theory, the behavior of a nonlinear dynamic system includes fixed points, LCO, and chaos. The derivative of a steady fluid dynamic or fluid-structure interaction problems are mature these days with the triumph of adjoint-based approach [54, 99]. Most solutions encountered for the steady problems are fixed points. Recent efforts include adding model fidelities [42, 111, 121] and including new constraints [61, 75]. The adjoint method for LCO is less developed with some previous work on LCO [135] and buffet [133]. For the chaotic system, the least square shadowing adjoint by Shimizu and Fidkowski [122], Wang et al. [143] was developed to deal with numerical stability issues.

The flutter point or LCO derivative computation method can be classified into two categories

based on model fidelity: (1). Non-CFD based approaches, and (2). CFD-based methods. For category (1), Bartels and Stanford [5] solved a structural optimization problem with the flutter constraint computed by eigenvalue analysis. Beran et al. [10], Kennedy et al. [60] developed adjoint equations for the flutter constraint that is formulated using Hopf bifurcation. Jonsson et al. [55, 56] proposed an adjoint method with an enhanced pk method that can track the change of flutter modes. Lupp and Cesnik [84], Lupp et al. [85] optimized the fuel burn of a blended wing body (BWB) [86] configuration and an undeflected Common Research Model (uCRM) 13.5 configuration [17] with geometric nonlinear flutter constraint, respectively. Though computationally efficient, most of the methods in this category are not able to predict important nonlinear flow phenomena such as shock waves motion as a function of the structural displacement.

There is a handful of CFD-based methods for flutter analysis with sensitivities in the literature. Stanford et al. [125] proposed a pk method with nonlinear Euler solver and time-linearized transonic small disturbance (TSD) analysis. Chen et al. [20] proposed a method using an Euler CFD solver and a boundary layer code. The derivative is evaluated by the CS method. The computational cost of this method scales with the number of design variables that makes it impractical for problems with a large number of design variables. Zhang et al. [150, 151] formulated a time-accurate adjoint for flutter analysis with an Euler solver. In their formulation, the flutter speed is not a variable to be solved for under given design variables. Instead, the flutter speed is a fixed parameter to be provided by the user and the optimizer's role is to find a configuration to make sure under the given speed the wing has a neutral response. This formulation is not flexible enough to provide a flutter constraint for aircraft design. Kiviaho et al. [66] developed a time-accurate adjoint by the matrix-pencil method. Leveraging the efficient harmonic balance solver, Thomas and Dowell [130], Thomas et al. [134], Thomas and Dowell [135] proposed a novel harmonic balance adjoint for the LCO. The proposed adjoint method is much cheaper in memory use compared with existing unsteady adjoint formulations because only a handful of time instances needed to be stored. They proposed to solve the adjoint equation in a segregated manner. Using the segregated

approach, the CSD equations are directly handled, and the CFD load are treated as a function of the structural displacements. To compute the LCO speed derivatives, the derivative of generalized CFD loads of each structural DOF for each time instance with respect to the design variables need to be computed. By applying the adjoint method for the CFD component to compute the aforementioned derivative, there will be $\mathcal{O}(N_{\text{CSD}} \times n)$ CFD adjoint equations waiting to be solved where N_{CSD} is the structural DOF, and n is the number of time instances. Even though the computational cost is no longer dependent on the number of design variables, this cost still seems to be too much.

One contribution of the thesis is to deal with the challenge related to the gradient evaluation. We propose to use a monolithic Krylov subspace method to solve the aeroelastic adjoint equation originally proposed by Thomas et al. [134] where a segregated approach is used to solve the adjoint equation. Compared to the segregated approach, our monolithic approach requires to solve a single slightly bigger linear system compared with solving $\mathcal{O}(N_{\text{CSD}})$ linear systems. Since the method is categorized as an adjoint method, the derivative computation time is independent of the number of design variables. The Krylov subspace method is applied for the equation solution that requires the evaluation of transpose Jacobian and vector products. To further reduce the memory cost, instead of forming the matrix explicitly, we apply a matrix-free RAD approach that was based on the previous work by Mader et al. [91].

1.2.4 Natural frequency and mode shape sensitivity computation methods

Eigenvalue and eigenvectors are essential metrics when characterizing dynamic system behavior and stability. They are widely used in engineering applications, such as structural dynamics with mode superposition [6], aeroelastic simulation [25, 57, 126, 127], laminar-turbulence transition prediction [26, 118, 120, 121], buffet-onset prediction [139, 145], reacting flow instability analysis [30], turbine blade mistuning prediction [8, 79, 89, 128], and dynamic system identification [77, 78]. In free-vibration problems, the eigenvalues represent the *natural frequencies*, and the eigenvectors represent the corresponding *mode shapes*. The eigenvalues and eigenvectors are

found by solving the generalized eigenvalue problem $\mathbf{M}\boldsymbol{\phi} = \mathbf{K}\boldsymbol{\phi}\lambda$ where $\mathbf{M}, \mathbf{K}$ are mass and stiffness matrices, respectively, and $\lambda, \boldsymbol{\phi}$ is an eigenvalue-eigenvector pair. Both matrices are real and symmetric. When performing gradient-based design optimization, the derivatives of the eigenvalues and the eigenvectors with respect to the design variables need to be computed efficiently and accurately.

There have been significant efforts in computing the eigenvalue and eigenvector derivatives. Fox and Kapoor [32] proposed a modal formulation, by which an eigenvector derivative vector is decomposed into a linear combination of eigenvectors. One drawback of the formulation is that it requires the knowledge of all the eigenvectors to compute one eigenvector derivative vector to machine precision. Liu et al. [82] discussed the truncation error of the formulations proposed by Fox and Kapoor [32] when only a subset of eigenvectors is used. Lim et al. [76] and Wang [141] proposed an improved modal formulation based on the formulations proposed by Fox and Kapoor [32] with a reduced basis. The improved modal formulation uses the calculated reduced eigenvectors to approximate the truncated terms. They showed that the improved modal formulation is more accurate than that computed by Fox and Kapoor [32]. Bernard and Bronowicki [11] and Zhang and Wei [149] extended the modal method to cases with repeated eigenvalues. Beck et al. [8] used the modal method to compute the derivative of cyclically symmetric bladed disks with repeated eigenvalues. Later, Nelson [107] proposed an alternative normalization condition that does not require knowledge of all the eigenvectors. Murthy and Haftka [105] reviewed the field and proposed an improved method of Nelson [107] that removes the dependency on the left eigenvectors. Friswell and Adhikari [33] extended Nelson's method to include the complex eigenvectors. Rudisill and Chu [116] proposed iterative and algebraic methods to compute derivatives for the eigenvalues and eigenvectors. This method requires the computation of the left eigenvectors in addition to the commonly used right eigenvectors. Lin et al. [80] provides a recent review of progress in this area.

However, most of these efforts propose formulations that do not scale well with the number of design variables. Because high-fidelity wing design optimization requires many design vari-

ables [55, 57], we focus on developing formulations that scale well with the number of design variables.

Algorithmic differentiation (AD) is a powerful tool for differentiating computer programs. Various tools have been developed that generate differentiated codes by transforming the source code line-by-line [45]. Although transforming highly optimized linear algebra libraries (e.g., LAPACK) is possible, it is tedious and requires significant implementation effort. Its success depends directly on the transformation tool used and on the source code programming paradigm. Furthermore, the transformed code's performance may be sub-optimal compared to the original routine both in terms of speed and memory usage.

Dwyer and Macphail [27] and Giles [35] showed that fundamental matrix operations such as matrix products, inversion, and eigenvalue and eigenvector computation, can be conveniently differentiated using analytic formulas suitable for AD. This is advantageous because the derivatives can then be computed using the optimized libraries, without having to differentiate the underlying library source code.

In this work, we derive analytic AD expressions for computing the derivatives of eigenvalues and eigenvectors of a real and symmetric generalized eigenvalue problem. Previous research on the eigenvalue and the eigenvector AD has largely focused on direct formulations known as forward algorithmic differentiation (FAD). FAD computes the derivatives by applying the chain rule in a forward sequence of operations propagating from the inputs (design variables) to the outputs (eigenvalues and eigenvectors in this case). The computational cost of FAD is proportional to the number of design variables.

Another approach for computing derivatives is the adjoint method, whose cost is independent of the number of design variables [95]. The adjoint analogue in AD is RAD, which computes the derivatives by applying the chain rule backward, starting with the outputs and ending with the inputs. Like the adjoint method, the computational cost of RAD is independent of the number of design variables but is proportional to the number of outputs. This is beneficial for the problems

with many design variables as inputs and few functions of interest as outputs.

One example of such a problem is multidisciplinary design optimization (MDO) problems that may have hundreds to thousands of design variables [17, 48]. Another example is the artificial neural network error backpropagation [7, 12, 15], which may involve millions of inputs and a single output (the loss function).

In many situations, we are only interested in a handful of eigenvalues and eigenvectors [50, 56, 74, 115, 124, 136].

Analytic eigenvalue differentiation methods suitable for RAD have been reported in the literature. Recently, Jonsson et al. [55] proposed a method based on reverse Lanczos iteration to efficiently compute derivatives using RAD. Giles [35] presented a collection of analytic matrix derivative results suitable for FAD and RAD. Specifically, Giles [35] presented a RAD formulation for a standard eigenvalue eigenvector problem.

In this work, we present three RAD formulations of the generalized eigenvalue problem for eigenvalue and eigenvector derivative computations. One of the formulations was originally proposed by Lee [69], and we include it here for completeness. We propose a projection-based RAD formulation based on the work of Fox and Kapoor [32]. We also present the truncation error of this method and discuss special conditions under which the truncation error vanishes. To reduce the error due to a truncated set of eigenvectors, we also develop a projection-based RAD formulation with correctional terms based on methods proposed by Lim et al. [76] and Wang [141]. As previously mentioned, we focus on problems with real and symmetric matrices and distinct eigenvalues.

1.3 Thesis overview

The contributions of the research are summarized as follows.

- We develop a CNK method to solve for flutter onset and LCO modeled by time-spectral

aeroelastic equations. The method is more efficient than the segregated Newton method in the literature. The segregated method needed $N_{\text{CSD}} \times n \times N_{\text{Newton}}$ CFD evaluations for each solution. While, our method only requires the CFD residual to be driven to zero twice: once at a warm start stage and the other during the time-spectral aeroelastic equations solution.

- We develop a monolithic Krylov subspace method to solve the aeroelastic adjoint equation originally proposed by Thomas et al. [134]. Compared to the segregated approach, the monolithic approach requires to solve a single slightly bigger linear system compared with solving $\mathcal{O}(N_{\text{CSD}})$ linear systems. By applying the adjoint method, the gradient evaluation time is independent of the number of design variables. And we apply the solver to optimize LCO speed for two-dimensional and three-dimensional configurations.

- We develop two RAD based formulae for mode shapes and natural frequencies derivative computation encountered in an aerostructural optimization problem. We demonstrate that the FAD based methods have a cost of $\mathcal{O}(rN_{\mathbf{x}})$. The proposed formulae reduce that to $\mathcal{O}(r)$ computation.

The thesis is organized as follows. In Chapter 2, we introduce the individual tools used in this research. The following chapters, Chapters 3 to 5, cover the theory developed in this research. To be specific, in Chapter 3, we present the coupled Newton–Krylov method. In Chapter 4, we detail the coupled ADjoint method. In Chapter 5, we derive the mode shape and natural frequency derivatives. Then, in Chapters 6 and 7, we demonstrate the capability of the solvers using two-dimensional airfoil and three-dimensional wing cases. In Chapter 6, the LCO and flutter boundary are solved based on the method developed in Chapter 3. And in Chapter 7, we verify the adjoint-based derivative computation based on the method developed earlier in Chapter 4 with the FD method, and we perform LCO speed optimization. Finally, in Chapter 8, we present the conclusion of the thesis.

CHAPTER 2

Computational Components

In this section, we present individual components later used in Chapters 3 to 5. Most of the tools were developed earlier and the new developments are noted. Because Chapters 3 to 5 motivate the use of the tools introduced here, it may be helpful to read the later chapters, Chapters 3 and 4, before reading this current chapter. This chapter is organized as follows: In Sections 2.1 to 2.3, we discuss the prescribed equations, time-spectral CSD equations, and time-spectral CFD equations, respectively. These three sets of equations are components for the governing equations for the LCO. Then, in Section 2.4, we cover the information transfer between the structural and the aerodynamic components. Finally, in Section 2.5, we cover the concept of AD.

2.1 Prescribed motion equations

In this section, we present derivation of the two equations constraining the magnitude and phase of the natural mode similar with [49]. States from different time-instances are defined as $\bar{\eta}_j^1, \bar{\eta}_j^2, \ldots, \bar{\eta}_j^n$ for the j^{th} mode. Defining $\omega = 2\pi/T$, where T is the time period, a harmonic motion for the j^{th} mode can be described by the following expression,

$$\bar{\eta}_{,j}(t) \approx c_{0,j} + c_{1,j}e^{i\omega t} + c_{2,j}e^{i2\omega t} + \ldots + c_{-2,j}e^{-i2\omega t} + c_{-1,j}e^{-i\omega t}, \tag{2.1}$$

where $\bar{\eta}_{,j}(t)$ denotes the j^{th} structural mode coefficient with its all resolved frequency components, and $c_{i,j}$ are coefficients for different frequency components. $\bar{\eta}_{,j}(t)$ is related with $\bar{\eta}_j^1, \bar{\eta}_j^2, \ldots, \bar{\eta}_j^n$ through taking time snapshots, i.e., $\bar{\eta}_j^i = \bar{\eta}_{,j}\left(((i-1)T)/n\right)$. $c_{i,j}$ are related with the snapshots $\bar{\eta}_j^1, \bar{\eta}_j^2, \ldots, \bar{\eta}_j^n$ through DFT,

$$[c_{0,j}, c_{1,j}, c_{2,j}, ..., c_{-2,j}, c_{-1,j}] = \frac{1}{n}\text{DFT}\left(\bar{\eta}_{,j}^1, \bar{\eta}_{,j}^2, ..., \bar{\eta}_{,j}^n\right). \tag{2.2}$$

The dominant temporal mode of the j^{th} structural mode is derived as

$$\begin{aligned}
\bar{\eta}_{\text{1st harmonic},j} &= c_{1,j}e^{i\omega t} + c_{-1,j}e^{-i\omega t} \\
&= [\Re(c_{1,j}) + \Re(c_{-1,j})]\cos(\omega t) + [\Im(-c_{1,j}) + \Im(c_{-1,j})]\sin(\omega t) + \text{pure imaginary number} \\
&= C_{c,j}\cos(\omega t) + C_{s,j}\sin(\omega t) + \text{pure imaginary number},
\end{aligned} \tag{2.3}$$

where the coefficients $C_{c,j}$ and $C_{s,j}$ are defined as

$$\begin{aligned}
C_{c,j} &= \Re(c_{1,j}) + \Re(c_{-1,j}), \\
C_{s,j} &= -\Im(c_{1,j}) + \Im(c_{-1,j}).
\end{aligned} \tag{2.4}$$

By dropping the imaginary part and using trigonometric identities, we obtain

$$\Re\left(\bar{\eta}_{\text{1st harmonic},j}\right) = \left|\bar{\eta}_{\text{1st harmonic},j}\right|\sin\left(\omega t + \theta_{\text{1st harmonic},j}\right), \tag{2.5}$$

where dominant mode magnitude and phase are given as

$$
\begin{aligned}
|\bar{\eta}_{\text{1st harmonic},j}| &= \sqrt{C_{c,j}^2 + C_{s,j}^2}, \\
\theta_{\text{1st harmonic},j} &= \sin^{-1}\left(\frac{C_{c,j}}{\sqrt{C_{c,j}^2 + C_{s,j}^2}}\right).
\end{aligned}
\tag{2.6}
$$

Finally, the prescribed motion residual are defined as

$$
\begin{aligned}
\boldsymbol{\mathcal{R}}_m &:= |\bar{\eta}_{\text{1st harmonic},j}| - \epsilon_{0,j}, \\
\boldsymbol{\mathcal{R}}_p &:= \theta_{\text{1st harmonic},j} - \theta_{0,j},
\end{aligned}
\tag{2.7}
$$

where $\epsilon_{0,j}, \theta_{0,j}$ are prescribed small motion magnitude and its phase, for the j^{th} structural mode, and $\boldsymbol{\mathcal{R}}_m, \boldsymbol{\mathcal{R}}_p$ are residuals for prescribed motion magnitude and phase, respectively. For the wing test case, the prescribed motion is applied to the first natural mode i.e. $j = 1$.

The above discussion is intended for the wing test case. The airfoil test case follows very similar derivations. The only difference is that for the airfoil test case, the constrained motion is the pitching motion α^i, $i = 1, \ldots, n$.

2.2 Time-spectral CSD equations

2.2.1 Airfoil

For the airfoil test case, we consider the two-dimensional airfoil model introduced by Isogai [53], which is shown in Fig. 2.1.

The CSD equation of motion for this model is

$$
\begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix} \begin{bmatrix} \frac{\ddot{h}}{b} \\ \ddot{\alpha} \end{bmatrix} + \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix} \begin{bmatrix} \frac{h}{b} \\ \alpha \end{bmatrix} = \frac{V_f^2}{\pi} \begin{bmatrix} -C_l \\ 2C_m \end{bmatrix},
\tag{2.8}
$$

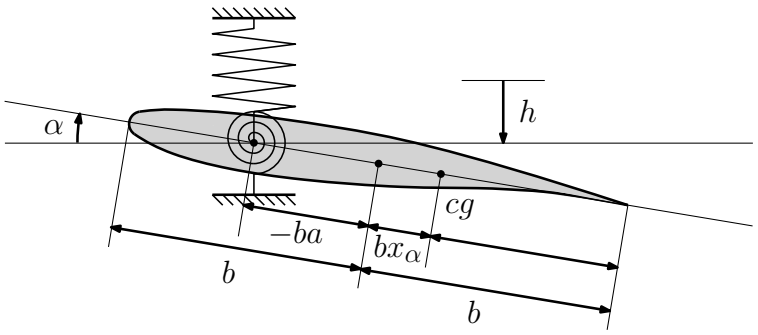


Figure 2.1: Typical section wing model; α and h represent the pitching and plunging motion, respectively; b is half chord length; cg is center of gravity; $-ba$ is the elastic center coordinate; bx_α is the center of gravity coordinate.

where x_α is the dimensionless static unbalance; r_α is the dimensionless section moment of inertia about the elastic axis, or the radius of gyration; ω_h, ω_α are the uncoupled natural frequencies of typical section in plunge and pitch, respectively; h/b is the dimensionless plunging motion, and α is the pitching motion; V_f is the LCO speed index defined as $U_f/b\omega_\alpha\sqrt{\mu}$, where μ is the airfoil mass ratio and U_f is the freestream speed. Equation (4.5) can be succinctly written as

$$\mathbf{M\ddot{u}} + \mathbf{Ku} = \mathbf{f}, \tag{2.9}$$

where

$$\mathbf{M} \coloneqq \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \ \mathbf{K} \coloneqq \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix}, \tag{2.10}$$

and

$$\mathbf{u} \coloneqq \begin{bmatrix} \frac{h}{b} \\ \alpha \end{bmatrix}, \ \mathbf{f} \coloneqq \frac{V_f^2}{\pi} \begin{bmatrix} -C_l \\ 2C_m \end{bmatrix}. \tag{2.11}$$

The time-spectral method can also be applied to the CSD equation. We pick snapshots in the time history and use spectral differentiation to get the time derivative following similar procedures

to those used for time-spectral CFD. The time-spectral CSD equation can be written as

$$\boldsymbol{\mathcal{S}}_{\text{TS}}(V_f, \omega, \mathbf{u}^n, \bar{\mathbf{f}}^n) \coloneqq \mathbf{M}^n \mathbf{D}_{\mathbf{Q}}(\omega)\mathbf{u}^n + \mathbf{K}^n \mathbf{u}^n - \frac{V_f^2}{\pi}\bar{\mathbf{f}}^n = 0 \tag{2.12}$$

where $\bar{\mathbf{f}}^n = [-C_l, 2C_m]^{n\top}$ and $\mathbf{u}^n$ denotes the displacement for all the time-instances and

$$\mathbf{M}^n \coloneqq \begin{bmatrix} \mathbf{M} & & \\ & \ddots & \\ & & \mathbf{M} \end{bmatrix}, \ \mathbf{K}^n \coloneqq \begin{bmatrix} \mathbf{K} & & \\ & \ddots & \\ & & \mathbf{K} \end{bmatrix}, \ \mathbf{D}_{\mathbf{Q}}(\omega) \coloneqq \mathbf{Q}^\top \begin{bmatrix} \mathbf{D}(\omega)^2 & & \\ & & \mathbf{D}(\omega)^2 \end{bmatrix} \mathbf{Q}, \tag{2.13}$$

The mass and stiffness matrices are repeated n times in the diagonal of $\mathbf{M}^n$ and $\mathbf{K}^n$, respectively. The permutation matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q}_{i,j} = \begin{cases} 1 & \text{if mod}(j, 2) = \lceil i/n \rceil \\ 0 & \text{otherwise} \end{cases}, \tag{2.14}$$

where i, j are index variables that are set to values between 1 and $2n$. We give an example with three time-instances for illustration in Appendix A.

2.2.2 Wing

A mode based structural model is used for the wing test case. The CSD equations are

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}, \tag{2.15}$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix, $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the stiffness matrix, $\mathbf{u} \in \mathbb{R}^N$ is the displacement, $\mathbf{f} \in \mathbb{R}^N$ is the external load, and N is the DOF of the CSD equations.

We construct the CSD equations using the structural natural mode shapes. At first, we conduct

a modal analysis,

$$\omega_j^2 \mathbf{M}\boldsymbol{\phi}_j = \mathbf{K}\boldsymbol{\phi}_j, \tag{2.16}$$

where $\omega_j, \boldsymbol{\phi}_j$ are the j^{th} natural frequency and mode shape, respectively. Rewrite Eq. (2.16) in matrix form

$$\mathbf{K}\boldsymbol{\Phi} = \mathbf{M}\boldsymbol{\Phi}\boldsymbol{\Omega}^2. \tag{2.17}$$

$\boldsymbol{\Phi}$ and $\boldsymbol{\Omega}$ are defined as

$$\begin{aligned} \boldsymbol{\Omega} &\coloneqq \text{Diag}(\omega_1, \cdots, \omega_r), \\ \boldsymbol{\Phi} &\coloneqq \begin{bmatrix} \boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_r \end{bmatrix}, \end{aligned} \tag{2.18}$$

where r is the number of modes and mode shapes computed which is typically much smaller than the structural DOF, i.e., $r \ll N$.

Next, we rewrite Eq. (2.15) in the generalized coordinates. Assuming that displacements can be approximated by $\mathbf{u} \approx \boldsymbol{\Phi}\boldsymbol{\eta}$ and pre-multiply Eq. (2.15) with $\boldsymbol{\Phi}^\intercal$, we have

$$\begin{aligned} \boldsymbol{\Phi}^\intercal \mathbf{M}\boldsymbol{\Phi}\ddot{\boldsymbol{\eta}} + \boldsymbol{\Phi}^\intercal \mathbf{K}\boldsymbol{\Phi}\boldsymbol{\eta} - \boldsymbol{\Phi}^\intercal \mathbf{f} = 0, \\ \Rightarrow \mathbf{M}_r\ddot{\boldsymbol{\eta}} + \mathbf{K}_r\boldsymbol{\eta} - \mathbf{f}_r = 0, \end{aligned} \tag{2.19}$$

where $\boldsymbol{\eta}$ is the general coordinate and subscript $\mathbf{M}_r, \mathbf{K}_r, \mathbf{f}_r$ denote the reduced or generalized mass, stiffness, and force matrices. $\mathbf{M}_r, \mathbf{K}_r$, and $\mathbf{f}_r$ are defined as

$$\begin{aligned} \mathbf{M}_r &= \boldsymbol{\Phi}^\intercal \mathbf{M}\boldsymbol{\Phi}, \\ \mathbf{K}_r &= \boldsymbol{\Phi}^\intercal \mathbf{K}\boldsymbol{\Phi}, \\ \mathbf{f}_r &= \boldsymbol{\Phi}^\intercal \mathbf{f}. \end{aligned} \tag{2.20}$$

Equation (2.19) can then be written as using Eq. (2.17)

$$\mathbf{M}_r\ddot{\boldsymbol{\eta}} + \mathbf{M}_r\boldsymbol{\Omega}^2\boldsymbol{\eta} - \mathbf{f}_r = 0. \tag{2.21}$$

Then, the time-spectral form of Eq. (2.21) can be written as,

$$\mathbf{M}_r^n \left(\mathbf{Q}^\intercal \mathbf{D}_{t,t} \mathbf{Q}\right) \boldsymbol{\eta}^n + \mathbf{M}_r (\boldsymbol{\Omega}_r^n)^2 \boldsymbol{\eta}^n - \mathbf{f}_r^n = 0, \tag{2.22}$$

where

$$\begin{aligned}
\mathbf{M}_r^n &\coloneqq \text{Diag}(\underbrace{\boldsymbol{\Phi}^\intercal \mathbf{M} \boldsymbol{\Phi}, \ldots, \boldsymbol{\Phi}^\intercal \mathbf{M} \boldsymbol{\Phi}}_{n}) = \text{Diag}(\underbrace{\mathbf{M}_r \ldots, \mathbf{M}_r}_{n}), \\
\boldsymbol{\Omega}^n &\coloneqq \text{Diag}(\underbrace{\boldsymbol{\Omega}, \ldots, \boldsymbol{\Omega}}_{n}), \\
\mathbf{Q}_{i,j} &= \begin{cases} 1 & \text{if mod}(j, r) = \lceil i/n \rceil), \\ 0 & \text{otherwise,} \end{cases} \\
\mathbf{D}_{t,t} &\coloneqq \text{Diag}(\underbrace{\mathbf{D}_t^2, \ldots, \mathbf{D}_t^2}_{r}) = \omega^2 \text{Diag}(\underbrace{\mathbf{D}^2, \ldots, \mathbf{D}^2}_{r}) = \omega^2 \bar{\mathbf{D}}^2, \\
\mathbf{f}_r^n &\coloneqq \begin{bmatrix} \boldsymbol{\Phi}^\intercal \mathbf{f}_1, & \ldots & , \boldsymbol{\Phi}^\intercal \mathbf{f}_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{f}_{r,1}, & \ldots & , \mathbf{f}_{r,n} \end{bmatrix}^\intercal.
\end{aligned} \tag{2.23}$$

Here $\mathbf{Q}$ is a permutation matrix and ω is the flow frequency as defined Section 2.1. Together with the second order spectral derivative matrix $\mathbf{D}_{t,t}$, the second time derivatives of state variables for different modes are obtained,

$$\ddot{\boldsymbol{\eta}}^n = \left(\mathbf{Q}^\intercal \mathbf{D}_{t,t} \mathbf{Q}\right) \boldsymbol{\eta}^n. \tag{2.24}$$

Finally, we derive the dimensionless form of Eq. (2.22). The aerodynamic forces are normalized by the dynamic pressure $q_\infty = (1/2)\, \rho_\infty U_\infty^2$ and the reference S_{ref} by the following equation

$$\bar{\mathbf{f}} = \frac{\mathbf{f}}{\frac{1}{2} \rho_\infty U_\infty^2 S_{\text{ref}}}, \tag{2.25}$$

where $\mathbf{f}$ is the dimensional load and $\bar{\mathbf{f}}$ is the dimensionless load. It follows that the normalized

generalized aerodynamic forces are then written as

$$\bar{\mathbf{f}}_r^n \coloneqq \begin{bmatrix} \boldsymbol{\Phi}^\intercal \bar{\mathbf{f}}_1 \\ \vdots \\ \boldsymbol{\Phi}^\intercal \bar{\mathbf{f}}_n \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{f}}_{r,1} \\ \vdots \\ \bar{\mathbf{f}}_{r,n} \end{bmatrix}. \tag{2.26}$$

To nondimensionalize Eq. (2.22) we use the wing mass m_0, the semi-chord b and the first torsion mode natural frequency, $\omega_\alpha = \omega_2$, which in this case is the second natural mode. The dimensionless CSD equation can then be written in residual form as,

$$\begin{aligned} \boldsymbol{\mathcal{S}}_{\text{TS}} &\coloneqq \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{Q}^\intercal \bar{\mathbf{D}}^2 \mathbf{Q}\right)\left(\frac{\boldsymbol{\eta}^n}{b}\right) + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\left(\frac{\boldsymbol{\eta}^n}{b}\right) - \frac{1}{2}\frac{\rho_\infty U_\infty^2 S_{\text{ref}}}{m_0 \omega_\alpha^2 b}\bar{\mathbf{f}}_r^n \\ &= \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{Q}^\intercal \bar{\mathbf{D}}^2 \mathbf{Q}\right)\bar{\boldsymbol{\eta}}^n + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\bar{\boldsymbol{\eta}}^n - \frac{1}{2}\left(\frac{S_{\text{ref}} b}{V_0}\right) V_f^2 \bar{\mathbf{f}}_r^n = 0, \end{aligned} \tag{2.27}$$

where the following dimensionless coefficients have been introduced and are defined as,

$$\begin{aligned} \mu &\coloneqq \frac{m_0}{\rho_\infty V_0}, \\ V_f &\coloneqq \frac{U_\infty}{\sqrt{\mu}\omega_\alpha b}, \\ \bar{\boldsymbol{\eta}}^n &\coloneqq \frac{\boldsymbol{\eta}^n}{b}. \end{aligned} \tag{2.28}$$

Here, μ is the mass ratio, V_0 is the volume of a conical frustum having root chord as lower base diameter, tip chord as upper base diameter, and panel span as height and V_f is the LCO speed index.

2.3 Time-spectral CFD equations

The aerodynamic model used for this work is the ADflow CFD solver [63, 92][1] a parallel, finite-volume, cell-centered, multiblock, and overset code that solves the Euler and RANS equations in either steady, unsteady, or time-spectral modes. For unsteady applications, the second-order implicit backward difference formula (BDF2) time integration scheme is used. For moving meshes, an arbitrary Lagrangian Eulerian (ALE) formulation satisfying GCL [138] has also been implemented in ADflow by Huang and Friedmann [52]. In this work, we consider both the Euler and the RANS equations. For the RANS equations, we use the Spalart–Allmaras (SA) turbulence model [87, 123]. The time-spectral solver in ADflow has second-order accuracy in space [90]. For the time-space parallelization strategy, our code is parallelized by block instead of time-instance Mader and Martins [90], Mader [93].

The time-spectral method is well-established in CFD [39, 44]. The method converts an unsteady CFD problem into a series of time-coupled steady-state problems. The equations generated by this set of coupled steady-state problems have two additional parameters: the time period considered (T) and the number of time-instances or points (n) to be solved within that period. If we write the time-dependent residual as $\boldsymbol{\mathcal{A}}(\boldsymbol{\zeta}(t)) = 0$, then the time-spectral form is $\boldsymbol{\mathcal{A}}(\boldsymbol{\zeta}^n, T) = 0$, where $\boldsymbol{\zeta}^n$ represents the state variables for all time-instances, i.e., n times the size of the steady-state solution.

The residual form of the time-spectral CFD equations can be written as

$$\boldsymbol{\mathcal{A}}_{\text{TS}} \coloneqq \mathbf{D}(\omega)\boldsymbol{\zeta}^n + \mathbf{R}(\boldsymbol{\zeta}^n) = 0. \tag{2.29}$$

When solving the Euler equations, $\mathbf{R}(\boldsymbol{\zeta}^n)$ is the inviscid flow residual vector, and $\boldsymbol{\zeta}^n$ is the vector of inviscid flow states for all time-instances. When solving the RANS equations, $\mathbf{R}(\boldsymbol{\zeta}^n)$ is the viscous flow residual vector concatenated with the SA turbulence model residuals, and $\boldsymbol{\zeta}^n$ is the

[1] https://github.com/mdolab/adflow

vector of viscous flow states concatenated with the SA turbulence model states. The matrix $\mathbf{D}(\omega)$ is an $n \times n$ matrix that depends on the angular velocity $\omega = 2\pi/T$, and is defined as

$$\mathbf{D}_{i,j}(\omega) = \begin{cases} \dfrac{\omega(-1)^{(j-i)}}{2\sin\left(\pi(j-i)/n\right)}, & \text{if } i \neq j, \\ 0, & \text{if } i = j, \end{cases} \tag{2.30}$$

where $\mathbf{D}_{i,j}$ is the entry at the i^{th} row and the j^{th} column of the matrix $\mathbf{D}$ and $i, j \in \{1, \ldots, n\}$. When more time-instances are added (for higher-frequency terms), the problem size increases, making it more challenging to converge the residual due to the larger off-diagonal terms in $\mathbf{D}$.

The LCO speed index (V_f) is also a variable. It affects the CFD equations through the boundary condition (or through the mesh velocity, if the air is set to be at rest, and the airfoil and the mesh are in motion). For reasons mentioned in the following section, the mesh nodal coordinates also affect the results. A more general form of the CFD equations is

$$\boldsymbol{\mathcal{A}}_{\text{TS}}(V_f, \omega, \boldsymbol{\zeta}^n, \mathbf{X}_V^n) = 0, \tag{2.31}$$

where $\mathbf{X}_V^n$ represents the mesh volume nodal coordinates for all time-instances. Equation (2.31) is efficiently solved using the approximate Newton–Krylov (ANK) method [148]. ANK allows for a robust start-up and a rapid terminal convergence of the flow solver.

2.3.1 Mesh deformation

Mesh quality is important for reliable results. In an aeroelastic computation, the geometry is altered when the structure deforms and the mesh is required to adjust accordingly. To ensure the quality of the deformed mesh, we use an analytic inverse distance method implemented in an open-source package IDWarp [119]. [2] Using this method, the displacements of the CFD volume

[2] https://github.com/mdolab/idwarp

mesh are a combination of all surface deformations weighted by the inverse of the distance to each surface node. The computational cost of a naive implementation of this method scales with the number of surface nodes. However, with a suitable fast spatial search algorithm and multipole-like expansion of the summation, the cost can be reduced to $\mathcal{O}(\log N)$[83]. IDWarp is fast and robust enough to be used in aerostructural optimization with large deflections [17, 18]. Therefore, we expect this algorithm to be able to handle the LCO displacements. This remains to be tested, in particular for cases with viscous wing meshes with large deformation under LCO. For a typical aerostructural analysis, the mesh movement scheme requires only 2–3% of the total solution time. With the mesh deformation algorithm, we have:

$$\mathbf{X}_S^n = \boldsymbol{\mathcal{G}}(\mathbf{u}^n), \tag{2.32}$$

$$\mathbf{X}_V^n = \boldsymbol{\mathcal{W}}(\mathbf{X}_S^n), \tag{2.33}$$

where the structural displacements ($\mathbf{u}^n$) are used to compute the updated aerodynamic surface coordinates ($\mathbf{X}_S^n$), which in turn are used to compute the updated deformed volume coordinates ($\mathbf{X}_V^n$). Given these quantities and the spectral differentiated mesh velocity, we can rewrite the CFD residual form in terms of the structural displacement as

$$\boldsymbol{\mathcal{A}}_{\text{TS}}(V_f, \omega, \boldsymbol{\zeta}^n, \mathbf{u}^n) = 0, \tag{2.34}$$

which is solved together with other components to capture the LCO response.

2.3.2 Mesh velocity computation

As shown in Eq. (2.31), the aerodynamic residual $\boldsymbol{\mathcal{A}}_{\text{TS}}$ is dependent on the volume mesh coordinates. One source of this dependency is through the computation of the flux term through a moving mesh surface. For a dynamic mesh CFD solution, the relative velocity is needed for flux

calculation, which we write as

$$V_x = v_x - v_{x,g}, \tag{2.35}$$

$$V_y = v_y - v_{y,g}, \tag{2.36}$$

$$V_z = v_z - v_{z,g}, \tag{2.37}$$

where V_x, V_y, and V_z are relative velocities, v_x, v_y, and v_z are the absolute velocities, and $v_{x,g}, v_{y,g}$, and $v_{z,g}$ are the surface mesh cell center velocities. This is a new feature that we add to ADflow.

We solve for the mesh velocity by spectral differentiation. The mesh motion can be approximated as a sum of harmonic functions,

$$x_g \approx \sum_{k=-(n-1)/2}^{(n-1)/2} \hat{x}_k e^{i\frac{2\pi k}{T}t}. \tag{2.38}$$

Only the x coordinate is shown here, but the y and z coordinates have the same form. Here, we assume that the total number of time-instances (n) is odd. If x_g is the x coordinate of a node in the mesh, $\hat{x}_k$ are their corresponding Fourier series coefficients. Using the approach used for the approximation of the temporal derivative term in time-spectral CFD [39, 44], we have:

$$\dot{\mathbf{x}}_g^n \approx \mathbf{D}(\omega)\mathbf{x}_g^n, \tag{2.39}$$

where $\dot{\mathbf{x}}_g^n$ is the vector of true mesh nodal velocities for all time-instances, and $\mathbf{x}_g^n$ are the mesh nodal coordinates for all time-instances.

The surface mesh cell center velocities $v_{x,g}, v_{y,g}$, and $v_{z,g}$, are approximated by averaging of the nodal velocities. Benoit and Nadarajah [9] observe that for an airfoil with a pitching magnitude of 5°, the maximum C_L computed by a solver satisfying GCL and another solver not satisfying GCL differs by less than 0.5%. However, for another case with 20° pitching amplitude, the value increases to 15.21%. In our case, since for all cases the pitching amplitude is within 2°, we expect GCL not having a significant impact on the solution. We discuss GCL in Section 2.3.3 and we

present two cases in Section 6.1.2.2 that support the claims made by Benoit and Nadarajah [9].

2.3.3 GCL

We propose a simple method to enforce the GCL that is based on the assumption that all nodes undergo harmonic motions [9]. Without GCL, we compute the area rate, $\dot{\mathbf{A}}^n$, and the area rate swept by each edge, $(\mathrm{d}A/\mathrm{d}t)_{i,j}$, by using the following formulae,

$$\begin{aligned} \dot{\mathbf{A}}^n &= \mathbf{D}(\omega)\mathbf{A}^n, \\ \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)_{i,j} &= \left(\frac{\mathbf{V}_i + \mathbf{V}_j}{2} \times \mathbf{v}_{i,j}, \mathbf{k}\right), \end{aligned} \tag{2.40}$$

where $\mathbf{A}^n = [A_1, \ldots, A_k, \ldots A_n]^\intercal$, A_k is the area of a cell from the k^{th} time-instance, $(\mathrm{d}A/\,\mathrm{d}t)_{i,j}$ denotes the area rate swept by an edge i, j (where i, j are the nodal indices), $\mathbf{V}_i$ and $\mathbf{V}_j$ are nodal velocities computed using Eq. (2.39), $\mathbf{v}_{i,j}$ denotes the vector $\mathbf{x}_j - \mathbf{x}_i$, and $\mathbf{k}$ is a unit vector perpendicular to the plane that the airfoil belongs to.

In general, the GCL is **not** satisfied using Eq. (2.40), i.e.,

$$\frac{\mathrm{d}A}{\mathrm{d}t} \neq \sum_{(i,j)\in\mathbb{E}} \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)_{i,j}, \tag{2.41}$$

where $\mathbb{E}$ is the set of all edges from a cell.

We enforce the GCL by keeping the second equation and removing the first equation from Eq. (2.40). We reconstruct the area rate from the rate swept by each surface by using the following formulae

$$\begin{aligned} \frac{\mathrm{d}A}{\mathrm{d}t} &= \sum_{(i,j)\in\mathbb{E}} \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)_{i,j}, \\ \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)_{i,j} &= \left(\frac{\mathbf{V}_i + \mathbf{V}_j}{2} \times \mathbf{v}_{i,j}, \mathbf{k}\right). \end{aligned} \tag{2.42}$$

Thus, the GCL is satisfied by construction.

The area rate swept by an edge used in Eq. (2.42) was verified with the analytic area rate for the quadrilateral element case proposed by Benoit and Nadarajah [9]. As shown in Fig. 2.2, the rate computed with the time-spectral method matches the analytic solution.

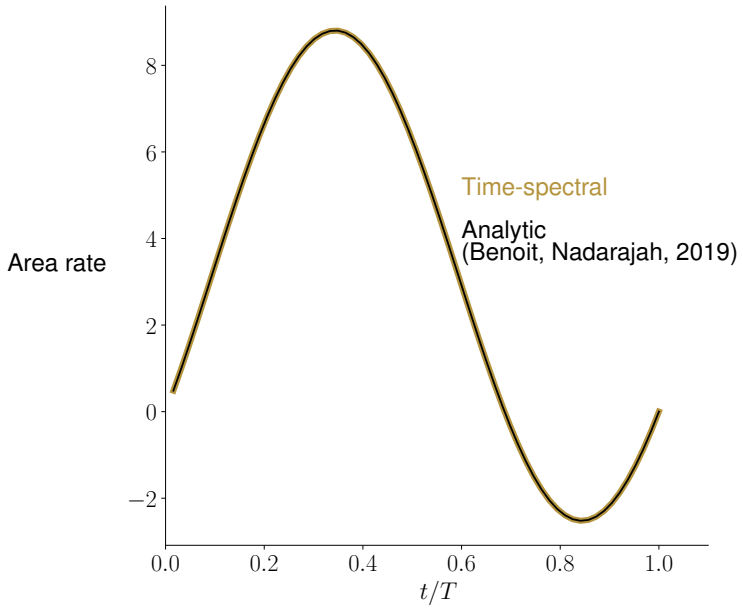


Figure 2.2: The implemented GCL compares well with analytic results Benoit and Nadarajah [9].

2.3.4 Boundary condition for the airfoil test case

The boundary conditions for the airfoil and the wing test cases are set in different ways following different conventions by Isogai [53], Yates [147], respectively. For the airfoil test case, the boundary condition for the CFD solver are composed of (T_∞, p_∞, M). The pressure, p_∞ is fixed to 101325.0 Pa and the Mach number, M is also given. The temperature, T_∞ is determined by V_f

and M. Using the definition of LCO speed index, Mach number, and the ideal gas law,

$$\begin{aligned} U_\infty &= V_f b \omega_\alpha \sqrt{\mu}, \\ a &= \frac{U_\infty}{M}, \\ T_\infty &= \frac{a^2}{\gamma R}, \end{aligned} \tag{2.43}$$

where a is speed of sound, we can compute the temperature.

2.3.5 Boundary condition for the wing test case

Following the convention by Yates [147], the triplet (T_∞, p_∞, M) is determined differently compared with the airfoil test case. In this analysis, we define the problem in terms of (M, μ, V_f). Thus, we need to compute (T_∞, p_∞). Here we detail the procedure to compute the boundary conditions. Flow density can be computed from previously defined nondimesnional coefficients in Eq. (2.28),

$$\rho_\infty = \frac{m_0}{\mu V_0}. \tag{2.44}$$

Similarly, from Eq. (2.28) and the dynamic pressure, the static pressure, p_∞ is obtained as

$$\begin{aligned} U_\infty &= V_f b \omega_\alpha \sqrt{\mu}, \\ q_\infty &= \frac{1}{2} \rho_\infty U_\infty^2, \\ p_\infty &= \frac{2 q_\infty}{\gamma M^2}. \end{aligned} \tag{2.45}$$

Finally, the temperature T_∞ is found by the ideal gas law,

$$T_\infty = \frac{p_\infty}{\rho_\infty R}, \tag{2.46}$$

where R is the gas constant for air.

Here, M and μ are taken as parameters whereas V_f is taken as an independent state variable. Subsequently, we can define the static pressure and temperatures as a function of the LCO speed index,

$$\begin{aligned} T_\infty &= T_\infty(V_f), \\ p_\infty &= p_\infty(V_f). \end{aligned} \tag{2.47}$$

2.3.6 Load calculation for the airfoil test case

The computation of aerodynamic loads for the airfoil test cases boils down to the computation of lift and moment coefficient, C_l and C_m, respectively. They are straightforward to compute and the derivation is omitted. For more detail, the readers are referred to the standard textbook by Anderson [3].

2.3.7 Load calculation for the wing test case

For the wing test case, the load transfer component need to know the aerodynamic load in dimensionless form. The aerodynamic load is computed at each nodes for each time-instance i

$$\mathbf{f}_i = \mathbf{f}_i(\mathbf{X}_{S,i}, \boldsymbol{\zeta}_i), \tag{2.48}$$

where $\mathbf{X}_{S,i}$ is the surface mesh for time-instance i. The dimensionless aerodynamic load is as previously defined Eq. (2.26)

$$\bar{\mathbf{f}}_i = \frac{1}{q_\infty S_{\text{ref}}} \mathbf{f}_i(\mathbf{X}_{S,i}, \boldsymbol{\zeta}_i).$$

Furthermore, since $U_\infty = U_\infty(V_f)$, we finally have

$$\bar{\mathbf{f}}_i = \bar{\mathbf{f}}_i(V_f, \mathbf{X}_{S,i}, \boldsymbol{\zeta}_i). \tag{2.49}$$

2.4 CFD–CSD load and displacement transfer

Between the CFD and CSD components, the displacement information is transferred from the CSD component to CFD component, and the load information is transferred in the opposite direction. For the airfoil test case, the displacement transfer is reduced to a rigid body motion problem and the load transfer is reduced to the computation of C_l and C_m as discussed before. These operations are simple to do and thus are omitted here. However, the information transfer for the wing test case is relatively complicated and we have done some new development here. This is detailed in this section.

2.4.1 Displacement transfer

In general, the aerodynamic and structural grids do not have the same topology or match in terms of surface grid point locations. Thus, an interpolation scheme is needed to transfer both loads and displacements the is suitable for not matching grids. In this work, a relatively simple strategy is adapted. Given the structure mode shapes $\mathbf{\Phi}$ at coordinates $\mathbf{X}_{\text{mode}}$ we fit $\mathbf{\Phi}$ with a fourth-order polynomial which we denote as $\breve{\mathbf{\Phi}}$. The aerodynamic mode shapes can then be computed from this function by evaluating it using the jig shape i.e. $\breve{\mathbf{\Phi}}_3(\mathbf{X}_J)$.

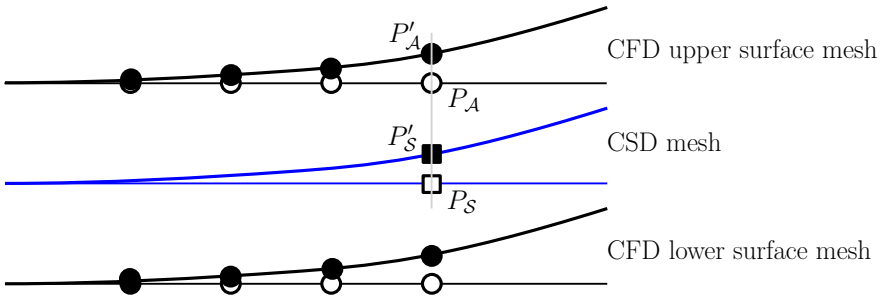


Figure 2.3: Transfer illustration. $P_{\mathcal{S}}$ is a projection of $P_{\mathcal{A}}$ on the structure mode surface. $|P_{\mathcal{A}}P'_{\mathcal{A}}| = |P_{\mathcal{S}}P'_{\mathcal{S}}|$.

The displacement of the CFD nodes in the i^{th} time-instance can be defined in term of the

aerodynamic modes shapes as,

$$\bar{\mathbf{u}}_{3,i} = \breve{\mathbf{\Phi}}_3(y_1, y_2)\bar{\boldsymbol{\eta}}_i. \tag{2.50}$$

Here we ignore any displacement in u_1 and u_2 since the magnitude of these two components is much smaller than compared with u_3. The surface deformation of the aerodynamic mesh can then be written as

$$\mathbf{X}_{S,i} = \mathbf{X}_J + \begin{bmatrix} 0 \\ 0 \\ \breve{\mathbf{\Phi}}_3(\mathbf{X}_J) \end{bmatrix} \bar{\boldsymbol{\eta}}_i b. \tag{2.51}$$

where $\mathbf{X}_J$ surface coordinates of the undeformed aerodynamic mesh i.e. the jig shape coordinates.

2.4.2 Load transfer

The virtual work on the CFD mesh after a small deformation can be written as

$$\begin{aligned} \delta\overline{W}_{\text{CFD},i} &= -\left(\bar{\mathbf{f}}_{3,i}(V_f, \boldsymbol{\zeta}_i, \mathbf{X}_{S,i})\right)^{\intercal} \delta\bar{\mathbf{u}}_{3,i} \\ &= -\left(\bar{\mathbf{f}}_{3,i}(V_f, \boldsymbol{\zeta}_i, \mathbf{X}_{S,i})\right)^{\intercal} \breve{\mathbf{\Phi}}_3(\mathbf{X}_J)\delta\bar{\boldsymbol{\eta}}_i, \end{aligned} \tag{2.52}$$

where Eq. (2.50) was applied. The virtual work on the CSD mesh is given as

$$\delta\overline{W}_{\text{CSD},i} = \left(\bar{\mathbf{f}}_{r,i}(\mathbf{X}_J, V_f, \boldsymbol{\zeta}_i, \mathbf{X}_{S,i})\right)^{\intercal} \delta\bar{\boldsymbol{\eta}}_i. \tag{2.53}$$

To make the transfer consistent, we have

$$\delta\overline{W}_{\text{CFD},i} + \delta\overline{W}_{\text{CSD},i} = 0, \tag{2.54}$$

for all virtual displacement. This gives

$$\bar{\mathbf{f}}_{r,i}(\mathbf{X}_J, V_f, \boldsymbol{\zeta}_i, \mathbf{X}_{S,i}) = \breve{\boldsymbol{\Phi}}_3(\mathbf{X}_J)^\intercal \left(\bar{\mathbf{f}}_{3,i}(V_f, \boldsymbol{\zeta}_i, \mathbf{X}_{S,i})\right). \tag{2.55}$$

We choose this light-weight transfer scheme because it is easier to implement than more involving transfer scheme as discussed by Kenway et al. [62].

2.5 AD

AD is a well known approach based on the systematic application of the differentiation chain rule to computer programs [40, 96, 106]. When implemented appropriately, AD can achieve machine precision. Its computational cost can be either proportional to the number of inputs or the number of outputs depending on the mode it is implemented with. We cover the two modes in the following sections. For a more detailed discussion on AD, we refer the readers to the textbook by Martins and Ning [96].

2.5.1 Variables and functions as lines of code

We represent the variables of the computer code by a sequence

$$v = v_1, v_2, \ldots, v_N. \tag{2.56}$$

Parts of v overlap with the inputs $\mathbf{x}$, and outputs $\mathbf{f}$. The rest of v is the intermediate variables. In general, a variable assignment corresponding with a line of code can be dependent on variables including itself:

$$v_i = V_i(v_1, \ldots, v_N), \tag{2.57}$$

where $V_i(\cdot)$ is an explicit function. By introducing additional variables, we can avoid variable substitution, and unroll the function such that a variable assignment is only dependent on the variables assigned previously. Then, we have

$$v_i = V_i(v_1, \ldots, v_{i-1}). \tag{2.58}$$

Using this definition, we can derive the derivatives using the chain rule. There are two modes of the chain rules. In the *forward mode*, we fix one input and compute all the output derivatives with this input. In the *reverse mode*, we fix one output and compute the derivative with respect to all the inputs.

2.5.2 FAD

The chain rule for the forward mode can be written as:

$$\frac{\mathrm{d}v_i}{\mathrm{d}v_j} = \sum_{k=j}^{i-1} \frac{\partial V_i}{\partial v_k} \frac{\mathrm{d}v_k}{\mathrm{d}v_j}. \tag{2.59}$$

For the forward mode, we fix j and incrementing i to get the derivative of all variables with respect to v_j. For a fixed input j, we define that

$$\dot{v}_i \coloneqq \frac{\mathrm{d}v_i}{\mathrm{d}v_j}. \tag{2.60}$$

We name $\dot{v}_i$ as the *forward seed*.

Supposing we have the following sequence of variables (v_1, v_2, v_3, v_4) where v_1, v_2 are the inputs, x_1, x_2, v_3 is an intermediate variable, and v_4 is the output, f. To compute the derivative of f with respect to x_2, we set $\dot{v}_2 = 1$, and increment i using Eq. (2.59). We collect the output and obtain $\dot{v}_4$ that equals $\mathrm{d}y/\,\mathrm{d}x_2$.

For the directional derivative, i.e., when we want to compute the derivative in the direction of a vector $\mathbf{c} = \mathbb{R}^{n_\mathbf{x}}$. This can be easily done by adding one additional node v_0, and edges $v_i = c_i v_0$ for $i = 1, \ldots, n_\mathbf{x}$. Then, by computing $\mathrm{d}v_i / \mathrm{d}v_0$, we obtain the directional derivative. Alternatively, we can simply set $\dot{v}_i$ to be equal to c_i for $i = 1, \ldots, n_\mathbf{x}$, and we will get the same result. In the rest of the section, we directly set the input forward seeds.

The number of FAD code calls is proportional to the number of inputs and is independent of the number of outputs.

2.5.3 RAD

The reverse mode is also based on the chain rule. It can be written as

$$\frac{\mathrm{d}v_i}{\mathrm{d}v_j} = \sum_{j+1}^{i} \frac{\partial V_k}{\partial v_j} \frac{\mathrm{d}v_i}{\mathrm{d}v_k}. \tag{2.61}$$

For a fixed output i, we define that

$$\bar{v}_j \coloneqq \frac{\mathrm{d}v_i}{\mathrm{d}v_j}. \tag{2.62}$$

We name $\bar{v}_j$ as the *reverse seed*.

Consider the same example used in the previous section. To compute the derivative of f with respect to x_2, we set $\bar{v}_4 = 1$, and increment j using Eq. (2.61). We collect the output and obtain $\bar{v}_2$ that equals $\mathrm{d}y / \mathrm{d}x_2$.

When we compute the derivative for the weighted output with the weight, $\mathbf{w} \in \mathbb{R}^{n_\mathbf{f}}$, we can simply add another node v_{N+1}, and edges based on $v_{N+1} = \sum_{j=1}^{n_\mathbf{f}} w_j v_{N-n_\mathbf{f}+1+j}$. By setting $\bar{v}_{N+1} = 1$, and collect $\bar{v}_i$, we obtain the total derivative of the weight output. Alternatively, we can simply set $\bar{v}_{N-n_\mathbf{f}+1+j}$ to be equal to w_j for $j = 1, \ldots, n_\mathbf{f}$, and we will get the same result. In the rest of the section, we directly set the input reverse seeds.

Different from the FAD method, for the RAD method, the number of RAD code calls is pro-

portional to the number of inputs and is independent of the number of outputs.

2.5.4 Example

2.5.4.1 Equivalency of FAD and RAD

The FAD and RAD formulae are two different ways to express the same underlying derivatives. We demonstrate it using a simple example with $v = v_1, v_2, v_3$, where v_1 and v_3 are the only input and output, respectively. The goal is to compute $\mathrm{d}v_3/\,\mathrm{d}v_1$.

Using Eq. (2.59), we set that $\dot{v}_1 = 1$, and we have

$$\begin{aligned}\frac{\mathrm{d}v_3}{\mathrm{d}v_1} &= \frac{\partial V_3}{\partial v_1}\frac{\mathrm{d}v_1}{\mathrm{d}v_1} + \frac{\partial V_3}{\partial v_2}\frac{\mathrm{d}v_2}{\mathrm{d}v_1} \\ &= \frac{\partial V_3}{\partial v_1}\dot{v}_1 + \frac{\partial V_3}{\partial v_2}\frac{\mathrm{d}v_2}{\mathrm{d}v_1} \\ &= \frac{\partial V_3}{\partial v_1} + \frac{\partial V_3}{\partial v_2}\frac{\mathrm{d}v_2}{\mathrm{d}v_1}.\end{aligned} \tag{2.63}$$

Alternatively, using Eq. (2.61), we set that $\bar{v}_3 = 1$, we have

$$\begin{aligned}\frac{\mathrm{d}v_3}{\mathrm{d}v_1} &= \frac{\partial V_2}{\partial v_1}\frac{\mathrm{d}v_3}{\mathrm{d}v_2} + \frac{\partial V_3}{\partial v_1}\frac{\mathrm{d}v_3}{\mathrm{d}v_3} \\ &= \frac{\partial V_2}{\partial v_1}\frac{\mathrm{d}v_3}{\mathrm{d}v_2} + \frac{\partial V_3}{\partial v_1}\bar{v}_3 \\ &= \frac{\partial V_2}{\partial v_1}\frac{\mathrm{d}v_3}{\mathrm{d}v_2} + \frac{\partial V_3}{\partial v_1}.\end{aligned} \tag{2.64}$$

Finally, we need to use the identity that

$$\frac{\mathrm{d}v_{i+1}}{\mathrm{d}v_i} = \frac{\partial V_{i+1}}{\partial v_i}, \tag{2.65}$$

which can be derived by plugging in $j = i-1$ into Eq. (2.59). Using this identity, we have $\mathrm{d}v_3/\,\mathrm{d}v_1$ derived from Eq. (2.63) and Eq. (2.64) are indeed identical.

2.5.4.2 Superiority of RAD method for problems with fewer outputs than inputs

As we mentioned earlier, RAD method outperforms FAD method for problems with fewer outputs than inputs. We demonstrate this point by the following example.

The function is defined as

```
def f(x1, x2):
    return x2 ** 2, sin(x1 * x2)
```

We want to compute $(\mathrm{d}\mathbf{f}_2/\,\mathrm{d}\mathbf{x})$ at the point $\mathbf{x} = (\pi/6, 1)$.

At first, we unroll the code by defining new intermediate variables

$$\begin{aligned} v_1 &= x_1, \\ v_2 &= x_2, \\ v_3 &= v_1 v_2, \\ v_4 &= v_2^2, \\ v_5 &= \sin v_3. \end{aligned} \tag{2.66}$$

We can differentiate the code using FAD and RAD. For the FAD method, we at first set $\dot{v}_1 = 1$ to compute $\mathrm{d}\mathbf{f}_2/\,\mathrm{d}\mathbf{x}_1$. Using Eq. (2.59), we have

$$\begin{aligned} \dot{v}_1 &= 1, \\ \dot{v}_2 &= 0, \\ \dot{v}_3 &= \frac{\partial V_3}{\partial v_1}\dot{v}_1 + \frac{\partial V_3}{\partial v_2}\dot{v}_2 = v_2 = 1, \\ \dot{v}_4 &= \frac{\partial V_4}{\partial v_1}\dot{v}_1 + \frac{\partial V_4}{\partial v_2}\dot{v}_2 + \frac{\partial V_4}{\partial v_3}\dot{v}_3 = 0, \\ \dot{v}_5 &= \frac{\partial V_5}{\partial v_1}\dot{v}_1 + \frac{\partial V_5}{\partial v_2}\dot{v}_2 + \frac{\partial V_5}{\partial v_3}\dot{v}_3 + \frac{\partial V_5}{\partial v_4}\dot{v}_4 = \cos v_3 \dot{v}_3 = \frac{\sqrt{3}}{2}. \end{aligned} \tag{2.67}$$

Thus, we have $\mathrm{d}\mathbf{f}_2/\,\mathrm{d}\mathbf{x}_1 = \sqrt{3}/2$. Similarly, by setting $\dot{v}_2 = 1$ and conducting a similar computa-

tion, we find that $\mathrm{d}\mathbf{f}_2/\,\mathrm{d}\mathbf{x}_2 = \left(\sqrt{3}\pi\right)/12$.

Alternatively, for the RAD method, we set $\bar{\mathbf{f}}_2 = 1$, i.e., $\bar{v}_5 = 1$. Using Eq. (2.61), we have

$$
\begin{aligned}
&\bar{v}_5 = 1,\\
&\bar{v}_4 = 0,\\
&\bar{v}_3 = \frac{\partial V_4}{\partial v_3}\bar{v}_4 + \frac{\partial V_5}{\partial v_3}\bar{v}_5 = \cos v_3 \bar{v}_5 = \frac{\sqrt{3}}{2},\\
&\bar{v}_2 = \frac{\partial V_3}{\partial v_2}\bar{v}_3 + \frac{\partial V_4}{\partial v_2}\bar{v}_4 + \frac{\partial V_5}{\partial v_2}\bar{v}_5 = v_1\bar{v}_3 + 2v_2\bar{v}_4 = \frac{\sqrt{3}\pi}{12},\\
&\bar{v}_1 = \frac{\partial V_2}{\partial v_1}\bar{v}_2 + \frac{\partial V_3}{\partial v_1}\bar{v}_3 + \frac{\partial V_4}{\partial v_1}\bar{v}_4 + \frac{\partial V_5}{\partial v_1}\bar{v}_5 = v_2\bar{v}_3 = \frac{\sqrt{3}}{2}.
\end{aligned}
\tag{2.68}
$$

For this example, we have one output function and two input variables. For the RAD method, it requires one computation to obtain the derivatives. While, for the FAD method, it requires two computations. Thus, the RAD method is better than FAD method for this example. Generally speaking, for problems with weighted output with more than one design variable, the RAD method is superior. This is exactly the reason we develop the RAD formulae in Chapter 5.

CHAPTER 3

Time-Spectral Aeroelastic Equations and Jacobian-Free Newton–Krylov Solver

In this chapter, we present our LCO solution method based on the CNK method. The derivation is written for the wing test case. For the airfoil case, we only need to switch the underlying modules as discussed in Chapter 2. For example, the displacement for the wing test case is defined as $\boldsymbol{\eta}^n$, and for the airfoil test case, the displacements are $\boldsymbol{\alpha}^n$ and $\boldsymbol{h}^n$ for pitching and plunging, respectively. The chapter is organized as follows: In Section 3.1, we present the time-spectral aeroelastic equations that are the governing equation of an LCO. Then, in Section 3.2, we discuss our proposed method to solve the time-spectral aeroelastic equations.

3.1 Time-spectral aeroelastic equations

In this section, we explain the motivation and the components of the time-spectral aeroelastic equations. As discussed in the introduction, we are interested in finding the LCO speed index, V_f, and the LCO frequency, ω for the prediction of an LCO. Considering those variables as state variables, in addition to the spectral CSD and CFD state variables ($\boldsymbol{\eta}^n$ and $\boldsymbol{\zeta}^n$, respectively), we can form a state vector, $\mathbf{q}$, of $2 + N_{\text{CSD}} \times n + N_{\text{CFD}} \times n$ states. However, once we take the CSD and CFD equations into account, we only have $N_{\text{CSD}} \times n + N_{\text{CFD}} \times n$ equations. Therefore, two more equations are needed to make sure there are equal numbers of equations and variables.

Without additional constraints, any point in Fig. 1.2 from either curve is a feasible solution satisfying both CSD and CFD equations. By specifying a constraint on the magnitude of the motion, the solution is limited to one point on the curve. This leaves the solution the freedom to shift the phase. Thus, an equation is added to constrain the phase, and a unique solution is obtained. This formulation was originally proposed by Thomas et al. [137] using the harmonic-balance method.

The time-spectral aeroelastic system of equations in residual form are defined as the aggregation of the motion equation residuals for the magnitude and phase ($\boldsymbol{\mathcal{R}}_m$ and $\boldsymbol{\mathcal{R}}_p$, respectively), the CSD equation residuals ($\boldsymbol{\mathcal{S}}_{\text{TS}}$), and the CFD equation residuals ($\boldsymbol{\mathcal{A}}_{\text{TS}}$):

$$\boldsymbol{\mathcal{R}}(\mathbf{q}) \coloneqq \begin{bmatrix} \boldsymbol{\mathcal{R}}_m \\ \boldsymbol{\mathcal{R}}_p \\ \boldsymbol{\mathcal{S}}_{\text{TS}} \\ \boldsymbol{\mathcal{A}}_{\text{TS}} \end{bmatrix}, \tag{3.1}$$

where the state variable vector $\mathbf{q}$ is defined as

$$\mathbf{q} \coloneqq \begin{bmatrix} V_f \\ \omega \\ \boldsymbol{\eta}^n \\ \boldsymbol{\zeta}^n \end{bmatrix}. \tag{3.2}$$

The above formulation is general in the sense that different fidelities could be used. For instance, a full finite-element model can be applied to compute $\boldsymbol{\mathcal{S}}_{\text{TS}}$ instead of the simpler two-dimensional spring model used in this work. We use the equation to predict both flutter and LCO. The difference is that for predicting the flutter point, a very small amplitude is used, and solving for LCO assumes larger amplitudes. In the following sections, we detail the components of

Eq. (3.1) specific to this work. Finally, in Section 3.3, we discuss the preconditioner design which is a critical element for the Krylov subspace methods.

3.2 Time-spectral aeroelastic solution

Now we present the solver for the time-spectral aeroelastic equations (3.1), which consists of a preconditioned coupled Jacobian-free Newton–Krylov method. Applying Newton's method to Eq. (3.1) is solved using Newton's method, which results in the following linear system:

$$\begin{aligned} \mathbf{J}\Delta\mathbf{q} &= -\boldsymbol{\mathcal{R}}(\mathbf{q}^{(k)}), \\ \mathbf{q}^{(k+1)} &= \mathbf{q}^{(k)} + \theta\Delta\mathbf{q}, \end{aligned} \tag{3.3}$$

where

$$\mathbf{J} = \frac{\partial\boldsymbol{\mathcal{R}}(\mathbf{q})}{\partial\mathbf{q}}\Big|_{\mathbf{q}=\mathbf{q}^{(k)}} \tag{3.4}$$

is the Jacobian evaluated at step k. Solving the linear system yields the step $\Delta\mathbf{q}$, which is then used to update the current value of the state vector, $\mathbf{q}^{(k)}$, to a new one, $\mathbf{q}^{(k+1)}$. In the update, θ is a positive step size determined by line-search methods.

In this work, we use the flexible generalized minimal residual (FGMRES) method in conjunction with the cubic line-search option to solve Eq. (3.3), leveraging the PETSc library [4]. Each increment, $\Delta\mathbf{q}$, is solved iteratively until the tolerance determined by the Eisenstat–Walker algorithm is met [28]. The FGMRES method is a Krylov subspace method that minimizes the residual norm $||\mathbf{r}||_2$ with respect to $\Delta\mathbf{q}$ [117] where $\mathbf{r} = \mathbf{J}\Delta\mathbf{q} + \boldsymbol{\mathcal{R}}(\mathbf{q}^{(k)})$. The method requires an initial guess for the step, $\Delta\mathbf{q}_0$ which is discussed later in this section. We set the Krylov subspace size to $m = 30$ in our computations. The most computationally demanding steps of this process are those related to the matrix-vector products, $\mathbf{J}\mathbf{v}$. Instead of evaluating all the terms in the Jacobian, storing them and directly applying matrix vector product, we use a directional finite-different

approximation,

$$\mathbf{J}\mathbf{v} \approx \frac{\boldsymbol{\mathcal{R}}(\mathbf{q}^{(k)} + \epsilon\mathbf{v}) - \boldsymbol{\mathcal{R}}(\mathbf{q}^{k})}{\epsilon}. \tag{3.5}$$

This approach is more efficient both in terms of computational time and memory. The step size ϵ is determined by [19],

$$\epsilon = \begin{cases} e_{\text{rel}}\mathbf{v}^\top\mathbf{q}^{(k)}/\|\mathbf{v}\|_2^2 & \text{if } \left|\mathbf{v}^\top\mathbf{q}^{(k)}\right| > u_{\min}\|\mathbf{v}\|_1 \\ e_{\text{rel}}u_{\min}\,\text{sign}\left(\mathbf{v}^\top\mathbf{q}^{(k)}\right)\|\mathbf{v}\|_1/\|\mathbf{v}\|_2^2 & \text{otherwise} \end{cases}, \tag{3.6}$$

where $u_{\min}$ and e_{rel} are set to 10^{-6} and 10^{-8}, respectively. This Jacobian-free Newton–Krylov method is detailed by Knoll and Keyes [67].

Solving the time-spectral aeroelastic equations has now been reduced to a sequence of residual evaluations. This residual evaluation is described in Algorithm 1. This algorithm is an extension of the steady-state aeroelastic solver developed by Kenway et al. [62].

Algorithm 1 Coupled nonlinear residual computation

function $\boldsymbol{\mathcal{R}}(V_f, \omega_f, \boldsymbol{\eta}^n, \boldsymbol{\zeta}^n)$
  $\mathbf{X}_S^n \leftarrow \boldsymbol{\mathcal{G}}(\boldsymbol{\eta}^n)$ ▷ Transfer displacements
  $\mathbf{X}_V^n \leftarrow \boldsymbol{\mathcal{W}}\left(\mathbf{X}_S^n\right)$ ▷ Deform volume mesh to match surface
  $\boldsymbol{\mathcal{A}}_{\text{TS}} \leftarrow \boldsymbol{\mathcal{A}}_{\text{TS}}\left(V_f, \omega_f, \boldsymbol{\zeta}^n, \mathbf{X}_V^n\right)$ ▷ Evaluate CFD residuals
  $\mathbf{f}_{\boldsymbol{\mathcal{A}}}^n \leftarrow \mathbf{f}_{\boldsymbol{\mathcal{A}}}^n\left(\boldsymbol{\zeta}^n, \mathbf{X}_S^n\right)$ ▷ Evaluate aerodynamic forces
  $\bar{\mathbf{f}}^n \leftarrow \boldsymbol{\mathcal{G}}'(\mathbf{f}_{\boldsymbol{\mathcal{A}}}^n)$ ▷ Transfer forces
  $\boldsymbol{\mathcal{S}}_{\text{TS}} \leftarrow \boldsymbol{\mathcal{S}}_{\text{TS}}(V_f, \omega_f, \boldsymbol{\eta}^n, \bar{\mathbf{f}}^n)$ ▷ Evaluate CSD residuals
  $\boldsymbol{\mathcal{R}}_m \leftarrow \boldsymbol{\mathcal{R}}_m(\boldsymbol{\eta}^n)$ ▷ Evaluate prescribed motion magnitude residual
  $\boldsymbol{\mathcal{R}}_p \leftarrow \boldsymbol{\mathcal{R}}_p(\boldsymbol{\eta}^n)$ ▷ Evaluate prescribed motion phase residual
  $\boldsymbol{\mathcal{R}} \leftarrow (\boldsymbol{\mathcal{R}}_m, \boldsymbol{\mathcal{R}}_p, \boldsymbol{\mathcal{S}}_{\text{TS}}, \boldsymbol{\mathcal{A}}_{\text{TS}})$ ▷ Combine residuals
  return $\boldsymbol{\mathcal{R}}$
end function

For a Newton-type method to converge, the initial guess needs to be close to the solution.

We use a previous solution $(V_{\text{f, prev}}, \omega_{\text{prev}}, \boldsymbol{\eta}^n_{\text{prev}}, \boldsymbol{\zeta}^n_{\text{prev}})$, at M_{prev}, as the initial guess for a new Mach number, M_{curr}.

This strategy assumes that the states at M_{curr} and M_{prev} are close. To further improve the initial solution quality, we perform the following two additional steps. First, we conduct one complete CFD analysis using $(V_{\text{f, prev}}, \omega_{\text{prev}}, \boldsymbol{\eta}^n_{\text{prev}})$ at M_{curr}, which gives $\boldsymbol{\zeta}^n_{\text{inter}}$. Then, we start the CNK solution at the new Mach number, M_{curr}, using $(V_{\text{f, prev}}, \omega_{\text{prev}}, \boldsymbol{\eta}^n_{\text{prev}}, \boldsymbol{\zeta}^n_{\text{inter}})$ as the initial guess. This approach is used in the application sections whenever a warm start is used. Other strategies exist in literature to obtain an initial guess. Thomas et al. [137] proposed using the linear flutter solution found by a time-linearized aerodynamic analysis as an initial point for the Newton solver. The method was shown to be effective, requiring only a few iterations to achieve convergence.

3.3 Preconditioner

When solving Eq. (3.3), one critical requirement for good performance of an iterative method is that the eigenvalues of $\mathbf{J}$ be close to each other. To guarantee this, we need to construct a suitable preconditioner. Similar to the previous steady aerostructural work, we implement a right block-Jacobi preconditioner $\mathbf{P}$ such that

$$\begin{aligned} (\mathbf{J}\mathbf{P}^{-1})\Delta\mathbf{y} &= -\boldsymbol{\mathcal{R}}(\mathbf{q}^{(k)}), \\ \mathbf{P}^{-1}\Delta\mathbf{y} &= \Delta\mathbf{q}. \end{aligned} \tag{3.7}$$

The second equation is expanded as

$$\begin{bmatrix} \mathbf{P}^{-1}_{\text{motion,CSD}} & 0 \\ 0 & \mathbf{P}^{-1}_{\text{CFD}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}_{V_f,\omega,\text{CSD}} \\ \Delta\mathbf{y}_{\text{CFD}} \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{q}_{V_f,\omega,\text{CSD}} \\ \Delta\mathbf{q}_{\text{CFD}} \end{bmatrix}. \tag{3.8}$$

The preconditioner already implemented for the steady CFD solver $\mathbf{P}^{-1}_{\text{CFD}}$ was reused [62]. The

preconditioner uses an additive Schwartz method with an overlap of 1 (ASM(1)) at the top level with an incomplete lower-upper decomposition using two levels of fill (ILU(2)) for the local block preconditioners. The preconditioner is based on a first-order discretization to reduce its bandwidth. For the time-spectral CFD preconditioner, the coupling between different time instances is ignored, resulting in n blocks of such preconditioners on the diagonal. For more discussion about the block diagonal preconditioner and the time-spectral preconditioner, we refer the reader to the aerostructural preconditioner demonstrated by Kenway et al. [62], and the time-spectral preconditioner demonstrated by Mader and Martins [90].

The preconditioner $\mathbf{P}_{\text{CSD}}$ for the motion equations and the CSD equations are new developments implemented in this work. Three preconditioning strategies are developed here and their numerical performance is compared in Section 6.1.4.

3.3.1 Direct-inversion preconditioner

Due to the relatively small problem size resulting from the airfoil problem considered in this work, a direct factorization is reasonable. A direct inversion of the Jacobian is used as the preconditioner for the motion equations and the CSD equations. The resulting preconditioner is:

$$\mathbf{P}^{-1}_{\text{motion, CSD}} = \begin{bmatrix} 0 & 0 & \frac{\partial |\bar{\eta}_{\text{1st harmonic},j}|}{\partial \boldsymbol{\eta}^n} \\ 0 & 0 & \frac{\partial \theta_{\text{1st harmonic},j}}{\partial \boldsymbol{\eta}^n} \\ -\left(\frac{S_{\text{ref}} b}{V_0}\right) V_f \left(\bar{\mathbf{f}}_r^n\right) & \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{2\omega}{\omega_\alpha^2}\right)\left(\mathbf{Q}^\intercal \bar{\mathbf{D}}^2 \mathbf{Q}\right)\left(\bar{\boldsymbol{\eta}}^n\right) & \left(\left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{Q}^\intercal \bar{\mathbf{D}}^2 \mathbf{Q}\right) + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\right) \end{bmatrix}^{-1}. \tag{3.9}$$

Unless otherwise stated, all results presented in this work use the direct-inversion preconditioner.

3.3.2 Schur complement based preconditioner

With a more involved case and a larger structure, such as a full finite element method (FEM) wing box model, the direct inversion of Eq. (3.9) may not be possible and a more suitable preconditioner strategy should be found. However, for a FEM solver with time-spectral capability, an existing routine could be used to invert the diagonal component $\mathbf{M}^n\mathbf{D_Q} + \mathbf{K}^n$. Based on that, we propose to form a preconditioner for $\mathbf{P}^{-1}_{\text{motion,CSD}}$ by applying Schur complement. We expand the equation for the structure and motion preconditioner,

$$\begin{aligned} \mathbf{P}_{\text{CSD}}\Delta\mathbf{q}_{V_f,\omega,\text{CSD}} &= \Delta\mathbf{y}_{V_f,\omega,\text{CSD}}, \\ \begin{bmatrix} \mathbf{0} & \mathbf{B}_2^\top \\ \mathbf{B}_1 & \mathbf{A} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{q}_1 \\ \Delta\mathbf{q}_2 \end{bmatrix} &= \begin{bmatrix} \Delta\mathbf{y}_1 \\ \Delta\mathbf{y}_2 \end{bmatrix}, \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} \mathbf{A} &= \mathbf{M}^n\mathbf{D_Q} + \mathbf{K}^n, \\ \mathbf{B}_1 &= \begin{bmatrix} -\frac{2V_f}{\pi}\mathbf{f}^n & \mathbf{M}^n\frac{d\mathbf{D_Q}}{d\omega}\mathbf{u}^n \end{bmatrix}, \\ \mathbf{B}_2 &= \begin{bmatrix} \frac{\partial|\alpha_1|}{\partial\mathbf{u}^n} & \frac{\partial\phi}{\partial\mathbf{u}^n} \end{bmatrix}. \end{aligned} \tag{3.11}$$

The solution for the updates is

$$\begin{bmatrix} \Delta\mathbf{q}_1 \\ \Delta\mathbf{q}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{S}_\mathbf{A}^{-1} & \mathbf{S}_\mathbf{A}^{-1}(-\mathbf{B}_2^\top\mathbf{A}^{-1}) \\ -\mathbf{A}^{-1}\mathbf{B}_1\mathbf{S}_\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}_1\mathbf{S}_\mathbf{A}^{-1}(-\mathbf{B}_2^\top\mathbf{A}^{-1}) + \mathbf{A}^{-1} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}_1 \\ \Delta\mathbf{y}_2 \end{bmatrix}, \tag{3.12}$$

where $\mathbf{S}_\mathbf{A} = -\mathbf{B}_2^\top\mathbf{A}^{-1}\mathbf{B}_1$ is the Schur complement of $\mathbf{A}$ in $\mathbf{P}_{\text{CSD}}$. In this way, we leverage on the existing solver for the inversion operation of $\mathbf{A}$. In a time-spectral FEM solver, if a direct solver is used, as with TACS [59], then $\mathbf{A}^{-1}$ is the direct factorization of the matrix. Alternatively, if an iterative solver is applied, $\mathbf{A}^{-1}$ could be approximated by the preconditioner used in the FEM solver.

3.3.3 Saddle point system preconditioner

Instead of computing all the terms in the matrix inversion 3.12, we can ignore secondary terms to make the computation more efficient. The SPS preconditioner is given by

$$\tilde{\mathbf{P}}_{\text{motion, CSD}} = \begin{bmatrix} \mathbf{S_A} & \\ & \mathbf{A} \end{bmatrix}. \tag{3.13}$$

where the off-diagonal terms and $-\mathbf{A}^{-1}\mathbf{B}_1\mathbf{S}_\mathbf{A}^{-1}(-\mathbf{B}_2^\top\mathbf{A}^{-1})$ from the 2nd diagonal term in Eq. (3.12) are all ignored. This makes it easier to compute the preconditioner, but makes the preconditioner less effective in improving the original linear system conditioning number. We evaluate the performance of the different preconditioners in Section 6.1.4.

3.3.4 Diagonal correction

For the airfoil test case, we find that the aforementioned preconditioners can be used as it is and have good performance in general. However, for the wing test case, we test with the direct-inversion preconditioner and the solver stalls at around 10^{-4} time-spectral aeroelastic residual. We find that the cause of the issue is that the first two rows from $\mathbf{P}_{\text{CSD}}$ is not diagonally dominant. We fix the issue by adding a diagonal correction, ϵ_{corr} to the two rows where $\epsilon_{\text{corr}} \in (10^{-3}, 10^{-5})$.

$$\mathbf{P}_{\text{motion, CSD, corr}} = \mathbf{P}_{\text{motion, CSD}} + \begin{bmatrix} \epsilon_{\text{corr}} & 0 & 0 \\ 0 & \epsilon_{\text{corr}} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.14}$$

This seemingly small change has a significant impact on the solution convergence. With the correction, the residual is finally driven to 10^{-10}. Later in Chapter 4, the transpose of the current preconditioners are used for the adjoint equation solutions. The correction has a similar impact on the adjoint equation convergence. However, the correction term seems does not have much effect

on the airfoil case. This is remained to be investigated.

CHAPTER 4

Time-Spectral Aeroealstic ADjoint and Krylov Subspace Solver

In this chapter, we present the coupled adjoint method developed for efficient derivative computation. This chapter is organized as follows: In Section 4.1, we present the coupled adjoint equation. Then in Section 4.2, we show the implementation detail of the coupled adjoint equation solution method. For the implementation, the Krylov subspace method is used to solve the linear coupled adjoint equations and AD is used to evaluate the matrix-vector product required by the Krylov subspace method. In Section 4.2, we present the related matrix-vector products. Finally, the coupled adjoint solution method is shown in Section 4.3. The preconditioning method which is critical for the convergence of a Krylov subspace method is shown in Section 4.3 as well.

4.1 Coupled adjoint overview

The function of interest (e.g. V_f) derivatives with respect to design variables are important design information. We apply the adjoint method to evaluate the sensitivity. For a more general treatment for the adjoint method, we refer the readers to Martins and Hwang [95]. The total

derivatives of the function of interest with respect to the design variables are given as follows

$$\begin{aligned}\frac{\mathrm{d}I}{\mathrm{d}\mathbf{x}} &= \frac{\partial I}{\partial \mathbf{x}} + \frac{\partial I}{\partial \mathbf{q}}\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\mathbf{x}}, \\ &= \frac{\partial I}{\partial \mathbf{x}} + \begin{bmatrix} \frac{\partial I}{\partial V_f} & \frac{\partial I}{\partial \omega} & \frac{\partial I}{\partial \boldsymbol{\eta}^n} & \frac{\partial I}{\partial \boldsymbol{\zeta}^n} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}V_f}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\omega}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\eta}^n}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\zeta}^n}{\mathrm{d}\mathbf{x}} \end{bmatrix},\end{aligned} \tag{4.1}$$

where set I as the function of interest, $\mathbf{x}$ as the design variables, and the state variable $\mathbf{q}$ is defined in Eq. (3.2). The total derivatives of the state variables $\mathrm{d}\mathbf{q}/\,\mathrm{d}\mathbf{x}$ with respect to the design variables from Eq. (4.1) also satisfies the following equation

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{\mathcal{R}}}{\mathrm{d}\mathbf{x}} &= \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{x}} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \mathbf{q}}\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\mathbf{x}} = 0, \\ \begin{bmatrix} \frac{\mathrm{d}R_m}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}R_p}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\mathrm{d}\mathbf{x}} \end{bmatrix} &= \begin{bmatrix} \frac{\partial R_m}{\partial \mathbf{x}} \\ \frac{\partial R_p}{\partial \mathbf{x}} \\ \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \mathbf{x}} \\ \frac{\partial \boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\partial \mathbf{x}} \end{bmatrix} + \begin{bmatrix} \frac{\partial R_m}{\partial V_f} & \frac{\partial R_m}{\partial \omega} & \frac{\partial R_m}{\partial \boldsymbol{\eta}^n} & \frac{\partial R_m}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial R_p}{\partial V_f} & \frac{\partial R_p}{\partial \omega} & \frac{\partial R_p}{\partial \boldsymbol{\eta}^n} & \frac{\partial R_p}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial V_f} & \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \omega} & \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \boldsymbol{\eta}^n} & \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial \boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\partial V_f} & \frac{\partial \boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\partial \omega} & \frac{\partial \boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\partial \boldsymbol{\eta}^n} & \frac{\partial \boldsymbol{\mathcal{A}}_{\mathrm{TS}}}{\partial \boldsymbol{\zeta}^n} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}V_f}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\omega}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\eta}^n}{\mathrm{d}\mathbf{x}} \\ \frac{\mathrm{d}\boldsymbol{\zeta}^n}{\mathrm{d}\mathbf{x}} \end{bmatrix} = 0.\end{aligned} \tag{4.2}$$

This is based on the fact that no matter what values we set for the design variables, the residual should always be zero for a physical solution. Combining Eq. (4.1) and Eq. (4.2), we obtain the

following equation

$$\frac{\mathrm{d}I}{\mathrm{d}\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} - \boldsymbol{\Psi}^{\intercal} \begin{bmatrix} \frac{\partial R_m}{\partial \mathbf{x}} \\ \frac{\partial R_p}{\partial \mathbf{x}} \\ \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \mathbf{x}} \\ \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \mathbf{x}} \end{bmatrix},$$
$$\begin{bmatrix} \frac{\partial R_m}{\partial V_f} & \frac{\partial R_m}{\partial \omega} & \frac{\partial R_m}{\partial \boldsymbol{\eta}^n} & \frac{\partial R_m}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial R_p}{\partial V_f} & \frac{\partial R_p}{\partial \omega} & \frac{\partial R_p}{\partial \boldsymbol{\eta}^n} & \frac{\partial R_p}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial V_f} & \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \omega} & \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \boldsymbol{\eta}^n} & \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \boldsymbol{\zeta}^n} \\ \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial V_f} & \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \omega} & \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \boldsymbol{\eta}^n} & \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \boldsymbol{\zeta}^n} \end{bmatrix}^{\intercal} \boldsymbol{\Psi} = \begin{bmatrix} \frac{\partial I}{\partial V_f} \\ \frac{\partial I}{\partial \omega} \\ \frac{\partial I}{\partial \boldsymbol{\eta}^n} \\ \frac{\partial I}{\partial \boldsymbol{\zeta}^n} \end{bmatrix}, \tag{4.3}$$

where $\boldsymbol{\Psi}$ is the adjoint variables. The second equation from Eq. (4.3) is the so-called time-spectral aeroelastic adjoint equation. The adjoint method has the advantage that the number of linear solutions required to get the total derivatives scales with the dimension of the function of interest I rather than the dimension of the design variables. This is an advantage in aerodynamic shape design problems, since we typically have few functions of interest but hundreds of design variables. If there are more functions of interest than design variables, we should do the opposite: associating $(\partial \boldsymbol{\mathcal{R}}/\partial q)^{-1}$ with $\partial \boldsymbol{\mathcal{R}}/\partial \mathbf{x}$. If V_f is chosen for I, then we have

$$\frac{\partial I}{\partial \mathbf{x}} = 0,$$
$$\frac{\partial I}{\partial \mathbf{q}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.4}$$

which simplifies our computation a bit.

4.2 Coupled adjoint implementation

Equation (4.3) says nothing about the solution methodology. There are many ways to solve for the adjoint equation. The linear GS method proposed by Kenway et al. [62], Martins et al. [98], Shi et al. [121], the coupled Krylov adjoint solver by Kenway et al. [62], Shi et al. [121], the Monte Carlo (MC) method by Wang et al. [142] which is developed mainly for unsteady adjoint though, and more recently by dynamic mode decomposition method [21]. For the current work, we apply a coupled Krylov method, since it has been demonstrated by Kenway et al. [62] to be computationally more efficient than the linear GS method.

One key component for the coupled Krylov is the matrix-vector products between the transpose of the Jacobian matrix with certain seeds. To compute this accurately and efficiently, we apply the reverse AD method, which is precise up to machine precision, following Mader et al. [91].

The coupled adjoint and function sensitivity evaluation involves four components as shown in Eqs. (4.1) and (4.2). $(\partial \boldsymbol{\mathcal{R}}/\partial q)^\intercal \Psi$, $(\partial \boldsymbol{\mathcal{R}}/\partial \mathbf{x})^\intercal \Psi$, $\partial I/\partial q$ and $\partial I/\partial \mathbf{x}$. In this work, $\partial I/\partial q$ is a simple constant vector and $\partial I/\partial \mathbf{x}$ is simply zero as mentioned in Eq. (4.4). We focus on the other two components in this section.

4.2.1 Prescribed motion residual partial derivatives, $\partial R_m/\partial \mathbf{q}, \partial R_p/\partial \mathbf{q}$

The prescribed motion residual is solely dependently on displacements $\boldsymbol{\eta}$. Since the dimension of $\partial R_m/\partial \boldsymbol{\eta}^n, \partial R_p/\partial \boldsymbol{\eta}^n$ are quite small, with a dimension of $n \times N_{CSD}$ for either of them, instead of giving the matrix-vector product form, we simply store the matrix explicitly. And the matrix entries are evaluated using the finite difference method.

4.2.2 Structural residual partial derivatives, $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \mathbf{x}$

The expression for CSD equations is defined earlier in Eq. (2.27) and we write it out again here in Eq. (4.5)

$$\boldsymbol{\mathcal{S}}_{\text{TS}} = \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{P}^\intercal \bar{\mathbf{D}}^2 \mathbf{P}\right)\bar{\boldsymbol{\eta}}^n + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\bar{\boldsymbol{\eta}}^n - \frac{1}{2}\left(\frac{S_{\text{ref}} b}{V_0}\right) V_f^2 \bar{\mathbf{f}}_r^n = 0. \tag{4.5}$$

The partial derivatives of structural residual with respect to design variables are given as

$$\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \mathbf{x}}\right)^\intercal \boldsymbol{\phi} = \left(\frac{\partial \mathbf{X}_{S,0}}{\partial \mathbf{x}}\right)^\intercal \left(\frac{\partial \mathbf{X}_S^n}{\partial \mathbf{X}_{S,0}}\right)^\intercal \left(\frac{\partial \mathbf{f}^n}{\partial \mathbf{X}_S^n}\right)^\intercal \left(\frac{\partial \bar{\mathbf{f}}_r^n}{\partial \mathbf{f}^n}\right)^\intercal \left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \bar{\mathbf{f}}_r^n}\right)^\intercal \boldsymbol{\phi}, \tag{4.6}$$

where $\boldsymbol{\phi}$ is an arbitrary seed for the structural residual, $\mathbf{X}_{S,0}$ is the jig shape aerodynamic surface mesh coordinate, $\mathbf{X}_S^n$ is deformed aerodynamic surface mesh coordinate from all time instances, $\mathbf{f}^n$ are aerodynamic load at each surface mesh point, and $\bar{\mathbf{f}}_r^n$ is the generalized dimensionless aerodynamic load. The design variables affect the structure residual only through the aerodynamic load. This is due to the assumption that the structure remains unchanged during optimization. However, for aerostructural optimization problems, there are additional terms such as $\partial \mathbf{M}_r^n/\partial \mathbf{x}$ to be taken into account.

4.2.3 Structural residual partial derivatives, $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \mathbf{q}$

In this section, we present the partial derivative $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \mathbf{q}$. Because $\mathbf{q} = (V_f, \omega, \boldsymbol{\eta}^n, \boldsymbol{\zeta}^n)$, there are four terms to be derived: (1). $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial V_f$, (2). $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \omega$, (3). $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \boldsymbol{\eta}^n$, and (4). $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \boldsymbol{\zeta}^n$.

The partial derivatives of structural residual with respect to V_f are given as:

$$\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial V_f}\right)^\intercal \boldsymbol{\phi} = -\left(\frac{S_{\text{ref}} b}{V_0}\right) V_f \left(\bar{\mathbf{f}}_r^n\right)^\intercal \boldsymbol{\phi}. \tag{4.7}$$

The partial derivatives of structural residual with respect to ω are given as:

$$\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \omega}\right)^{\intercal} \boldsymbol{\phi} = \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{2\omega}{\omega_\alpha^2}\right)\left(\mathbf{Q}^{\intercal}\bar{\mathbf{D}}^2\mathbf{Q}\right)\left(\bar{\boldsymbol{\eta}}^n\right)^{\intercal}\boldsymbol{\phi}. \tag{4.8}$$

The partial derivatives of structural residual with respect to structural state variables are more complicated. It is composed of two ingredients: (1). Directly through $\boldsymbol{\eta}$ shown in Eq. (4.5), and (2). Indirectly through the generalized aerodynamic loads. The result is shown below

$$\begin{aligned}
&\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \boldsymbol{\eta}^n}\right)^{\intercal}\boldsymbol{\phi} \\
&= \underbrace{\left(\left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{Q}^{\intercal}\bar{\mathbf{D}}^2\mathbf{Q}\right) + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\right)^{\intercal}\boldsymbol{\phi}}_{(1)} + \underbrace{\left(\frac{\partial \mathbf{X}_S^n}{\partial \boldsymbol{\eta}^n}\right)^{\intercal}\left(\frac{\partial \mathbf{f}^n}{\partial \boldsymbol{X}_S^n}\right)^{\intercal}\left(\frac{\partial \bar{\mathbf{f}}_r^n}{\partial \mathbf{f}^n}\right)^{\intercal}\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \bar{\mathbf{f}}_r^n}\right)^{\intercal}\boldsymbol{\phi}}_{(2)}.
\end{aligned} \tag{4.9}$$

Notice that $\partial \mathbf{X}_S^n/\partial \boldsymbol{\eta}^n$, $\partial \mathbf{f}^n/\partial \boldsymbol{X}_S^n$, $\partial \bar{\mathbf{f}}_r^n/\partial \mathbf{f}^n$, and $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \bar{\mathbf{f}}_r^n$, are all block diagonal matrices.

Finally, the partial derivatives of structural residual with respect to aerodynamic state variables are given as

$$\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \boldsymbol{\zeta}^n}\right)^{\intercal}\boldsymbol{\phi} = \left(\frac{\partial \bar{\mathbf{f}}_r^n}{\partial \boldsymbol{\zeta}^n}\right)^{\intercal}\left(\frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \bar{\mathbf{f}}_r^n}\right)^{\intercal}\boldsymbol{\phi}. \tag{4.10}$$

Similar with $\partial \mathbf{X}_S^n/\partial \boldsymbol{\eta}^n$ and $\partial \boldsymbol{\mathcal{S}}_{\text{TS}}/\partial \bar{\mathbf{f}}_r^n$, $\partial \bar{\mathbf{f}}_r^n/\partial \boldsymbol{\zeta}^n$ is also block diagonal because the aerodynamic state variables only affect generalized aerodynamic load from the same time instance.

4.2.4 Aerodynamic residual partial derivatives, $\partial \boldsymbol{\mathcal{A}}_{\text{TS}}/\partial \mathbf{x}$

The partial derivative of aerodynamic residual with respect to the design variables multiplied with an aerodynamic residual seed is expanded as

$$\left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \boldsymbol{\psi} = \left(\frac{\partial \mathbf{X}_{S,0}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \left(\frac{\partial \mathbf{X}_S^n}{\partial \mathbf{X}_{S,0}}\right)^{\mathsf{T}} \left(\frac{\partial \mathbf{X}_V^n}{\partial \mathbf{X}_S^n}\right)^{\mathsf{T}} \left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \mathbf{X}_V^n}\right)^{\mathsf{T}} \boldsymbol{\psi}, \tag{4.11}$$

where $\mathbf{X}_V^n$ represents the deformed volume coordinates for n time instances, and $\boldsymbol{\psi}$ represents an arbitrary vector encountered when we use a Krylov subspace solver. Since we use RAD for the matrix-vector product, none of the matrices is formed explicitly here. The product $(\partial {\boldsymbol{\mathcal{A}}_{\text{TS}}}^n/\partial \mathbf{x}_V^n)^{\mathsf{T}} \boldsymbol{\psi}$ is computed using ADflow by Kenway et al. [63], Mader and Martins [90]. The product with $(\partial \mathbf{X}_V^n/\partial \mathbf{X}_S^n)^{\mathsf{T}}$ is computed using IDWarp [119]. The original mesh deformation method is proposed by Luke et al. [83] which scales with $\boldsymbol{\mathcal{O}}(N \log(N))$ where N is the number of 3D elements. Notice that $(\partial {\boldsymbol{\mathcal{A}}_{\text{TS}}}^n/\partial \mathbf{x}_V^n)^{\mathsf{T}}$ is coupled for different time instances. On the contrary, $(\partial \mathbf{X}_V^n/\partial \mathbf{x}_S^n)^{\mathsf{T}}$ is decoupled for different time instances. The product with $(\partial \mathbf{X}_S^n/\partial \mathbf{X}_{S,0})^{\mathsf{T}}$ is newly implemented in the transfer class. The product with $\partial \mathbf{X}_{S,0}/\partial \mathbf{x}$ is evaluated using pyGeo by Kenway et al. [64].

4.2.5 Aerodynamic residual partial derivatives, $\partial \boldsymbol{\mathcal{A}}_{\text{TS}}/\partial \mathbf{q}$

We covered the matrix-vector multiplication between aerodynamic residual partial derivatives with respect to design variables in the previous section. Now we switched to consider the partial derivatives with respect to the state variables.

The LCO speed index affects the aerodynamic residual by perturbing the boundary conditions as discussed Section 2.3.5. The matrix-vector product between the aerodynamic residual partial derivative with respect to the LCO speed index and an aerodynamic seed is given as

$$\left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial V_f}\right)^{\mathsf{T}} \boldsymbol{\psi} = \left(\frac{dT_\infty}{dV_f}\right)^{\mathsf{T}} \left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial T_\infty}\right)^{\mathsf{T}} \boldsymbol{\psi} + \left(\frac{dp_\infty}{dV_f}\right)^{\mathsf{T}} \left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial p_\infty}\right)^{\mathsf{T}} \boldsymbol{\psi} \tag{4.12}$$

where T_∞, p_∞ are the boundary temperature and pressure, respectively.

The partials for angular velocity ω is computed directly by

$$\left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \omega}\right)^{\mathsf{T}} \boldsymbol{\psi}. \tag{4.13}$$

With spectral interpolated grid velocity [49], the grid velocity will be dependent on the time period.

The matrix-vector multiplication between aerodynamic residual partial derivatives with respect to structural displacement $\boldsymbol{\eta}^n$ and aerodynamic residual seeds is given as

$$\left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \boldsymbol{\eta}^n}\right)^{\mathsf{T}} \boldsymbol{\psi} = \left(\frac{\partial \mathbf{X}_S^n}{\partial \boldsymbol{\eta}^n}\right)^{\mathsf{T}} \left(\frac{\partial \mathbf{X}_V^n}{\partial \mathbf{X}_S^n}\right)^{\mathsf{T}} \left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \mathbf{X}_V^n}\right)^{\mathsf{T}} \boldsymbol{\psi}, \tag{4.14}$$

where the term $(\partial \mathbf{X}_S^n / \partial \boldsymbol{\eta}^n)^{\mathsf{T}}$ is related to how the structural displacement impact the surface coordinates, and other terms were discussed in Eq. (4.11). Similar with $(\partial \mathbf{X}_V^n / \partial \mathbf{X}_S^n)^{\mathsf{T}}$, $(\partial \mathbf{X}_S^n / \partial \boldsymbol{\eta}^n)^{\mathsf{T}}$ is decoupled between time instances. Finally, the $(\partial \boldsymbol{\mathcal{A}}_{\text{TS}} / \partial \boldsymbol{\eta}^n)^{\mathsf{T}}$ is dense – each displacement is likely to affect all the aerodynamic residuals. The displacement affects the aerodynamic residual within its own time instance. It affects other time instances by affecting the spectral interpolated grid velocity.

Finally, the partial derivative of aerodynamic residual with respect to aerodynamic state variables

$$\left(\frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \boldsymbol{\zeta}^n}\right)^{\mathsf{T}} \boldsymbol{\psi}, \tag{4.15}$$

has already been developed and discussed in detail by Mader and Martins [90].

4.3 Coupled adjoint solution

4.3.1 Coupled Krylov solver

To solve the coupled adjoint equation, we apply the Krylov subspace method. Krylov subspace method has the advantage that it is not required to store the matrix explicitly and only the matrix-vector products are required for the solution. Since the matrix-vector products are between the transpose of Jacobian matrices and vectors, we apply RAD to this operation. The pseudocode for the operation is given in Algorithm 2.

Algorithm 2 Coupled Krylov method linear operator

1: **function** MULT($\mathbf{w}$)
2: $(\mathbf{w}_{\boldsymbol{\mathcal{R}}_m}, \mathbf{w}_{\boldsymbol{\mathcal{R}}_p}, \mathbf{w}_{\boldsymbol{\mathcal{S}}_{\text{TS}}}, \mathbf{w}_{\boldsymbol{\mathcal{A}}_{\text{TS}}}) \leftarrow \mathbf{w}$ ▷ Extract prescribed motion magnitude, prescribed motion phase, structural, and aerodynamic residual seeds
3: $\mathbf{w}_{\mathbf{f}^n} \leftarrow \frac{\partial \bar{\mathbf{f}}^n}{\partial \mathbf{f}^n}^\intercal \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \mathbf{f}^n}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{S}}_{\text{TS}}}$ ▷ Compute the aerodynamic load seeds
4: $\mathbf{w}_{\mathbf{X}_S^n} \leftarrow \frac{\partial \mathbf{X}_V^n}{\partial \mathbf{X}_S^n}^\intercal \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \mathbf{X}_V^n}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{A}}_{\text{TS}}} + \frac{\partial \mathbf{f}^n}{\partial \mathbf{X}_S^n}^\intercal \mathbf{w}_{\mathbf{f}^n}$ ▷ Compute the aerodynamic surface coordinates seeds
5: $\mathbf{y}_{V_f} \leftarrow \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial V_f}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{A}}_{\text{TS}}} + \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial V_f}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{S}}_{\text{TS}}}$ ▷ Compute the speed index seed with aerodynamic and structural contributions
6: $\mathbf{y}_{\omega} \leftarrow \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \omega}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{A}}_{\text{TS}}} + \frac{\partial \boldsymbol{\mathcal{S}}_{\text{TS}}}{\partial \omega}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{S}}_{\text{TS}}}$ ▷ Compute the frequency seed with aerodynamic and structural contributions
7: $\mathbf{y}_{\boldsymbol{\eta}^n} \leftarrow \left(\left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\omega^2}{\omega_\alpha^2}\right)\left(\mathbf{Q}^\intercal \bar{\mathbf{D}}^2 \mathbf{Q}\right) + \left(\frac{\mathbf{M}_r^n}{m_0}\right)\left(\frac{\boldsymbol{\Omega}^2}{\omega_\alpha^2}\right)\right)^\intercal \mathbf{w}_{\boldsymbol{\mathcal{S}}_{\text{TS}}} + \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\eta}^n}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{R}}_m} + \frac{\partial \boldsymbol{\mathcal{R}}_p}{\partial \boldsymbol{\eta}^n}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{R}}_p} + \frac{\partial \mathbf{X}_S^n}{\partial \boldsymbol{\eta}^n}^\intercal \mathbf{w}_{\mathbf{X}_S^n}$ ▷ Compute structural displacement seeds with prescribed motion, aerodynamic and structural contributions
8: $\mathbf{y}_{\boldsymbol{\zeta}^n} \leftarrow \frac{\partial \boldsymbol{\mathcal{A}}_{\text{TS}}}{\partial \boldsymbol{\zeta}^n}^\intercal \mathbf{w}_{\boldsymbol{\mathcal{A}}_{\text{TS}}} + \frac{\partial \mathbf{f}^n}{\partial \boldsymbol{\zeta}^n}^\intercal \mathbf{w}_{\mathbf{f}^n}$ ▷ Compute aerodynamic state variable seeds with structural and aerodynamic contributions
9: $\mathbf{y} \leftarrow (\boldsymbol{y}_{V_f}, \boldsymbol{y}_{\omega}, \mathbf{y}_{\boldsymbol{\eta}^n}, \mathbf{y}_{\boldsymbol{\zeta}^n})$ ▷ Combining the output seeds
10: **return** $\mathbf{y}$
11: **end function**

In Algorithm 2, we use the partial derivatives derived in the previous sections after merging similar terms. In Line 3 of Algorithm 2, we compute the $\mathbf{f}^n$ seed later used in Eq. (4.9) and Eq. (4.10). In Line 4, we compute the $\mathbf{X}_S^n$ seed needed for Eq. (4.14) and Eq. (4.9). In Line 5, we compute the V_f seed using Eq. (4.12) and Eq. (4.7). Similarly, in Line 6, we compute the ω seed using Eq. (4.13) and Eq. (4.8). In Line 7, we compute the $\boldsymbol{\zeta}^n$ seed using Eq. (4.9), $\mathbf{X}_S^n$ seed

computed previously in Line 4, and the prescribed motion seed described in Section 4.2.1. In Line 8, we compute $\boldsymbol{\zeta}^n$ seeds using Eq. (4.15) and $\mathbf{f}^n$ seed computed in Line 3.

To improve the convergence of the Krylov method, we apply a block Jacobi preconditioner. The reason we choose block Jacobi preconditioner is that it will allow the structural and aerodynamic preconditioning to be carried out in parallel and it allows the reuse of the time-spectral aerodynamic preconditioner developed in [90]. The preconditioner is defined by

$$\begin{aligned} \left(\mathbf{J}^\intercal \mathbf{P}^{-\intercal}\right) \boldsymbol{\tau} &= \frac{\partial I}{\partial \mathbf{q}}, \\ \mathbf{P}^{-\intercal} \boldsymbol{\tau} &= \boldsymbol{\Psi}, \end{aligned} \tag{4.16}$$

where $\mathbf{P}^\intercal$ is the preconditioner, $\boldsymbol{\tau}$ is the solution of the preconditioned system. To be more specific, the second equation is expanded as

$$\begin{bmatrix} \mathbf{P}^{-\intercal}_{\text{motion, CSD}} & 0 \\ 0 & \mathbf{P}^{-\intercal}_{\text{CFD}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_{\text{motion, CSD}} \\ \boldsymbol{\tau}_{\text{CFD}} \end{bmatrix} = \begin{bmatrix} \Psi_{\boldsymbol{\mathcal{R}}_m, \boldsymbol{\mathcal{R}}_p, \boldsymbol{\mathcal{S}}_{\text{TS}}} \\ \Psi_{\boldsymbol{\mathcal{A}}_{\text{TS}}} \end{bmatrix}. \tag{4.17}$$

where $\mathbf{P}^{-1}_{\text{mot, }\boldsymbol{\mathcal{S}}_{\text{TS}}}$ is the Jacobian of the prescribed motion residual and structural residual with respect to LCO speed index, frequency and structural displacements, $\boldsymbol{\tau}_{\text{motion, CSD}}$ and $\boldsymbol{\tau}_{\text{CFD}}$ are components from $\boldsymbol{\tau}$, and $\Psi_{\boldsymbol{\mathcal{R}}_m, \boldsymbol{\mathcal{R}}_p, \boldsymbol{\mathcal{S}}_{\text{TS}}}$ and $\Psi_{\boldsymbol{\mathcal{A}}_{\text{TS}}}$ are components from $\boldsymbol{\Psi}$. As mentioned before, the CFD preconditioner $\mathbf{P}^{-\intercal}_{\text{CFD}}$ has been implemented previously by Mader and Martins [90]. For the prescribed motion and CSD preconditioner, we use the transportation of the preconditioning matrices from Section 3.3.

Similar to the preconditioner for our Newton–Krylov solver, the diagonal correction Section 3.3.4 is critical for the convergence of the adjoint equation residuals. Without the correction, the adjoint equation residual stalls at about 10^{-4}. With the correction, the adjoint equation residual can be reduced to about 10^{-10}.

CHAPTER 5

Derivatives for Eigenvalues and Eigenvectors for Analytic RAD

In this chapter, we present methods to efficiently compute the eigenvalue and eigenvector derivatives. Currently, we make the assumption that during the aerodynamic shape optimization (see Chapter 7), the structural model is unchanged. In reality, the derivative of the structural natural frequencies and mode shapes with respect to the design variables needs to be taken into account. We develop effcient RAD formulae for this purpose. This chapter is organized as follows: In Section 5.1, we define the generalized eigenvalue problem encountered in structural mechanics. In Section 5.2, we present the background. We discuss how the proposed formulae in this chapter are related with the LCO speed optimization. In Section 5.3, we present our new formulae for the derivative computation. In Section 5.4, we give some recommendations on how to implement the formulae efficiently both in speed and memory. Later, in Section 5.5, we verify the proposed formulae using an Euler–Bernoulli beam test case. Finally, we summarize the research in Section 5.6. For a brief review of the AD technique and the conventions of symbols used in this chapter e.g. $\bar{\cdot}$ and $\dot{\cdot}$, we refer the readers to Section 2.5.

5.1 Generalized eigenvalue problem

The generalized eigenvalue problem in the case of a structural problem with a full basis can be written as,

$$\begin{cases} \mathbf{M}\boldsymbol{\Phi} - \mathbf{K}\boldsymbol{\Phi}\boldsymbol{\Lambda} = 0 \\ \\ \boldsymbol{\Phi}^{\intercal}\mathbf{M}\boldsymbol{\Phi} - \mathbf{I} = 0 \end{cases}, \tag{5.1}$$

where $\mathbf{M}, \mathbf{K} \in \mathbb{R}^{n\times n}$ are the mass and stiffness matrices, which are symmetric, $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with n eigenvalues occupying the diagonal, $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$ is a matrix of n eigenvectors corresponding with the n eigenvalues from $\boldsymbol{\Lambda}$, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix. We assume that there is no repeated eigenvalues, i.e., $\boldsymbol{\Lambda}_{ii} \neq \boldsymbol{\Lambda}_{jj},\ \forall i \neq j$. The mass and stiffness matrices are dependent on the design variables $\mathbf{x}$, but we omit them here to simplify notation. For structural applications, we are interested in the lowest natural frequencies and the associated mode shapes. In this case, the lowest natural frequency corresponds to the dominant eigenvalue. The i^{th} dominant eigenvalue, $\boldsymbol{\Lambda}_{ii}$, is related with the i^{th} natural frequency, ω_i, by

$$\boldsymbol{\Lambda}_{ii} = \frac{1}{\omega_i^2}. \tag{5.2}$$

In many practical applications, such as structural and aeroelastic optimization, only a handful of eigenvalues and the associated eigenvectors are needed and therefore $r \ll n$. The full basis eigenvalue and eigenvectors, $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$, respectively, can be written as,

$$\begin{aligned} \boldsymbol{\Phi} &= \begin{bmatrix} \hat{\boldsymbol{\Phi}} & \tilde{\boldsymbol{\Phi}} \end{bmatrix}, \\ \boldsymbol{\Lambda} &= \begin{bmatrix} \hat{\boldsymbol{\Lambda}} & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\Lambda}} \end{bmatrix}, \end{aligned} \tag{5.3}$$

where $\hat{\boldsymbol{\Lambda}} \in \mathbb{R}^{r\times r}$ and $\hat{\boldsymbol{\Phi}} \in \mathbb{R}^{n\times r}$ are the reduced eigenvalues and eigenvectors, and $\tilde{\boldsymbol{\Lambda}} \in \mathbb{R}^{(n-r)\times(n-r)}$ and $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{n\times(n-r)}$ are the truncated eigenvalues and eigenvectors, respectively. The reduced gen-

eralized eigenvalue problem can then be written as,

$$\begin{cases} \mathbf{M}\hat{\mathbf{\Phi}} - \mathbf{K}\hat{\mathbf{\Phi}}\hat{\mathbf{\Lambda}} = 0 \\ \hat{\mathbf{\Phi}}^{\intercal}\mathbf{M}\hat{\mathbf{\Phi}} - \hat{\mathbf{I}} = 0 \end{cases}, \tag{5.4}$$

where $\hat{\mathbf{I}}$ is an $r \times r$ identity matrix.

5.2 Background

Even though the adjoint method is written in a general form, some simplifications are made in Chapter 7 to make the problem more approachable. One simplification is that when we update the geometry of the wing during an aerodynamic shape optimization process, the mode shapes remain unchanged. If we only consider the shape variables, this simplification will not affect the final result much. However, when we consider aerostructural optimization with planform variables, the assumption no longer holds. Taking into this additional dependency, most of the derivation in Chapter 4 of partial derivatives are valid except for one

$$\frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \mathbf{x}}^{\intercal} \boldsymbol{\psi}, \tag{5.5}$$

which is decomposed into

$$\frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{x}}^{\intercal} \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Phi}}}^{\intercal} \boldsymbol{\psi} + \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{x}}^{\intercal} \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Lambda}}}^{\intercal} \boldsymbol{\psi} + \text{other terms}, \tag{5.6}$$

where $\hat{\mathbf{\Phi}}$ is the mode shapes, and $\hat{\mathbf{\Lambda}}$ is the natural frequencies. The derivative of the mode shapes and natural frequencies with respect to the design variables are defined using total derivative symbols $\mathrm{d}\,(\cdot)$ meaning that the derivative is computed satisfying the Eq. (5.1). With an aerostructural optimization, when the design variables are updated, the mode shapes and natural frequencies

change. Thus, $\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{x}$ and $\mathrm{d}\hat{\mathbf{\Lambda}}/\,\mathrm{d}\mathbf{x}$ are no longer equal to zero. The "other terms" are usually straightforward to compute, we focus on the computation of $\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{x}$ and $\mathrm{d}\hat{\mathbf{\Lambda}}/\,\mathrm{d}\mathbf{x}$.

We can expand Eq. (5.6) and directly relate Eq. (5.6) with Eq. (5.4). For Eq. (5.6), we define that

$$\begin{aligned}\bar{\hat{\mathbf{\Phi}}} &\coloneqq \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Phi}}}^{\intercal} \boldsymbol{\psi}, \\ \bar{\hat{\mathbf{\Lambda}}} &\coloneqq \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Lambda}}}^{\intercal} \boldsymbol{\psi},\end{aligned} \tag{5.7}$$

where $\bar{\hat{\mathbf{\Phi}}}$ and $\bar{\hat{\mathbf{\Lambda}}}$ are the weights of the outputs $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Lambda}}$. Notice that the design variables, $\mathbf{x}$, affect the mode shapes, $\hat{\mathbf{\Phi}}$, and natural frequencies, $\hat{\mathbf{\Lambda}}$ by changing the mass matrix, $\mathbf{M}$, and stiffness matrix, $\mathbf{K}$. Thus, using Eq. (5.7), the two spelled out terms from Eq. (5.6) can be written as

$$\begin{aligned}\frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{x}}^{\intercal} \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Phi}}}^{\intercal} \boldsymbol{\psi} &= \frac{\partial \mathbf{M}}{\partial \mathbf{x}}^{\intercal} \frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{M}}^{\intercal} \bar{\hat{\mathbf{\Phi}}} + \frac{\partial \mathbf{K}}{\partial \mathbf{x}}^{\intercal} \frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{K}}^{\intercal} \bar{\hat{\mathbf{\Phi}}}, \\ \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{x}}^{\intercal} \frac{\partial \boldsymbol{\mathcal{S}}_{\mathrm{TS}}}{\partial \hat{\mathbf{\Lambda}}}^{\intercal} \boldsymbol{\psi} &= \frac{\partial \mathbf{M}}{\partial \mathbf{x}}^{\intercal} \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{M}}^{\intercal} \bar{\hat{\mathbf{\Lambda}}} + \frac{\partial \mathbf{K}}{\partial \mathbf{x}}^{\intercal} \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{K}}^{\intercal} \bar{\hat{\mathbf{\Lambda}}}.\end{aligned} \tag{5.8}$$

Equation (5.8) could be expensive to evaluate if it is not done carefully. For example, a naive way to compute the component $(\partial \mathbf{M}/\partial \mathbf{x})^{\intercal} \left(\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{M}\right)^{\intercal} \bar{\hat{\mathbf{\Phi}}}$ is to apply FD method to evaluate each row (for each $\mathbf{x}_i,\ i = 1, \ldots n_{\mathbf{x}}$) from $\left(\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{M}\right)(\partial \mathbf{M}/\partial \mathbf{x})$. This requires us to solve the equations Eq. (5.1) $n_{\mathbf{x}}$ times (see Fig. 5.1). In the context of high-fidelity optimization, $n_{\mathbf{x}}$ is in the order of 10^2 to 10^3, and r is about 10 to 10^2. Thus, this naive way to compute this partial derivative requires 10^3 to 10^5 modal equation solutions! This becomes a bottleneck of the coupled adjoint derivative computation. This analysis is also valid for a FAD implementation. We iterate through each coordinate $i = 1, \ldots, n_{\mathbf{x}}$ where we set the forward seed with $\dot{\mathbf{x}}_i = 1$ and the rest be zero. Then, we compute the corresponding forward seeds $\dot{\mathbf{M}} = \partial \mathbf{M}/\partial \mathbf{x}_i$. Next, we call the FAD code (see Fig. 5.1) to compute $\dot{\hat{\mathbf{\Phi}}} = \left(\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{M}\right)\dot{\mathbf{M}}$. Finally, we evaluate the inner product of flattened $\dot{\hat{\mathbf{\Phi}}}$ and $\bar{\hat{\mathbf{\Phi}}}$. Thus, as we mentioned before, this method also requires us to call the FAD function $n_{\mathbf{x}}$ times (within each function call, there are r internal iterations through the eigenpairs). Similar

scaling analysis holds true for the rest of the terms from the right hand side (RHS) of Eq. (5.8).

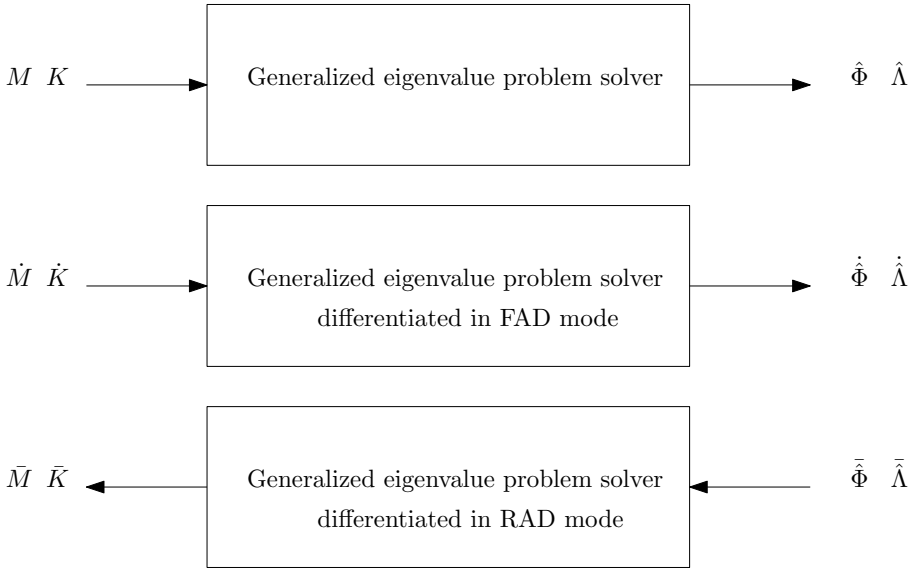


Figure 5.1: Schematic of the original generalized eigenvalue solver, the FAD mode solver, and the RAD mode solver. $\dot{\mathbf{M}}$ and $\dot{\mathbf{K}}$ are the coefficients of the directional derivatives, and $\bar{\hat{\mathbf{\Phi}}}$ and $\bar{\hat{\mathbf{\Lambda}}}$ are the weights of the weighted output derivative.

Fortunately, what we really want to compute is the derivative of a weighted output, i.e., $\hat{\mathbf{\Phi}}^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}}$ instead of the individual terms from $(\partial\mathbf{M}/\partial\mathbf{x})^{\mathsf{T}}\left(\mathrm{d}\hat{\mathbf{\Phi}}/\mathrm{d}\mathbf{M}\right)^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}}$. In this chapter, we develop two RAD based formulae that can evaluate this product with $\mathcal{O}(r)$ computations. The first formula only requires us to evaluate matrix-vector products and is free from any equation solution at the cost of some solution accuracy. And the second formula adds additional correction terms to the first formula to make it more accurate. Compared with $\mathcal{O}(r) \times n_{\mathbf{x}}$ function calls made by FAD or FD methods, the RAD formulae are preferred.

5.3 Derivation of the RAD formulae

In this section, we present methods that compute the following weighted output derivatives efficiently

$$\frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{M}}^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}}, \frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{K}}^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}}, \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{M}}^{\mathsf{T}}\bar{\hat{\mathbf{\Lambda}}}, \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{K}}^{\mathsf{T}}\bar{\hat{\mathbf{\Lambda}}}. \tag{5.9}$$

It is possible that not all the weights are non-zero. For example, we may only need to compute the derivative of the eigenvalues. Then, we have that $\bar{\hat{\mathbf{\Lambda}}} \neq 0$ and $\bar{\hat{\mathbf{\Phi}}} = 0$. Since the ultimate goal is to compute the derivative with respect to the design variables, $\mathbf{x}$, Eq. (5.8), it pays off to combine similar terms to reduce the computational cost. Thus, we derive formulae to compute $\overline{\mathbf{M}}$ and $\overline{\mathbf{K}}$ defined as follows

$$\begin{aligned}\overline{\mathbf{M}} &\coloneqq \frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{M}}^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}} + \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{M}}^{\mathsf{T}}\bar{\hat{\mathbf{\Lambda}}},\\ \overline{\mathbf{K}} &\coloneqq \frac{\mathrm{d}\hat{\mathbf{\Phi}}}{\mathrm{d}\mathbf{K}}^{\mathsf{T}}\bar{\hat{\mathbf{\Phi}}} + \frac{\mathrm{d}\hat{\mathbf{\Lambda}}}{\mathrm{d}\mathbf{K}}^{\mathsf{T}}\bar{\hat{\mathbf{\Lambda}}}.\end{aligned} \tag{5.10}$$

We present three methods: (1) The adjoint method, (2) the modal method, and (3) the improved modal method. The first formula treats the weighted outputs as the objective function and the equations of the generalized eigenvalue problem Eq. (5.1) as constraints. Then, an adjoint approach is applied to compute the derivatives. This approach was proposed by Lee [69], but here we provide a derivation and expressions suitable for RAD and adjoint implementations. The second and the third formulae, which we call the "*modal method*" and the "*improved modal method*", are novel and new contributions. Both formulae project the modal derivative onto the reduced space spanned by the computed modes. Due to the components of the derivatives that are orthogonal to the reduced space, neither method achieves machine precision. However, in the improved modal method, we introduce correction terms to improve its accuracy. Both RAD derivations rely on techniques proposed by Giles [35], Minka [103]. Furthermore, in these derivations, we leverage previously developed FAD based formulae proposed by Fox and Kapoor [32] for the modal method, and Lim et al. [76] and Wang [141] for the improved modal method.

The derivations presented detail the dependency between $(\mathbf{M}, \mathbf{K})$ and $(\mathbf{\Lambda}, \mathbf{\Phi})$. We omit the step relating derivatives of $(\mathbf{M}, \mathbf{K})$ to the design variables $\mathbf{x}$, which is problem-specific. In the case of a structural optimization problem, the omitted step relating $(\mathbf{M}, \mathbf{K})$ to $\mathbf{x}$ is usually provided by the FEM software with optimization capability. An example of such implementations can be found in the TACS open-source FEM package [59].[1]

5.3.1 Adjoint method

In this section, we present an adjoint derivation to compute the derivatives of the generalized eigenvalue problem. This method is based on the work proposed by Lee [69], but the derivation presented here is better suited for RAD implementation. In this approach, the weighted output function (a linear function) is treated as a function of interest and the generalized eigenvalue problem Eq. (5.1) as constraints. By applying the adjoint method to compute the derivatives, the computational cost is no longer dependent on the number of design variables, which motivates the development of this method.

The function of interest and the constraints are given as

$$\begin{aligned} I &= \bar{\boldsymbol{\phi}}_i^{\mathsf{T}} \boldsymbol{\phi}_i + \bar{\lambda}_i \lambda_i, \\ \mathbf{R} &= \begin{bmatrix} \mathbf{R}_1 \\ \mathrm{R}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{M}\boldsymbol{\phi}_i - \mathbf{K}\boldsymbol{\phi}_i \lambda_i \\ \boldsymbol{\phi}_i^{\mathsf{T}} \mathbf{M} \boldsymbol{\phi}_i - 1 \end{bmatrix}, \end{aligned} \tag{5.11}$$

where I is a function of interest, $\mathbf{R}$ are the residuals, λ_i and ϕ_i are the i^{th} eigenpair, and the $\bar{\cdot}$ represents the weight for the weighted output. We write the weighted output in the most general form. However, it is possible that we are only interest with portion of them, e.g., only the eigenvalue derivative is of interest. For this case, we can simply set $\bar{\boldsymbol{\phi}}_i = 0$. We further define the state

[1] https://github.com/gjkennedy/tacs

variables, $\mathbf{q}$, of the problem as,

$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\phi}_i \\ \lambda_i \end{bmatrix}. \tag{5.12}$$

The derivative of the function of interest, $\mathrm{d}I/\,\mathrm{d}\mathbf{x}$, equals

$$\frac{\mathrm{d}I}{\mathrm{d}\mathbf{x}} = \frac{\mathrm{d}\boldsymbol{\phi}_i}{\mathrm{d}\mathbf{x}}^{\mathsf{T}} \bar{\boldsymbol{\phi}}_i + \frac{\mathrm{d}\lambda_i}{d\mathbf{x}} \bar{\lambda}_i. \tag{5.13}$$

The procedures to compute the operations discussed above is presented in Fig. 5.2 using an extended design structure matrix (XDSM) format [68]. The function of interest, I, of the problem is only directly dependent on the state variable $\mathbf{q}$ and not the design variables. However, it is affected indirectly by the design variables through the state variables, which are impacted by $\mathbf{M}(\mathbf{x})$, and $\mathbf{K}(\mathbf{x})$. Therefore, we have, $\partial I/\partial \mathbf{x} = 0$ but $\mathrm{d}I/\,\mathrm{d}\mathbf{x} \neq 0$.

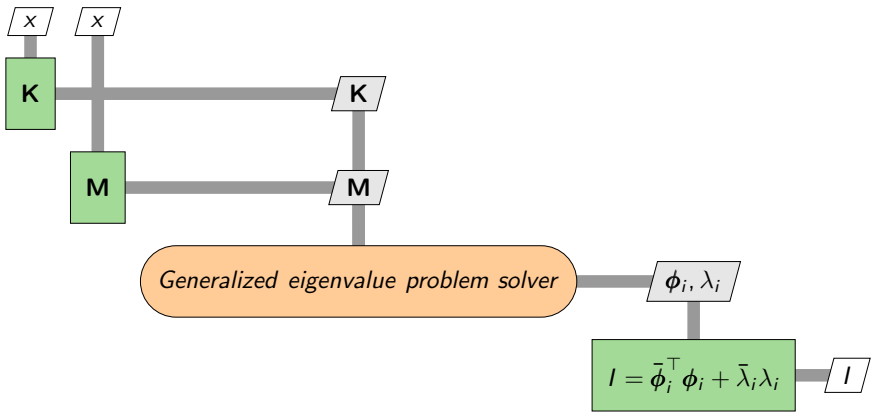


Figure 5.2: XDSM for the eigenvalue problem.

Differentiating both the function of interest and the residual with respect to the design variables and applying the adjoint method [1, 54], the total derivative of the function of interest can be written

as,

$$\begin{aligned}\frac{\mathrm{d}I}{\mathrm{d}\mathbf{x}} &= \frac{\partial I}{\partial \mathbf{x}} - \boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \\ &= -\boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{x}},\end{aligned} \tag{5.14}$$

where $\partial I/\partial \mathbf{x}$ is zero as explained in the previous paragraph, and the adjoint vector $\boldsymbol{\psi}_i$ is the solution of the adjoint equation,

$$\frac{\partial \mathbf{R}}{\partial \mathbf{q}}^\intercal \boldsymbol{\psi}_i = \frac{\partial I}{\partial \mathbf{q}}. \tag{5.15}$$

Expanding the residual and adjoint vectors and taking the partial derivatives to find the blocks of the Jacobian $\partial \mathbf{R}/\partial \mathbf{q}$, we obtain

$$\begin{bmatrix} \mathbf{M} - \mathbf{K}\lambda_i & -\mathbf{K}\boldsymbol{\phi}_i \\ 2\boldsymbol{\phi}_i^\intercal \mathbf{M} & \mathbf{0} \end{bmatrix}^\intercal \begin{bmatrix} \boldsymbol{\psi}_{i,\mathbf{R}_1} \\ \psi_{i,\mathrm{R}_2} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\phi}}_i \\ \bar{\lambda}_i \end{bmatrix}, \quad \boldsymbol{\psi}_i = \begin{bmatrix} \boldsymbol{\psi}_{i,\mathbf{R}_1} \\ \psi_{i,\mathrm{R}_2} \end{bmatrix}, \tag{5.16}$$

where $\boldsymbol{\psi}_{i,\mathbf{R}_1}$ is a subvector of $\boldsymbol{\psi}_i$, and ψ_{i,R_2} is an entry of $\boldsymbol{\psi}_i$.

Applying the chain rule to the Jacobian of the residuals, $\partial \mathbf{R}/\partial \mathbf{x}$, we obtain

$$\frac{\partial \mathbf{R}}{\partial \mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{M}} \frac{\partial \mathbf{M}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{K}} \frac{\partial \mathbf{K}}{\partial \mathbf{x}}, \tag{5.17}$$

where $\partial \mathbf{R}/\partial \mathbf{M}$ and $\partial \mathbf{R}/\partial \mathbf{K}$ are three-dimension tensors with dimension $\mathbb{R}^{n\times n\times n}$, where $(\partial \mathbf{R}/\partial \mathbf{M})_{ijk} = \partial \mathbf{R}_i/\partial \mathbf{M}_{jk}$, and similarly for $\partial \mathbf{R}/\partial \mathbf{K}$. Furthermore, $\partial \mathbf{M}/\partial \mathbf{x}$ and $\partial \mathbf{K}/\partial \mathbf{x}$ are three-dimensional tensors with dimension of $\mathbb{R}^{n\times n\times n_\mathbf{x}}$, where $(\partial \mathbf{M}/\partial \mathbf{x})_{ijk} = \partial \mathbf{M}_{ij}/\partial \mathbf{x}_k$, and similarly for $\partial \mathbf{K}/\partial \mathbf{x}$. Inserting Eq. (5.17) into Eq. (5.14) we obtain,

$$\begin{aligned}\frac{\mathrm{d}I}{\mathrm{d}\mathbf{x}} &= -\boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{M}} \frac{\partial \mathbf{M}}{\partial \mathbf{x}} - \boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{K}} \frac{\partial \mathbf{K}}{\partial \mathbf{x}}, \\ \overline{\mathbf{M}}_i^\intercal &\coloneqq -\boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{M}}, \\ \overline{\mathbf{K}}_i^\intercal &\coloneqq -\boldsymbol{\psi}_i^\intercal \frac{\partial \mathbf{R}}{\partial \mathbf{K}}.\end{aligned} \tag{5.18}$$

The products $-\boldsymbol{\psi}_i^\intercal \partial \mathbf{R}/\partial \mathbf{M}$ and $-\boldsymbol{\psi}_i^\intercal \partial \mathbf{R}/\partial \mathbf{K}$ are evaluated using RAD. The resulting derivatives expressions are derived as

$$\begin{aligned} \overline{\mathbf{M}}_i &= -\left(\boldsymbol{\psi}_{i,\mathbf{R}_1} + \boldsymbol{\phi}_i \psi_{i,\mathrm{R}_2}\right) \boldsymbol{\phi}_i^\intercal, \\ \overline{\mathbf{K}}_i &= \boldsymbol{\psi}_{i,\mathbf{R}_1} \lambda_i \boldsymbol{\phi}_i^\intercal, \end{aligned} \tag{5.19}$$

where $\overline{\mathbf{M}}_i \in \mathbb{R}^{n \times n}$ is the reduced mass matrix reverse seed, and $\overline{\mathbf{K}}_i \in \mathbb{R}^{n \times n}$ is the reduced stiffness matrix reverse seed (see the Appendix B.4 for the detailed derivation).

When multiple eigenvectors are considered, the previous expression can be more conveniently written as

$$\begin{aligned} \overline{\hat{\mathbf{M}}} &= -\left(\hat{\boldsymbol{\Psi}}_{\mathbf{R}_1} + \hat{\boldsymbol{\Phi}} \mathrm{Diag}\left(\hat{\boldsymbol{\Psi}}_{\mathrm{R}_2}\right)\right) \hat{\boldsymbol{\Phi}}^\intercal, \\ \overline{\hat{\mathbf{K}}} &= \hat{\boldsymbol{\Psi}}_{\mathbf{R}_1} \hat{\boldsymbol{\Lambda}} \hat{\boldsymbol{\Phi}}^\intercal, \end{aligned} \tag{5.20}$$

where $\overline{\hat{\mathbf{M}}} \in \mathbb{R}^{n \times n}$ is the reduced mass matrix reverse seed, $\overline{\hat{\mathbf{K}}} \in \mathbb{R}^{n \times n}$ is the reduced stiffness matrix reverse seed, $\hat{\boldsymbol{\Psi}}_{\mathbf{R}_1}$ columns are composed of $\boldsymbol{\psi}_{i,\mathbf{R}_1}$ with $i = 1, \ldots, r$, and $\hat{\boldsymbol{\Psi}}_{\mathrm{R}_2}$ diagonal is composed of ψ_{i,R_2} with $i = 1, \ldots, r$. To compute for the final solution, the adjoint method requires us to solve Eq. (5.16) r times the adjoint. For completeness, the direct method for computing the same derivatives is included in the Appendix B.3.

5.3.2 Modal method

In this section, we present a novel RAD formula that eliminates the need for solving Eq. (5.16) r times in the previous adjoint formula. The proposed method only requires evaluating $\mathcal{O}(r)$ of matrix-vector products instead of solving any equations, making it much cheaper at the cost of additional truncation error. Its better efficiency is achieved by projecting the derivative onto a reduced eigenspace. Later, we show that the relative error is about 1% when about 10 eigenpairs are considered for a beam problem.

5.3.2.1 Derivation

We derive the RAD formula using the FAD formula. We show the FAD formula in the literature at first. For the FAD based formula, we project the derivative in the basis spanned by the eigenvectors. The FAD expressions were derived by Fox and Kapoor [32] and can be equivalently written as

$$\begin{aligned}\dot{\mathbf{\Lambda}} &= \mathbf{\Lambda}\left(\mathbf{I}\circ\left(-\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\mathbf{\Phi}\mathbf{\Lambda}+\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)\right),\\ \dot{\mathbf{\Phi}} &= \mathbf{\Phi}\left(\mathbf{F}\circ\left(-\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\mathbf{\Phi}\mathbf{\Lambda}+\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)\right)-\frac{1}{2}\mathbf{\Phi}\left(\mathbf{I}\circ\left(\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)\right),\end{aligned} \tag{5.21}$$

where $\dot{\mathbf{K}},\dot{\mathbf{M}}$ are the input derivative seeds for the stiffness and mass matrices, respectively, $\dot{\mathbf{\Lambda}},\dot{\mathbf{\Phi}}$ are the output seeds for the eigenvalues and eigenvectors, respectively. The matrix $\mathbf{F}$ is defined as $\mathbf{F}_{ij}=\lambda_i/\left(\lambda_j-\lambda_i\right)$, the operator "$\circ$" is the Hadamard product, defined as $\left(\mathbf{A}\circ\mathbf{B}\right)_{i,j}\coloneqq A_{ij}B_{ij}$ where $\mathbf{A}$ and $\mathbf{B}$ are matrices.

The RAD expressions are derived using the matrix trace identities

$$\mathrm{Tr}(\overline{\mathbf{\Lambda}}^{\intercal}\dot{\mathbf{\Lambda}})+\mathrm{Tr}(\overline{\mathbf{\Phi}}^{\intercal}\dot{\mathbf{\Phi}})=\mathrm{Tr}(\overline{\mathbf{K}}^{\intercal}\dot{\mathbf{K}})+\mathrm{Tr}(\overline{\mathbf{M}}^{\intercal}\dot{\mathbf{M}}). \tag{5.22}$$

More details about the identity is given in Appendix B.2. Substituting in the FAD expressions, Eq. (5.21), and manipulating the equation we obtain

$$\begin{aligned}&\mathrm{Tr}\left(\overline{\mathbf{M}}^{\intercal}\dot{\mathbf{M}}\right)+\mathrm{Tr}\left(\overline{\mathbf{K}}^{\intercal}\dot{\mathbf{K}}\right)\\ =&\mathrm{Tr}\left(\left(\mathbf{\Phi}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}+\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}-\frac{1}{2}\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{M}}\right)\\ &+\mathrm{Tr}\left(\left(-\mathbf{\Phi}\mathbf{\Lambda}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}-\mathbf{\Phi}\mathbf{\Lambda}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{K}}\right).\end{aligned} \tag{5.23}$$

The full derivation of the RAD result is given in Appendix B.5. By matching the left hand side

(LHS) and RHS of Eq. (5.23), we conclude that

$$\begin{aligned}\overline{\mathbf{M}} &= \boldsymbol{\Phi}\left(\boldsymbol{\Lambda}\overline{\boldsymbol{\Lambda}} + \mathbf{F}\circ\left(\boldsymbol{\Phi}^{\intercal}\overline{\boldsymbol{\Phi}}\right) - \frac{1}{2}\mathbf{I}\circ\left(\boldsymbol{\Phi}^{\intercal}\overline{\boldsymbol{\Phi}}\right)\right)\boldsymbol{\Phi}^{\intercal},\\ \overline{\mathbf{K}} &= -\boldsymbol{\Phi}\left(\boldsymbol{\Lambda}\overline{\boldsymbol{\Lambda}} + \mathbf{F}\circ\left(\boldsymbol{\Phi}^{\intercal}\overline{\boldsymbol{\Phi}}\right)\right)\boldsymbol{\Lambda}\boldsymbol{\Phi}^{\intercal}.\end{aligned} \tag{5.24}$$

The above derivation assumes the knowledge of the full basis. If we want to work with a reduced set of eigenvalues and eigenvectors, we can write the same equation but for the reduced matrices,

$$\begin{aligned}\overline{\hat{\mathbf{M}}} &= \hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Lambda}}\overline{\hat{\boldsymbol{\Lambda}}} + \hat{\mathbf{F}}\circ\left(\hat{\boldsymbol{\Phi}}^{\intercal}\overline{\hat{\boldsymbol{\Phi}}}\right) - \frac{1}{2}\hat{\mathbf{I}}\circ\left(\hat{\boldsymbol{\Phi}}^{\intercal}\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Phi}}^{\intercal},\\ \overline{\hat{\mathbf{K}}} &= -\hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Lambda}}\overline{\hat{\boldsymbol{\Lambda}}} + \hat{\mathbf{F}}\circ\left(\hat{\boldsymbol{\Phi}}^{\intercal}\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Phi}}^{\intercal},\end{aligned} \tag{5.25}$$

where $\hat{\mathbf{F}} \in \mathbb{R}^{r\times r}$ is a submatrix of $\mathbf{F}$, defined as $\hat{\mathbf{F}} = \mathbf{F}(1:r, 1:r)$. In a memory-efficient implementation, the matrices $\overline{\hat{\mathbf{M}}}$ and $\overline{\hat{\mathbf{K}}}$ should not be stored explicitly. Instead, only the individual terms in the RHS of Eq. (5.25) should be stored. To be more specific, the specific size of each matrices are: $\hat{\boldsymbol{\Phi}}, \overline{\hat{\boldsymbol{\Phi}}} \in \mathbb{R}^{n\times r}$, $\hat{\boldsymbol{\Lambda}}, \overline{\hat{\boldsymbol{\Lambda}}} \in \mathbb{R}^{r\times r}$ (only the diagonal terms are non-zero), and $\hat{\mathbf{F}} \in \mathbb{R}^{r\times r}$. Further implementation recommendation are discussed in Section 5.4. Together with Eq. (5.18), we can compute the total derivatives.

5.3.2.2 Truncation error analysis

Due to the use of a reduced basis, a truncation error is implicitly introduced with the proposed method. The truncation error of the mass and stiffness matrices is defined as

$$\begin{aligned}\Delta\overline{\mathbf{M}} &= \overline{\mathbf{M}} - \overline{\hat{\mathbf{M}}},\\ \Delta\overline{\mathbf{K}} &= \overline{\mathbf{K}} - \overline{\hat{\mathbf{K}}}.\end{aligned} \tag{5.26}$$

By expanding Eq. (5.24) in terms of a reduced and truncated basis and subtracting Eq. (5.25), the truncation error is derived as

$$\begin{aligned}\Delta\overline{\mathbf{M}} &= \tilde{\boldsymbol{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Phi}}^\intercal,\\ \Delta\overline{\mathbf{K}} &= -\tilde{\boldsymbol{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Phi}}^\intercal,\end{aligned} \tag{5.27}$$

where $\tilde{\mathbf{F}}_2 = \mathbf{F}[r+1:n, 1:r]$. The full derivation of this result is included in Appendix B.6.

We can make several observations on the truncation error. First, we observe that if only the eigenvalues are seeded, i.e., $\overline{\hat{\boldsymbol{\Phi}}} = 0$, then $\Delta\overline{\mathbf{M}} = \Delta\overline{\mathbf{K}} = 0$. In other words, the eigenvalue RAD is computed accurately without any truncation error using the modal formula.

Second, let us consider the truncation error related to the eigenvector seeds. If the reverse seed, $\overline{\hat{\boldsymbol{\Phi}}}$, is in the space spanned by $\hat{\boldsymbol{\Phi}}$, then we have that $\tilde{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}} = 0$ due to orthogonality of the eigenvectors. Thus, by Eq. (5.27), the truncation error is zero for this special case. For more general cases the residual norm $\left|\left|\overline{\hat{\boldsymbol{\Phi}}} - \hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right)\right|\right|$ would indicate whether the truncation error is significant.

Finally, compared with the adjoint method, which requires solving linear equations, the proposed formula only involves matrix products. However, the disadvantage of the current approach is that it does not achieve machine precision like the adjoint method when eigenvector seeds are involved. Nevertheless, in Section 5.5, we demonstrate that a few eigenvectors (about 6 for the test case presented) give satisfactory results. To remedy this inaccuracy, we also propose an improved modal method in the following section. Also, as discussed before, if only the eigenvalue seeds are involved, we should use the modal based method due to its efficiency and accuracy.

5.3.3 Improved modal method

Instead of neglecting the higher-order eigenvectors' contribution completely, we can approximate it using the method proposed by Lim et al. [76], Wang [141]. Following this approach, the

eigenvalue output seed is the same as for the modal based method expressed in Eq. (5.21). While, the eigenvector output seed in Eq. (5.21) needs to be modified. The full derivation of the FAD formulae is given in Appendix B.7. The final expression for the eigenvector seed is restated here

$$\begin{aligned}\dot{\boldsymbol{\phi}}_i = &\sum_{l=1, l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i - \lambda_l} \\ &- \sum_{l=1}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} + \mathbf{K}^{-1}(-\dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{1}{\lambda_i} \\ &- \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right),\end{aligned} \tag{5.28}$$

where $i = 1, \ldots, r$. The equation can then be cast back into the matrix form as,

$$\begin{aligned}\dot{\hat{\boldsymbol{\Phi}}} = &\hat{\boldsymbol{\Phi}}\left(\hat{\mathbf{F}} \circ \left(-\hat{\boldsymbol{\Phi}}^\intercal \dot{\mathbf{K}}\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}} + \hat{\boldsymbol{\Phi}}^\intercal \dot{\mathbf{M}}\hat{\boldsymbol{\Phi}}\right)\right) \\ &- \hat{\boldsymbol{\Phi}}\left(\hat{\mathbf{G}} \circ \left(-\hat{\boldsymbol{\Phi}}^\intercal \dot{\mathbf{K}}\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}} + \hat{\boldsymbol{\Phi}}^\intercal \dot{\mathbf{M}}\hat{\boldsymbol{\Phi}}\right)\right) + \mathbf{K}^{-1}\left(-\dot{\mathbf{K}}\hat{\boldsymbol{\Phi}} + \dot{\mathbf{M}}\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}}^{-1}\right) \\ &- \frac{1}{2}\hat{\boldsymbol{\Phi}}\left(\mathbf{I} \circ \left(\hat{\boldsymbol{\Phi}}^\intercal \dot{\mathbf{M}}\hat{\boldsymbol{\Phi}}\right)\right),\end{aligned} \tag{5.29}$$

where $\hat{\mathbf{G}} \in \mathbb{R}^{r\times r}$ is the approximate eigenvalue ratio, $\hat{\mathbf{G}}_{ij} \coloneqq \lambda_i/\lambda_j$.

The reverse derivative can be derived similarly to the modal method approach discussed in the previous section and is given as,

$$\begin{aligned}\overline{\hat{\mathbf{M}}} &= \hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Lambda}}\overline{\hat{\boldsymbol{\Lambda}}} + (\hat{\mathbf{F}} - \hat{\mathbf{G}}) \circ \left(\hat{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right) - \frac{1}{2}\hat{\mathbf{I}} \circ \left(\hat{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Phi}}^\intercal + \left(\mathbf{K}^{-1}\overline{\hat{\boldsymbol{\Phi}}}\right)\hat{\boldsymbol{\Lambda}}^{-1}\hat{\boldsymbol{\Phi}}^\intercal, \\ \overline{\hat{\mathbf{K}}} &= -\hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Lambda}}\overline{\hat{\boldsymbol{\Lambda}}} + (\hat{\mathbf{F}} - \hat{\mathbf{G}}) \circ \left(\hat{\boldsymbol{\Phi}}^\intercal\overline{\hat{\boldsymbol{\Phi}}}\right)\right)\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Phi}}^\intercal - \left(\mathbf{K}^{-1}\overline{\hat{\boldsymbol{\Phi}}}\right)\hat{\boldsymbol{\Phi}}^\intercal.\end{aligned} \tag{5.30}$$

This approach relies on solving the system of equations $\mathbf{K}\hat{\mathbf{y}}_i = \overline{\hat{\boldsymbol{\phi}}}_i$ of size n for $i = 1, \ldots, r$. We can write this in matrix form

$$\mathbf{K}\hat{\mathbf{Y}} = \overline{\hat{\boldsymbol{\Phi}}}, \tag{5.31}$$

where $\hat{\mathbf{y}}_i$ for $i = 1, \ldots r$ are the columns of $\hat{\mathbf{Y}}$. Finally, as observed previously in Section 5.3.2,

the matrices $\overline{\hat{\mathbf{M}}}, \overline{\hat{\mathbf{K}}}$ should not be stored explicitly, and instead, we should store the terms in the RHS. This is explained in more detail in Section 5.4.

5.4 Implementation recommendations

In the previous section, we presented the mathematical derivations of the proposed formulae. However, for a successful implementation in practice, additional considerations are often necessary. In this section, we give some recommendations on how to implement the proposed methods in practice to optimize both their memory requirement and speed.

5.4.1 Recommendation 1: Never form $\overline{\hat{\mathbf{M}}}, \overline{\hat{\mathbf{K}}}$ matrices explicitly

This recommendation is related to memory use efficiency. Inspecting Eq. (5.20), Eq. (5.25), and Eq. (5.30) it is evident that $\overline{\hat{\mathbf{M}}}, \overline{\hat{\mathbf{K}}}$ are dense $n \times n$ matrices, with rank of no more than r. Further, we notice that all formulae can be expressed as

$$\sum_{j=1}^{l} \hat{\mathbf{P}}_j \mathbf{S}_j \hat{\mathbf{\Phi}}^{\intercal}, \tag{5.32}$$

where $\hat{\mathbf{P}}_j \in \mathbb{R}^{n \times r}$, $\mathbf{S}_j \in \mathbb{R}^{r \times r}$, and l are number of terms. For example, consider $\overline{\hat{\mathbf{K}}}$ from the adjoint method, Eq. (5.20), where $\hat{\mathbf{P}}_1 = \hat{\mathbf{\Psi}}_{\mathbf{R}_1}$, $\mathbf{S}_1 = \hat{\mathbf{\Lambda}}$ and $l = 1$. Instead of storing each of the matrices explicitly, the individual terms $\hat{\mathbf{P}}_j$ and $\mathbf{S}_j$ are stored, in addition to the already stored reduced set of eigenvalue and eigenvectors. This requires $\mathcal{O}(n \times r)$ units of memory for each matrix. If the matrices are stored explicitly, $\mathcal{O}(n \times n)$ units of memory are required for each of them, which could be a memory bottleneck for a program considering the original matrix is stored in sparse format. Also, by not evaluating the product, we reduce the computational cost.

5.4.2 Recommendation 2: Reuse factorized matrices

In the adjoint and improved modal methods, we need to solve linear equations. The specialized adjoint equation is probably not implemented for a regular FEM code with optimization capability. However, for any FEM solver, the procedure to compute and solve equations on the same form as Eq. (5.31) must already exist as it resembles the linear elastic equation. If a direct solver is used (see [59]), then the matrix $\mathbf{K}$ has already been factorized during the solution stage before we compute the derivatives. Thus, solving Eq. (5.31) can be done with a relatively low cost by taking advantage of the already factorized matrix.

5.5 Derivative verification

To verify the three formulae from the previous section, we present verification cases based on a beam structural model. We benchmark the proposed methods by comparing against previously implemented reverse Lanczos iteration method [55].

5.5.1 Numerical model

We use a simple beam FEM to construct representative mass and stiffness matrices that can be used to verify the proposed formulae. The geometry under consideration, shown in Fig. 5.3, is a cantilevered beam discretized using Euler–Bernoulli beam elements. The geometry dimensions,

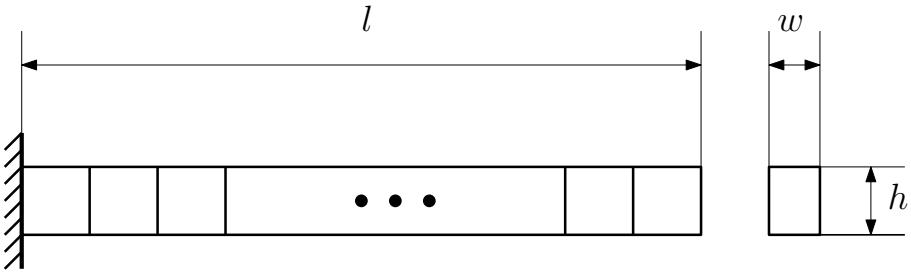


Figure 5.3: Finite-element model of the beam.

discretization, and material property are defined in Table 5.1.

Table 5.1: Geometry dimensions, discretization, and material properties of the beam model.

Property	**Value**
l	1 m
w	0.01 m
h	0.01 m
N_{elements}	40
E	200 GPa
ρ	8000 kg/m^3

5.5.2 Dot product test

To verify the RAD analytic expression derived in the previous sections, we perform several derivative verification tests. To verify the reverse implementation, we perform a dot product test where the input seeds are randomly generated [46]. A dot product test demonstrates that the forward and reverse formulae and implementation are consistent; it is a necessary but not sufficient condition for correct derivatives. In our case, the dot product test can be done by comparing both sides of Eq. (B.8), which should agree to machine precision. Representative results from the dot product are listed in Table 5.2. Here, we apply *numpy.linalg* package [140] to solve Eqs. (5.16) and (5.31) which is a wrapper for underlying LAPACK modules [2] using pivoted LU decomposition for linear system solution [37]. All methods match each other close to machine precision.

5.5.3 Lanczos iteration benchmark

To verify the derivative correctness and accuracy of the adjoint and modal methods, we apply a previously verified Lanczos implementation [55] in reverse mode. In this study, we randomly

Table 5.2: Dot product test results.

	$\mathrm{Tr}\left(\overline{\mathbf{M}}^\intercal\dot{\mathbf{M}}+\overline{\mathbf{K}}^\intercal\dot{\mathbf{K}}\right)$	$\mathrm{Tr}\left(\overline{\mathbf{\Phi}}^\intercal\dot{\mathbf{\Phi}}+\overline{\mathbf{\Lambda}}^\intercal\dot{\mathbf{\Lambda}}\right)$
Adjoint method	-224164.0517095295	-224164.0517088695
Modal method (full basis)	-224164.051708143588	-224164.051708143582
Improved modal method (full basis)	-224164.051708078874	-224164.051708069960

generate the input seeds for the eigenvalue and eigenvector, and check the error of the outputs $\overline{\mathbf{M}}$ and $\overline{\mathbf{K}}$. We use the following definitions to quantify the errors. The RAD relative error is defined as

$$\begin{aligned}\epsilon_{\overline{\mathbf{M}}} &:= \frac{\left\|\overline{\mathbf{M}}-\overline{\mathbf{M}}_{\text{ref}}\right\|_F}{\left\|\overline{\mathbf{M}}_{\text{ref}}\right\|_F},\\ \epsilon_{\overline{\mathbf{K}}} &:= \frac{\left\|\overline{\mathbf{K}}-\overline{\mathbf{K}}_{\text{ref}}\right\|_F}{\left\|\overline{\mathbf{K}}_{\text{ref}}\right\|_F},\end{aligned} \tag{5.33}$$

where $\cdot_F$ denotes the Frobenius norm, $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j}|a_{ij}|^2}$ and $\overline{\mathbf{M}}_{\text{ref}}$ and $\overline{\mathbf{K}}_{\text{ref}}$ are the reference values for the reverse mass and stiffness matrices. Another frequently used norm for matrix is p-norms, defined as $\|\mathbf{A}\|_p = \sup\frac{\|\mathbf{A}\mathbf{y}\|_p}{\|\mathbf{y}\|_p}$ induced by vector p-norms. However, the Frobenius norm is commonly used as it is easier and more directly computed than the 2-norm for matrices. It can be shown that the 2-norm is related with the Frobenius norm through the following inequality

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq \sqrt{n}\,\|\mathbf{A}\|_2\,, \tag{5.34}$$

where $\mathbf{A}$ is an $n \times n$ matrix. This is equivalently written as,

$$\frac{1}{\sqrt{n}}\,\|\mathbf{A}\|_F \leq \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F\,. \tag{5.35}$$

Thus, the 2-norm is bound by the Frobenius norm by multiplying a factor, and vice versa.

We compare the adjoint method and modal method with full basis with a reverse mode imple-

mentation of the Lanczos method. The results are listed in Table 5.3. Overall, the results agree well with the reference but do not reach a full machine precision. The error in the eigenvector is greater than the error in the eigenvalues. The reason for this may be that the reverse Lanczos does not guarantee to give machine precise results due to its iterative nature.

Table 5.3: RAD relative error using a random eigenvector or eigenvalue seed compared with a reverse mode implementation of a Lanczos method [55].

		$\epsilon_{\overline{\mathbf{M}}}$	$\epsilon_{\overline{\mathbf{K}}}$
Eigenvector	Adjoint method	$9.585173902423778 \times 10^{-7}$	$5.960441261582237 \times 10^{-8}$
	Modal method (full basis)	$9.585170660956007 \times 10^{-7}$	$5.959330021020872 \times 10^{-8}$
	Improved modal method (full basis)	$9.585174968559026 \times 10^{-7}$	$5.960289226431842 \times 10^{-8}$
Eigenvalue	Adjoint method	$1.7853147735487777 \times 10^{-10}$	$5.479618561347628 \times 10^{-12}$
	Modal method (full basis)	$1.3251425390595622 \times 10^{-11}$	$2.408203987921325 \times 10^{-11}$
	Improved modal method (full basis)	$1.3251425390595622 \times 10^{-11}$	$2.408203987921325 \times 10^{-11}$

We compare the modal method with a different number of truncated basis, against the reverse Lanczos implementation. First, we conduct a test with $\bar{\lambda}_1 \neq 0$ and all other eigenvalues and eigenvectors with zero seeds. The results are listed in Fig. 5.4; they show that the error is around 10^{-11}, and is independent of the number of modes considered. The modal and the improved modal methods exhibit the same error. As discussed in the previous section, the reason for this is that the eigenvalue RAD does not contribute to the truncation error.

Next, we conduct another test in which $\bar{\phi}_1 \neq 0$ and all eigenvalues and other eigenvectors have zero seeds. In Fig. 5.4, we show the eigenvector RAD results compared against the Lanczos method. When about 10 modes are considered, the relative error in the mass and stiffness reverse seeds is reduced by about 2 orders. For the improved modal method, the relative error quickly reduces to below 10^{-6} using fewer than 10 modes. This demonstrates the superiority in accuracy of the proposed method.

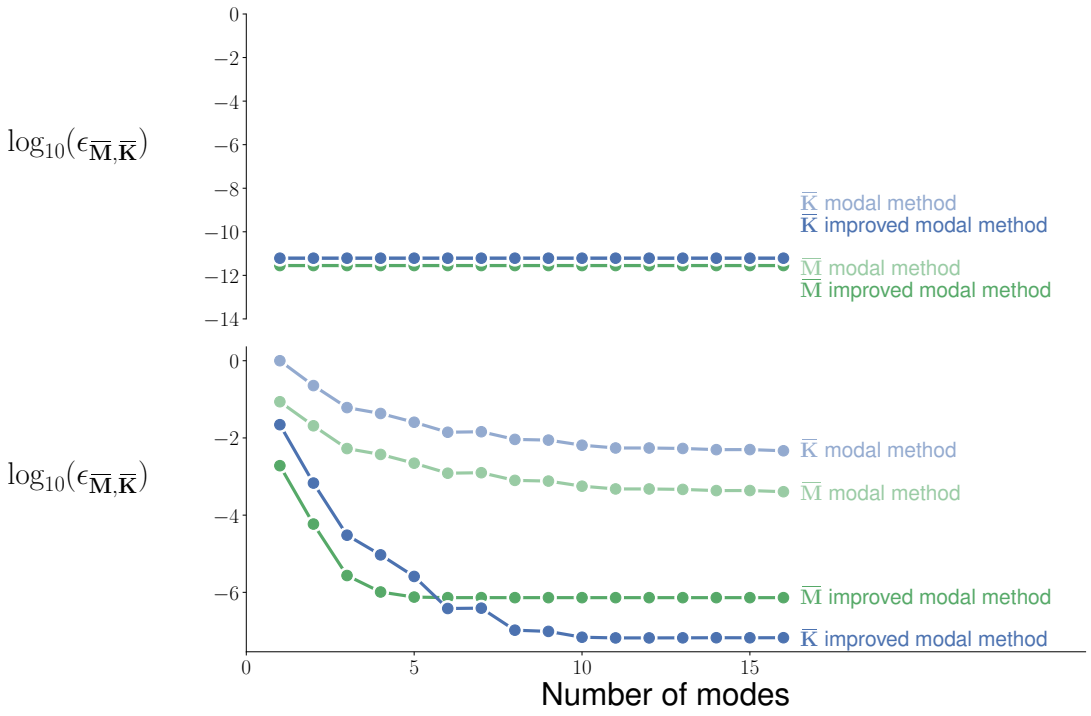


Figure 5.4: Upper: Eigenvalue derivative relative error. Lower: Eigenvector derivative relative error. We test RAD: $\bar{\lambda}_1 \rightarrow \overline{\mathbf{M}}, \overline{\mathbf{K}}$, and compare with a reverse mode Lanczos iteration. Total degrees-of-freedom is 80.

To conclude, if great accuracy is the primary concern, we recommend the use of the adjoint method. However, if about 1% error is good enough and computational efficiency is more important than accuracy, then we recommend the modal method. The improved modal method is somewhere in between these two extremes—it is more accurate compared to the modal method but also more expensive. Compared to the adjoint method, it is easier to solve the equations for the improved modal method than the adjoint method. This is because the linear equations of the improved modal method have the same coefficients as the linear elastic equations, whose solution strategy has been extensively studied. Also, if a direct solver for the elastic equation is used for an optimization problem, the stiffness matrix, $\mathbf{K}$, is factorized for each major step, making $\mathbf{K}^{-1}\mathbf{y}_i$

much cheaper to evaluate [59].

5.6 Summary

In this chapter, we presented RAD based formulae for the eigenvalue and eigenvectors derivative computations. These approaches are suitable for applications in optimization problems where eigenvalues and eigenvectors are affected by design variables, such as, flutter, LCO, and transition prediction. Methods in the literature focus on problems with more outputs than inputs. When there are more inputs than outputs, we need to develop new efficient formulae. RAD based formulae can address this need. We presented three formulae: (1) The adjoint method, (2) the modal method, and (3) the improved modal method. Out of the three methods, (2) and (3) are novel. The three methods have different characteristics and can be applied in accordance with the desired accuracy and speed. The adjoint method theoretically can achieve the machine precision but it requires solving $n+1$ dimensional linear equations r times. The proposed modal method projects the derivative in the space spanned by the eigenvectors. It only requires evaluating matrix products to compute the RAD derivatives. However, it is not as accurate as the adjoint method when only a handful of eigenpairs are known. The proposed improved modal method is based on the modal method with additional correction terms. It is more accurate than the modal method, requiring fewer modes to obtain the same accuracy. However, to achieve this improved accuracy, we need to solve r systems of equations $n \times n$ in size, increasing the computational cost. When implementing these RAD methods, we need to take care of memory footprint and leveraging existing tools as discussed in Section 5.4. The formulae and theory derived in this chapter were verified and found to be consistent with our previous reverse mode Lanczos iteration implementation. These conclusions are summarized in Table 5.4.

Table 5.4: Summary of methods.

	$\overline{\mathbf{M}}$	$\overline{\mathbf{K}}$	Equation to solve	Accurate or approximate
Adjoint method	$-\left(\hat{\mathbf{\Psi}}_{\mathbf{R}_1}+\hat{\mathbf{\Phi}}\text{Diag}\left(\hat{\mathbf{\Psi}}_{\mathrm{R}_2}\right)\right)\hat{\mathbf{\Phi}}^{\intercal}$	$\hat{\mathbf{\Psi}}_{\mathbf{R}_1}\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^{\intercal}$	$\begin{bmatrix}\mathbf{M}-\mathbf{K}\lambda_i & -\mathbf{K}\boldsymbol{\phi}_i\\ 2\boldsymbol{\phi}_i^{\intercal}\mathbf{M} & 0\end{bmatrix}^{\intercal}\begin{bmatrix}\psi_{i,\mathbf{R}_1}\\ \psi_{i,\mathrm{R}_2}\end{bmatrix}=\begin{bmatrix}\bar{\phi}_i\\ \bar{\lambda}_i\end{bmatrix}, i=1,2,\ldots,r$	Accurate
Modal method	$\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}}+\hat{\mathbf{F}}\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^{\intercal}$ $+\hat{\mathbf{\Phi}}\left(-\frac{1}{2}\hat{\mathbf{I}}\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^{\intercal}$	$-\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}}+\hat{\mathbf{F}}\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^{\intercal}$	N/A	Approximate
Improved modal method	$\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}}+(\hat{\mathbf{F}}-\hat{\mathbf{G}})\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^{\intercal}$ $+\hat{\mathbf{\Phi}}\left(-\frac{1}{2}\hat{\mathbf{I}}\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^{\intercal}$ $+\left(\mathbf{K}^{-1}\overline{\hat{\mathbf{\Phi}}}\right)\hat{\mathbf{\Lambda}}^{-1}\hat{\mathbf{\Phi}}^{\intercal}$	$-\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}}+(\hat{\mathbf{F}}-\hat{\mathbf{G}})\circ\left(\hat{\mathbf{\Phi}}^{\intercal}\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^{\intercal}$ $-\left(\mathbf{K}^{-1}\overline{\hat{\mathbf{\Phi}}}\right)\hat{\mathbf{\Phi}}^{\intercal}$	$\mathbf{K}\hat{\mathbf{y}}_i=\bar{\hat{\boldsymbol{\phi}}}_i$ $i=1,2,\ldots,r$	Approximate

CHAPTER 6

Flutter and LCO Analysis Results

In this chapter, we present flutter and LCO simulation results using CNK method discussed earlier in Chapter 3. This chapter is organized as follows: In Section 6.1, we present the two-dimensional airfoil results. In addition to the flutter and LCO simulation, we also present aerodynamic only benchmark cases. We also study GCL, and preconditioner performance in this section. Then, in Section 6.2, we demonstrate the three-dimensional wing results.

6.1 Airfoil results

In this section, we present numerical results for aerodynamic, LCO, flutter boundary, and preconditioner performance cases. The computations were completed using a high-performance parallel computer with nodes composed of two 3.0 GHz Intel Xeon Gold processors, for a total of 36 cores and 180 GB memory per node.

6.1.1 Aerodynamic benchmark with prescribed motion

Here we study the flutter and LCO characteristics of the NACA 64A010 airfoil using the proposed time-spectral approach. This test case–known as CT6—has been extensively studied under various conditions in the literature, both experimentally [23], and numerically [70, 137]. The specifications of the flow case are listed in Table 6.1.

Table 6.1: Specifications for the CT6 test case [23].

Parameter	Value
Mean angle of attack	$0.0°$
Pitching amplitude	$1.02°$
Mach number	0.796
Reduced frequency (for half-chord length)	0.202
Pitching axis	$0.248c$

Three levels of meshes are generated for both Euler and RANS cases using the following steps. The airfoil geometry is fit with a spline using the open-source package pySpline[1] to generate a suitable surface discretization. For the Euler case, the trailing edge is sharp, but for the RANS case, the trailing edge is blunt. The blunt trailing edge is obtained by cutting the original airfoil at 99% chord length and then resizing it to 100% chord length. The volume mesh is then extruded using the hyperbolic mesh generator pyHyp[2], such that the far-field is about 10 chords away from the geometry. All meshes are one cell in the spanwise direction, with symmetry planes on both sides to simulate two-dimensional flow.

The Euler and RANS meshes are shown in Fig. 6.1 and Fig. 6.2, respectively. The detailed mesh size information is listed in Table 6.2. The first layer cell thickness is 10^{-6} of the chord length for the RANS fine mesh, which corresponds to $y^+ = 0.29$ for the aerodynamic test case.

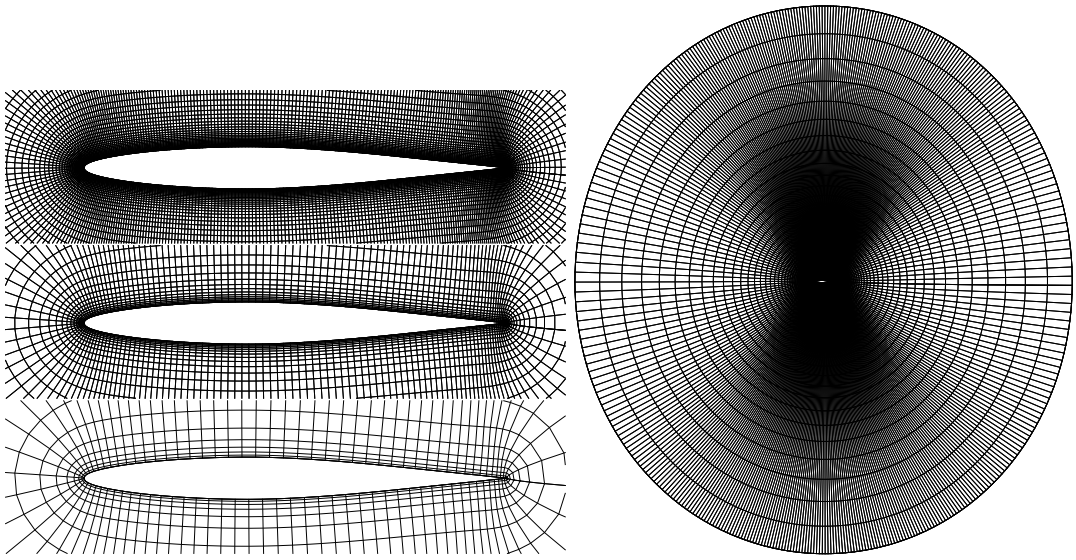
Table 6.2: Mesh sizes.

Type	Coarse	Medium	Fine
Euler	96×32	192×64	384×128
RANS	136×32	272×64	544×128

For the boundary conditions, we use far-field boundary conditions at the edges of the mesh for

[1] https://github.com/mdolab/pyspline
[2] https://github.com/mdolab/pyhyp

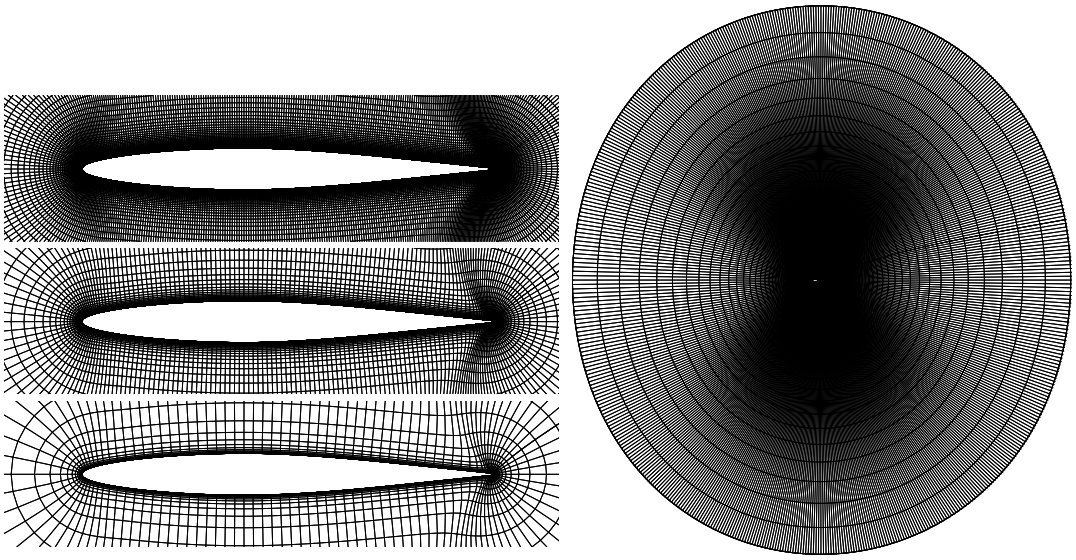
both Euler and RANS meshes. A symmetrical boundary condition is applied on the planes on both sides of the airfoil. For the Euler case, an inviscid wall boundary condition is applied and for the RANS case an adiabatic viscous wall boundary condition is used. For the SA model, we enforce turbulence state variables to be zero at the wall and apply the turbulence model far-field boundary condition at the edges of the mesh.



(a) Detailed view of the fine, medium, and coarse meshes.

(b) Fine mesh complete domain. The radius of the far field is about 10 chord lengths.

Figure 6.1: NACA 64A010 Euler meshes.



(a) Detailed view of the fine, medium, and coarse meshes.

(b) Fine mesh complete domain. The radius of the far field is about 10 chord lengths.

Figure 6.2: NACA 64A010 RANS meshes.

Figures 6.3 and 6.4 show computed C_l and C_m hysteresis compared to the experimental results for the inviscid and viscous cases, respectively.

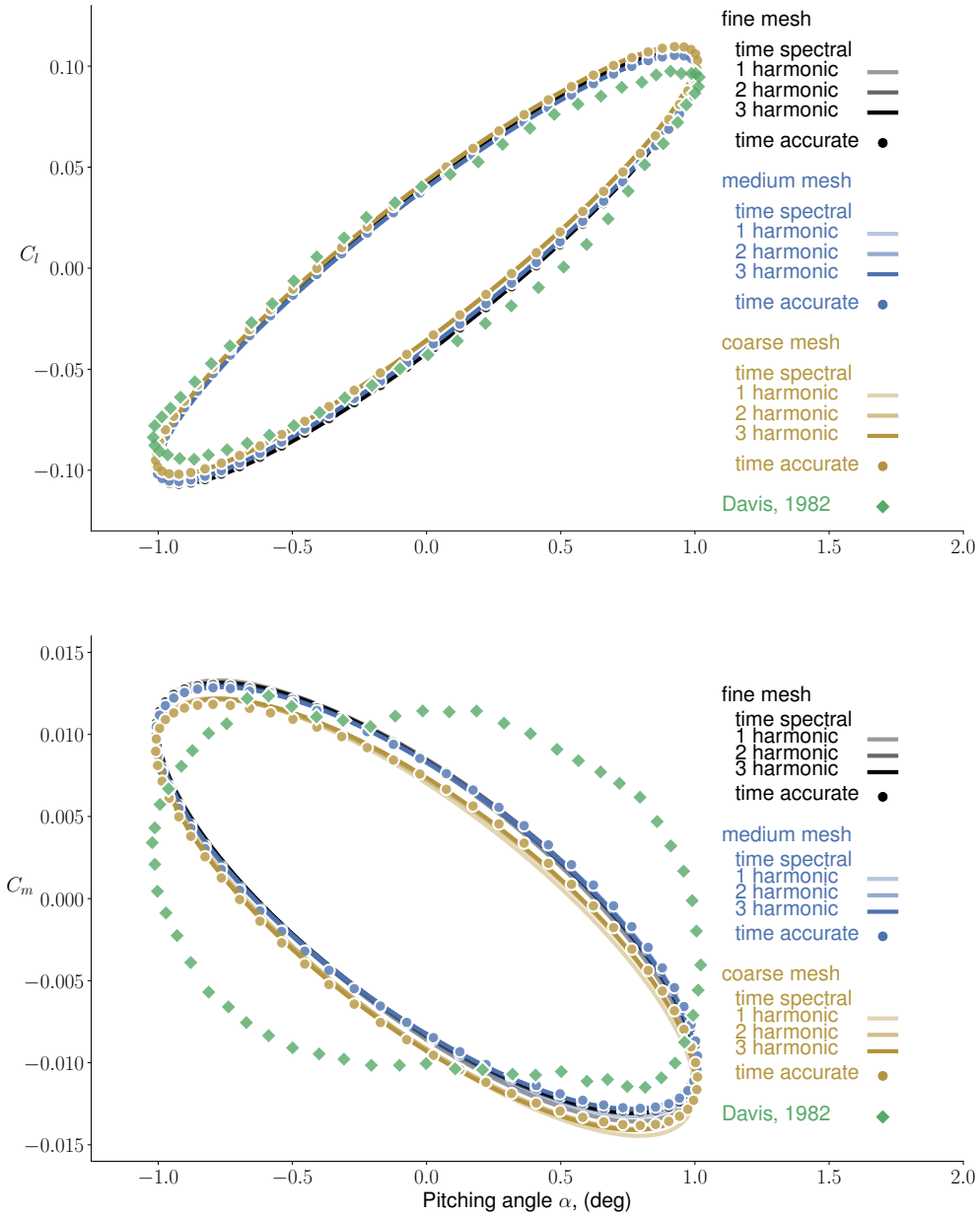


Figure 6.3: NACA 64A010 prescribed motion inviscid C_l, C_m curves benchmarked with experimental results by Davis [23]. $N = 256$ for the time-accurate solutions. Every fourth point is shown.

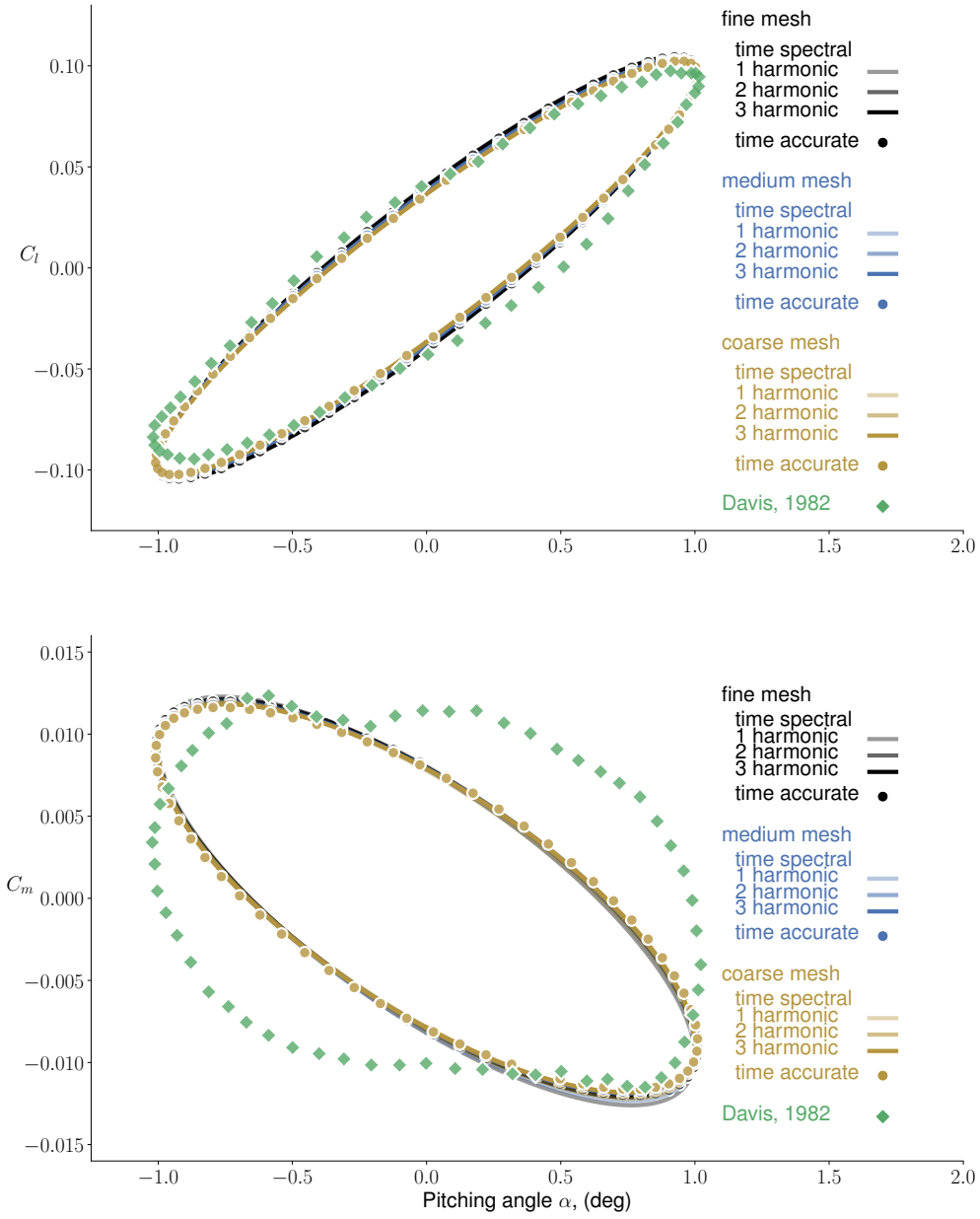


Figure 6.4: NACA 64A010 prescribed motion viscous C_l, C_m curves benchmarked with experimental results by Davis [23]. $N = 256$ for the time-accurate solutions. Every fourth point is shown.

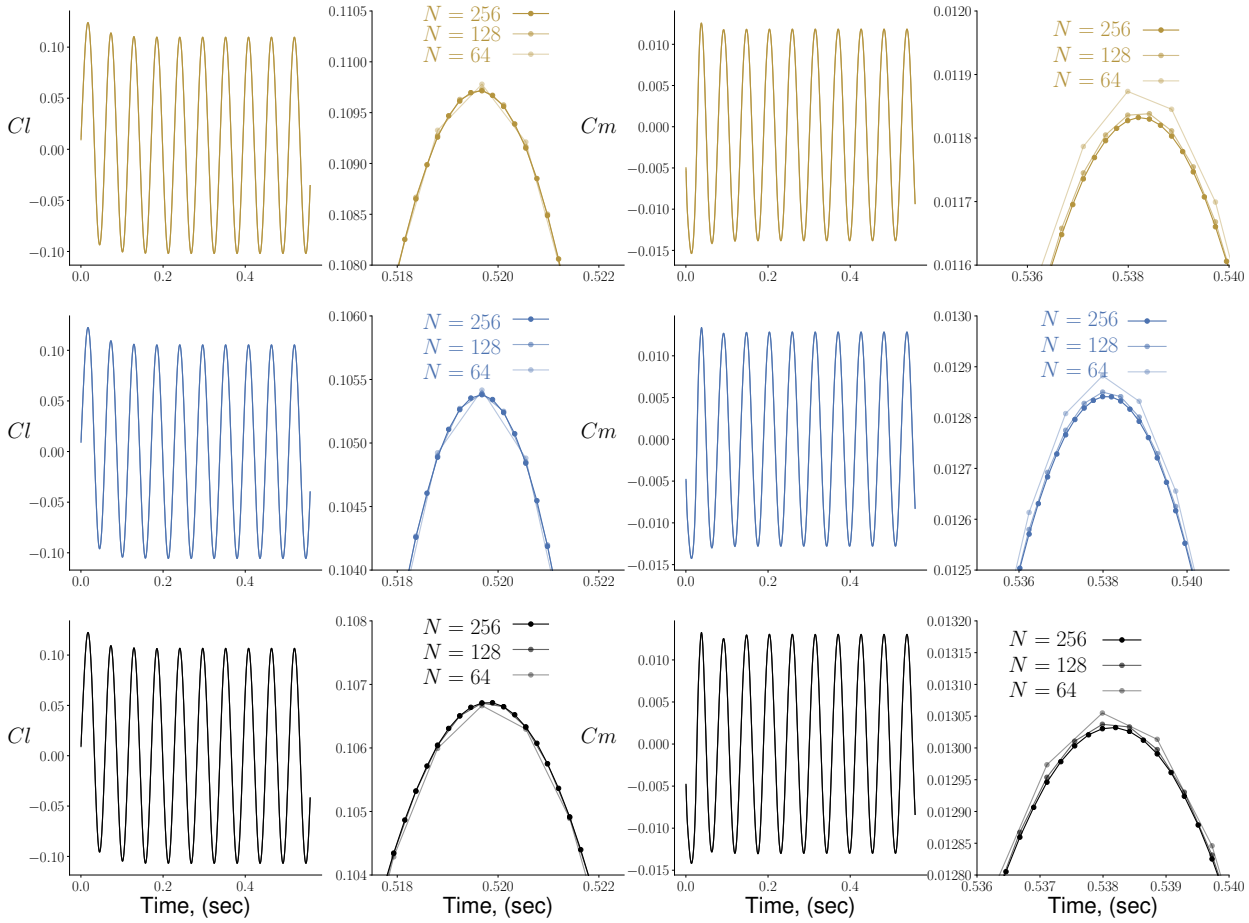


Figure 6.5: Inviscid time-accurate load history with different steps sizes and mesh levels. From top to bottom are coarse, medium and fine mesh results, respectively.

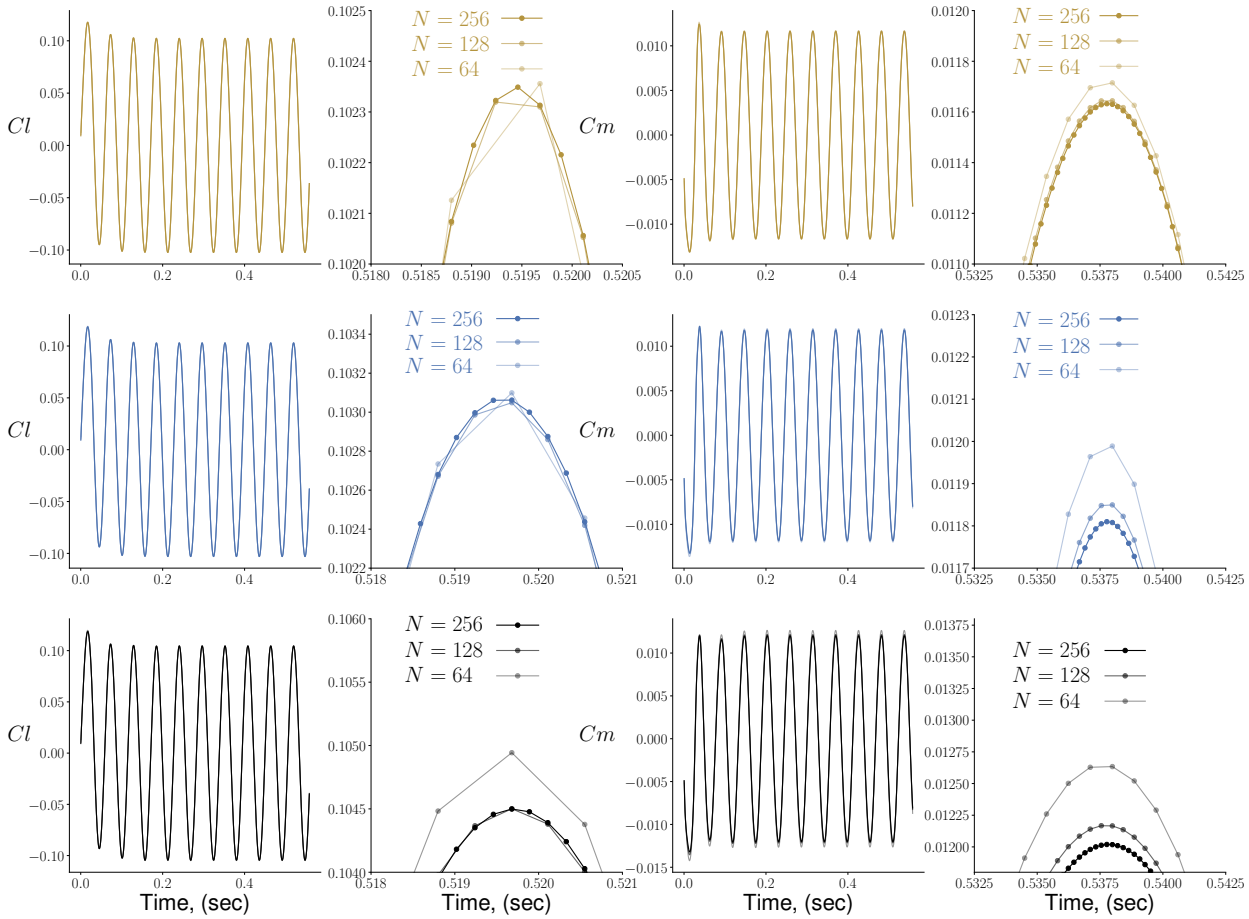


Figure 6.6: Viscous time-accurate load history with different steps sizes and mesh levels. From top to bottom are coarse, medium and fine mesh results, respectively.

For verification, the time-spectral solution is compared with a forced motion time-accurate solution, which is conducted for all three mesh levels for both inviscid and viscous test cases. For the time-accurate solution, we conduct a step size convergence study using $N = 64, 128$ and 256 where N is the number of steps per period. The results are shown in Figs. 6.5 and 6.6. Furthermore, on each mesh level, the number of harmonics is varied to study the effect on the solution. In general, for both C_l and C_m, the time-spectral and time-accurate (using $N = 256$) solutions show good agreement on all meshes Figs. 6.3 and 6.4. The detailed information for the relative error and simulation times are shown in Tables 6.3 to 6.6. The solution time using the time-spectral method

is smaller than the time-accurate solution. The RANS time-spectral simulation using the fine mesh does not have a similar computational time enhancement compared with other levels of meshes. This was found to be caused by multiple fractional steps taken during approximate Newton–Krylov (ANK) solution process. In general, for the time-spectral method with more frequencies included, the prediction matches better with that of the time-accurate method at a cost of increased simulation time. Though there is an exception that using 3 harmonics for the inviscid case is not showing consistent improvement over using 2 harmonics. The non-monotonic behavior of the maximum C_l error could be caused by several reasons: 1) The time-accurate reference is not fully converged with respect to the total simulation time or the number of time steps used per period; 2) The time-spectral solver does not reach the monotonical convergence region with respect number of time instances, suggesting that more harmonics may be needed to see a monotonic convergence. More tests remain to be done to get more accurate solutions. Using more frequency components results in more computational time as shown in Tables 6.5 and 6.6. We find that C_l predicted by the time-spectral method matches better with that predicted by time-accurate compared with C_m.

Table 6.3: Maximum C_l and C_m predicted by time-accurate and time-spectral method using inviscid flow model.

Mesh	Property	TA ($N = 256$)	TS 1	TS 2	TS 3
C_l	coarse	0.109715	0.109495 (-0.201%)	0.109812 (0.0885%)	0.109700 (-0.0144%)
	medium	0.105380	0.105313 (-0.0633%)	0.105400 (0.0191%)	0.105178 (-0.192%)
	fine	0.106712	0.106686 (-0.0237%)	0.106773 (0.0579%)	0.106581 (-0.120%)
C_m	coarse	0.0118323	0.0122157 (3.24%)	0.0121171 (2.41%)	0.0121752 (2.90%)
	medium	0.0128413	0.0130259 (1.44%)	0.0128974 (0.437%)	0.0128871 (0.357%)
	fine	0.0130321	0.0132904 (1.98%)	0.0131580 (0.967%)	0.0131364 (0.800%)

6.1.2 LCO prediction

After verifying the accuracy of the time-spectral CFD solver with a deforming mesh, we use it to predict LCO using the algorithm as described in Section 3.2. We predict multiple LCOs under different motion magnitudes, following the case setup of Thomas et al. [137]. The detailed settings

Table 6.4: Maximum C_l and C_m predicted by time-accurate and time-spectral method using viscous flow model.

Mesh	Property	TA ($N = 256$)	TS 1	TS 2	TS 3
C_l	coarse	0.102349	0.102407 (0.0563%)	0.102470 (1.19%)	0.102334 (−0.0143%)
	medium	0.103062	0.103194 (0.128%)	0.103115 (0.0515%)	0.102967 (−0.0919%)
	fine	0.104499	0.104680 (0.173%)	0.104593 (0.0901%)	0.104426 (−0.0701%)
C_m	coarse	0.0116326	0.0119574 (2.79%)	0.0118772 (2.10%)	0.0118085 (1.51%)
	medium	0.0118100	0.0120467 (2.00%)	0.0119127 (0.870%)	0.0118733 (0.536%)
	fine	0.0120191	0.0122853 (2.21%)	0.0121452 (1.05%)	0.0120940 (0.622%)

Table 6.5: Simulation time (wall-time) (sec) by time-accurate and time-spectral method using inviscid flow model.

Mesh	TA	TS 1	TS 2	TS 3
coarse	351.620	4.916	11.662	22.876
medium	498.816	22.313	38.828	58.434
fine	1001.420	20.319	49.683	88.728

are given in Table 6.7.

The Mach number is 0.8 and the nondimensional location of airfoil elastic axis a is -0.6. The mean angle of attack α_m is zero. The way we set the boundary condition based on the current estimation of V_f is described in Appendix C. The $|\alpha_1|$ versus V_f and ω versus V_f plots are shown in Fig. 6.7 and Fig. 6.8 for the inviscid and viscous case, respectively.

Table 6.6: Simulation time (wall-time) (sec) by time-accurate and time-spectral method using viscous flow model.

Mesh	TA	TS 1	TS 2	TS 3
coarse	385.156	6.970	29.302	30.834
medium	1113.613	20.335	67.806	132.095
fine	4287.730	325.443	736.514	1658.282

Table 6.7: Airfoil structural properties for LCO prediction [137].

Parameter	Expression	Value
Static unbalance	$x_\alpha = S_\alpha / mb$	0.25
Radius of gyration (squared)	$r_\alpha^2 = I_\alpha / mb^2$	0.75
Plunging natural frequency [rad/s]	ω_h	50.0
Pitching natural frequency [rad/s]	ω_α	100.0
Frequency ratio	ω_h / ω_α	0.5
Mass ratio	$\mu = m / \pi \rho_\infty b^2$	75.0

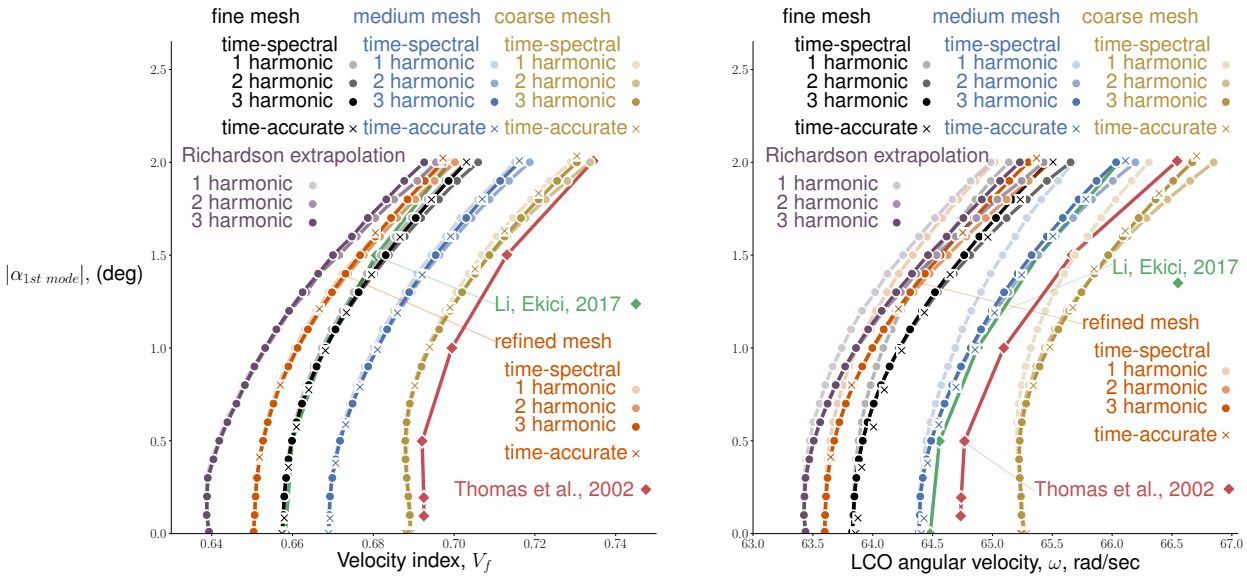


Figure 6.7: LCO responses under various V_f at $M = 0.8$ with time-spectral and time-accurate methods using an inviscid flow model. The results from Li and Ekici [70], Thomas et al. [137] are also included.

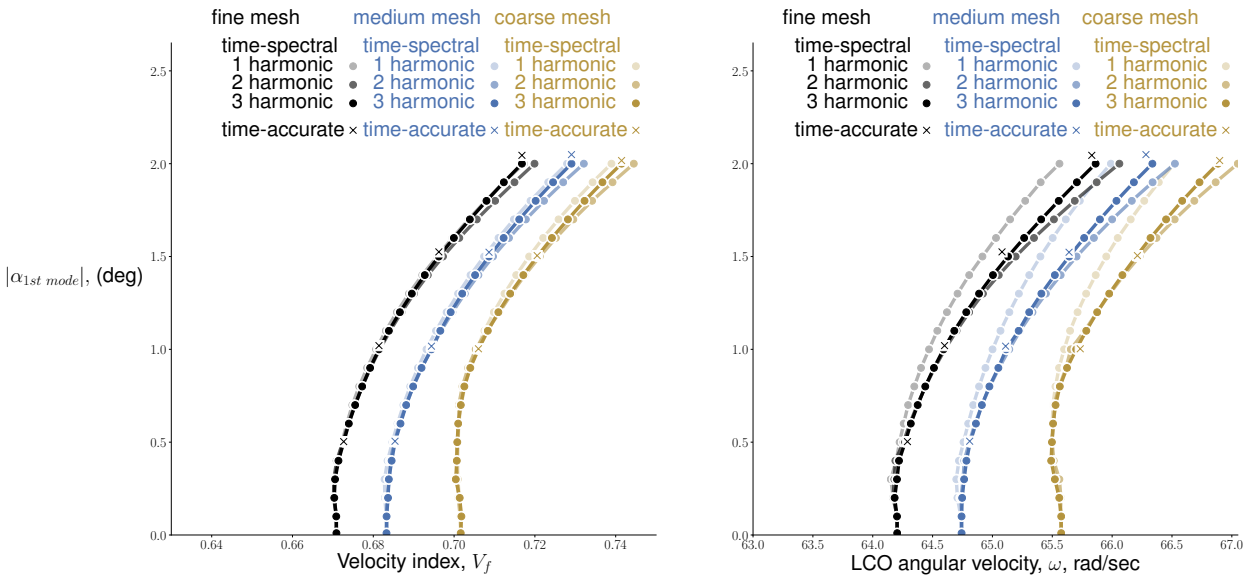


Figure 6.8: LCO responses under various V_f at $M = 0.8$ with time-spectral and time-accurate methods using a viscous flow model.

For verification purposes, time-accurate aeroelastic LCO solutions are generated for each of the three meshes. A Newmark-beta scheme is employed to integrate the structural equation of motion in time. For each time step, the CFD residual is reduced by 5×10^{-2} using a diagonalized alternating direction implicit iterations (DADI). The maximum number of DADI iterations is set to 30. The Courant—Friedrichs—Lewy (CFL) number is set to be 6.0 and 1.5 for inviscid and viscous cases, respectively. For each velocity index, the time-accurate aeroelastic solution is executed sufficiently long to obtain a converged LCO solution. The time step is set to be 0.0008 s, giving over 100 points for each period. LCO solutions with a very low magnitude, i.e., at velocities immediately past the flutter point, require a large number of time steps to reach a neutral response. Solutions with a larger magnitude converge quicker.

We observe that the time-spectral results (with three harmonics) compare well with the time-accurate results (less than 1% relative error for all cases). At the largest magnitude, $|\alpha_1| = 2°$, a model with three harmonics agrees better to the time-accurate solution compared with a model with

two harmonics, which in turn agrees better than that with one harmonic. As the prescribed pitching magnitude decreases, the difference between those three curves reduces, eventually ending at the flutter point. Thus, to predict the flutter speed, one harmonic is sufficient. However, to predict a finite amplitude LCO, two or more harmonics are necessary.

We also add a refined mesh (768×256 cells) result for the inviscid mesh. This was done to obtain a spatially independent mesh. A Richardson extrapolation is then conducted based on results of the medium, fine and refined meshes. The relative error between the Richardson extrapolation and refined mesh is about 1.5%. A similar study was attempted for the viscous case using the refined mesh with 1088×256 cells, but we had difficulty to converge the aeroelastic residual. The refined mesh RANS cases were only partially converged as we experienced convergence issues with the linear solution inside the Newton iteration. It might be caused by the fact that the preconditioner is not accurate enough. This can likely be addressed using a stronger, more accurate preconditioner for the linear system.

The mesh density has an impact on whether the LCO response is supercritical or subcritical. From the V_f-$|\alpha_1|$ plot in Fig. 6.7, for the medium and fine mesh, a supercritical response is observed. However, for the coarse mesh, the response is subcritical. Thus, to determine whether an LCO is supercritical or subcritical, we recommend: (1) use a mesh as fine as possible, (2) conduct a mesh convergence study. Otherwise, we may arrive at an opposite conclusion as in this case.

One additional advantage of the time-spectral method over the time-accurate method is that it can trace the unstable branch of a subcritical response (refer to the dashed line from Fig. 1.2), and the time-accurate can not. With the unstable branch captured, we can determine whether the airfoil has a supercritical response or subcritical response. Additionally, we can parametrize the curve and optimize its shape. This is evident from the time-accurate coarse mesh solution shown in Fig. 6.7. All time-accurate solutions with a velocity index less than the flutter point (approaching from the left) follow the stable branch and result in a steady-state response. Once the velocity is increased past the flutter point, the time-accurate solution demonstrates a rapid increase in magnitude and

eventually reaches the stable branch. This is why one time-accurate solution appears to have zero amplitude before the flutter point.

Finally, for each level of meshes, the RANS results have higher flutter velocity indices compared to the Euler results. It is as expected from Fig. 1.1. And a similar trend for the flutter boundary is observed in Section 6.1.3.

6.1.2.1 Step size convergence study

We demonstrate the LCO responses under three step-sizes for the medium mesh for inviscid and viscous cases. The results are shown in Figs. 6.9 and 6.10 for inviscid and viscous cases, respectively.

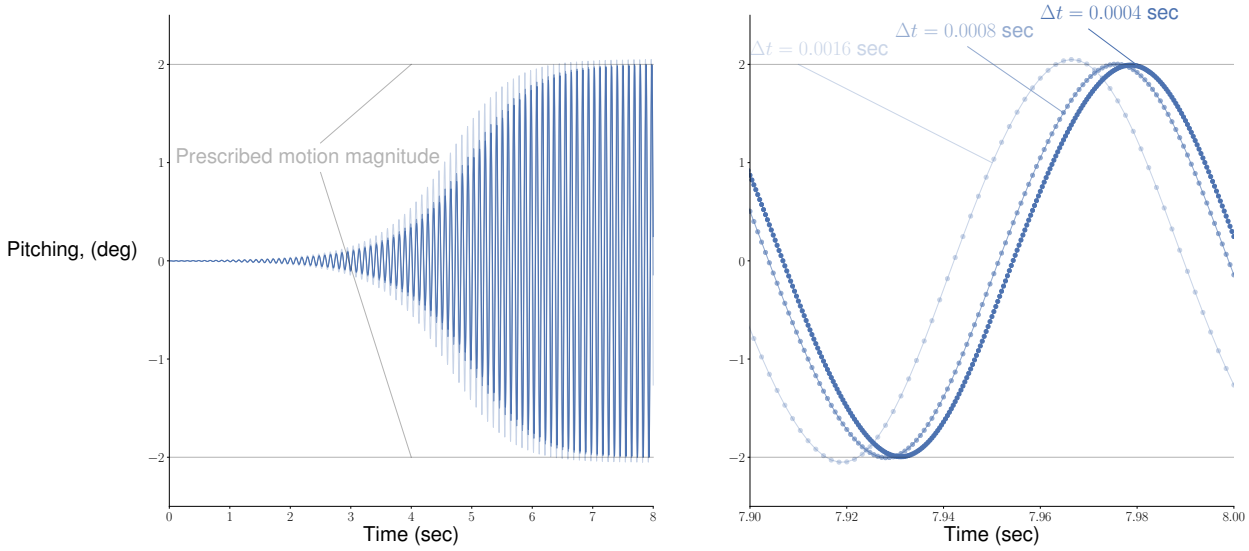


Figure 6.9: Time-accurate LCO responses under $V_f = 0.716$ at $M = 0.8$ for medium mesh with an inviscid flow model. The reference prescribed pitching magnitude for time-spectral method with 7 time instances is $2°$.

For the inviscid case, the relative error of the pitching magnitude between $\Delta t = 0.0004$ sec and $\Delta t = 0.0008$ sec and between $\Delta t = 0.0008$ sec and $\Delta t = 0.0016$ are 0.48% and 2.36%,

respectively. The result with $\Delta t = 0.0004$ sec is close to being step-size independent. Using 36 cores, the computational time for $\Delta t = 0.0016, 0.0008$, and 0.0004 sec are 555 sec, 1114 sec, and 2275 sec for one flow condition at $V_f = 0.716$. Using time-spectral method, the LCO curve (with 21 flow conditions) with medium mesh under 3, 5, and 7 time instances take 834 sec, 1912 sec, and 2763 sec (with the warm start time included, see Section 3.2), respectively.

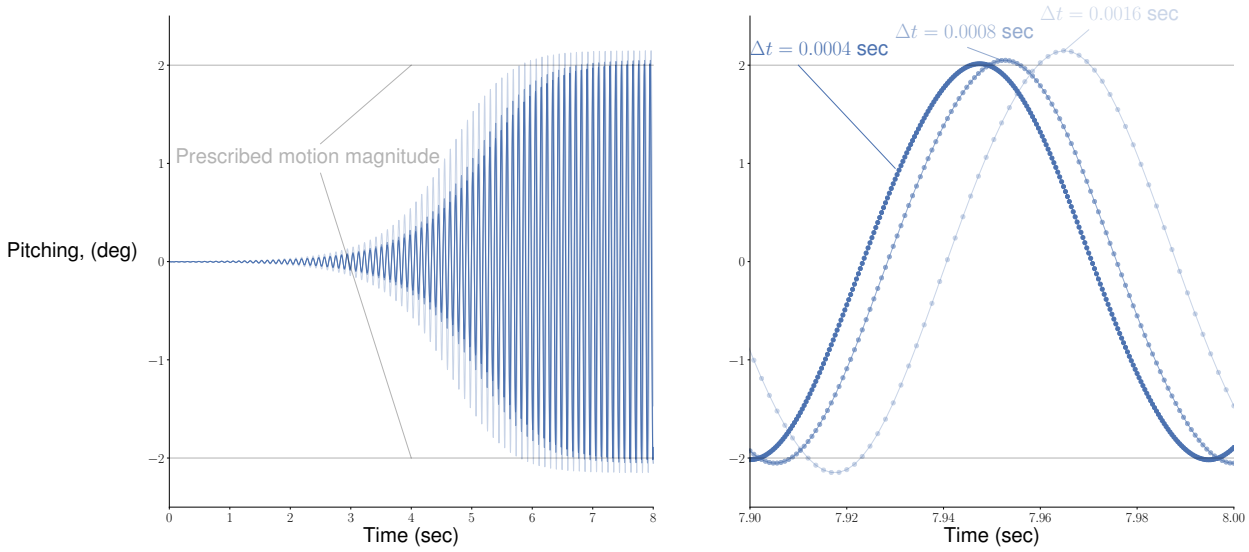


Figure 6.10: Time-accurate LCO responses under $V_f = 0.729$ at $M = 0.8$ for medium mesh with a viscous flow model. The reference prescribed pitching magnitude for time-spectral method with 7 time instances is 2°.

For the viscous case the relative error of the pitching magnitude between $\Delta t = 0.0004$ sec and $\Delta t = 0.0008$ sec and between $\Delta t = 0.0008$ sec and $\Delta t = 0.0016$ are 1.86% and 4.65%, respectively. The result with $\Delta t = 0.0004$ sec is close to being step-size independent, but an even finer mesh could be added to further reduce the error. Using 36 cores, the computational time for $\Delta t = 0.0016, 0.0008$, and 0.0004 sec are 1238 sec, 2533 sec, and 5187 sec for one flow condition at $V_f = 0.729$. Using time-spectral method, the LCO curve (with 21 flow conditions) with medium mesh under 3, 5, and 7 time instances take 1022 sec, 3501 sec, and 5978 sec (with the warm start time included, see Section 3.2), respectively.

6.1.2.2 GCL

Here we compare the LCO result with and without GCL. The results are shown in Figs. 6.11 and 6.12 for the inviscid and viscous cases, respectively. The results presented here are obtained using the medium mesh.

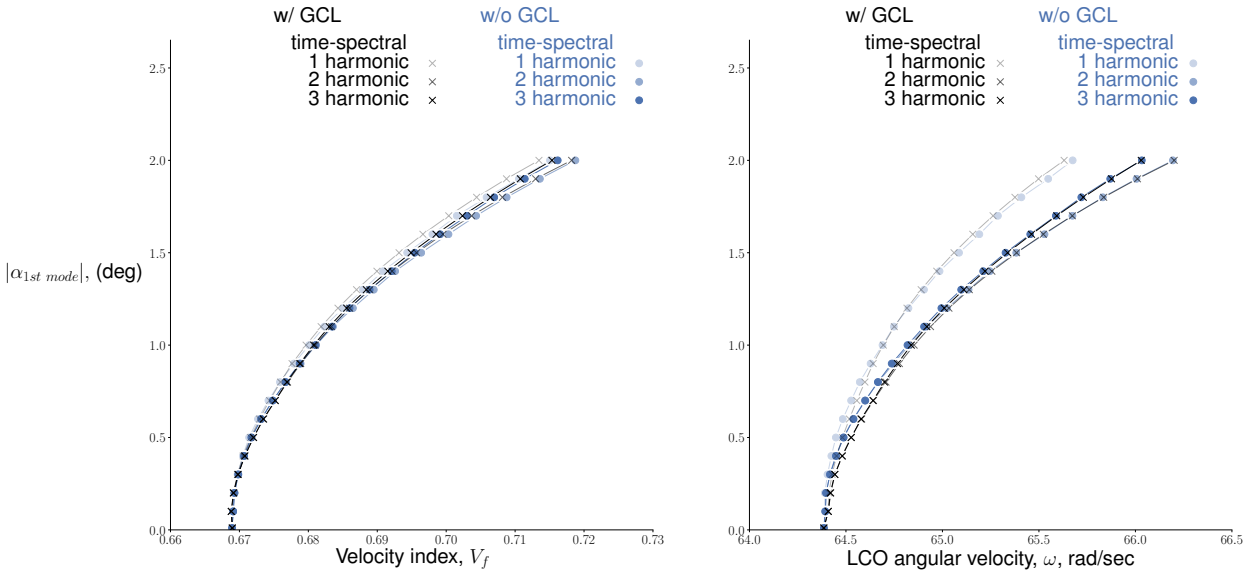


Figure 6.11: LCO with inviscid flow model with and without GCL.

For the inviscid case, the velocity index predicted with GCL is slightly lower than without GCL. As the prescribed motion magnitude decreases, the difference between the two reduces. The maximum relative error of V_f using one, two, and three harmonics are $0.25\%, 0.11\%$, and 0.12%, respectively. For the angular velocity prediction, when the prescribed motion magnitude is within 0° to 1°, the angular velocity predicted with GCL is higher than without GCL. With only one harmonic, the angular velocity predicted is lower with GCL than without GCL for the prescribed motion magnitude between 1° and 2°. For two or three harmonics the predicted values, with or without GCL, are close to identical between 1° and 2°. The maximum relative error of ω using one, two, and three harmonics are $0.077\%, 0.068\%$, and 0.062%, respectively.

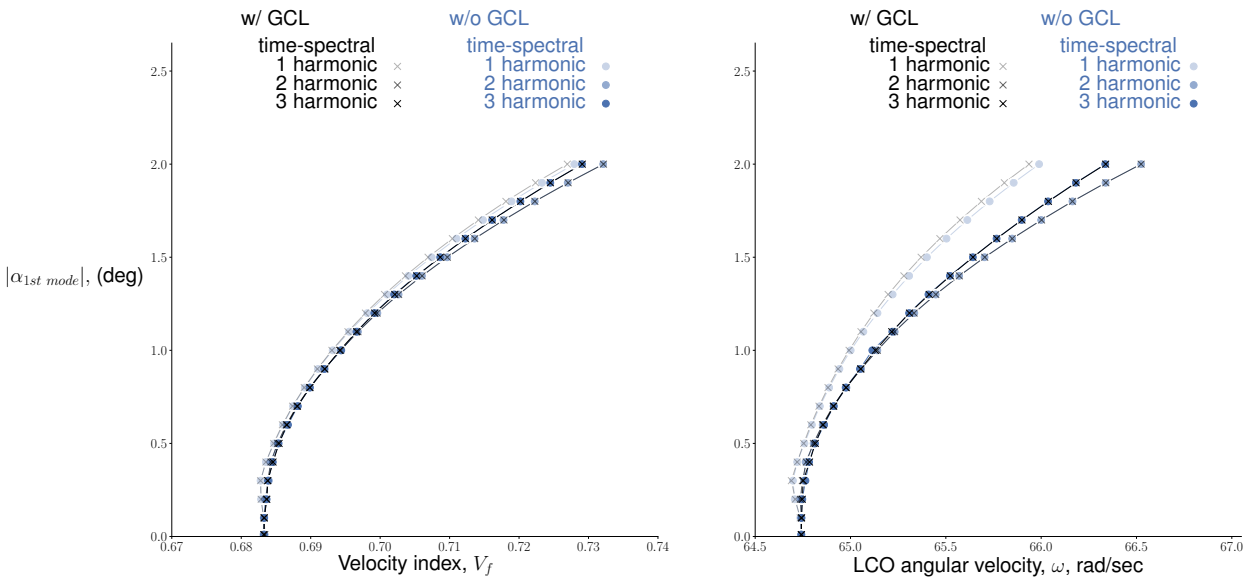


Figure 6.12: LCO with viscous flow model with and without GCL.

For the viscous case, the velocity index and angular velocity predicted with GCL is lower than without GCL when considering only one harmonic. For two or three harmonics the results, the curves with or without GCL, almost overlap with each other. The maximum relative error of V_f using one, two, and three harmonics are $0.14\%, 0.029\%$, and 0.030%, respectively. And the maximum relative error of ω using one, two, and three harmonics are $0.081\%, 0.012\%$, and 0.025%, respectively.

For inviscid and viscous cases, within the motion magnitude range considered in the paper, the introduction of GCL affects the LCO results at most by 0.25%, both for the velocity index and angular velocity predictions. Thus, we use models without GCL in the rest of the paper.

6.1.3 Flutter boundary prediction

In this section, we use the algorithm described in Section 3.2 to compute the flutter boundary. We demonstrate our method by showing its capability to accurately capture the transonic dip described in Section 1.2 for the Isogai aeroelastic benchmark case [53], whose structural properties

Table 6.8: Airfoil structural properties of the Isogai case [53]

Parameter	Expression	Value
Static unbalance	$x_\alpha = S_\alpha/mb$	1.8
Radius of gyration	$r_\alpha^2 = I_\alpha/mb^2$	3.48
Plunging natural frequency	ω_h	100.0
Pitching natural frequency	ω_α	100.0
Frequency ratio	ω_h/ω_α	1.0
Mass ratio	$\mu = m/\pi\rho_\infty b^2$	60.0

are listed in Table 6.8.

The freestream angle-of-attack is set to zero and the prescribed motion magnitude is set to 0.1°. We choose this motion magnitude because if we decrease it further, the convergence rate becomes lower. We use one harmonic (three time-instances) for this simulation, a choice that is based on the results in Fig. 6.7—as the LCO magnitude is reduced, the impact of increasing the number of time instances decreases. The flutter boundary for the time-spectral method is sequentially determined starting from $M = 0.7$. Each higher Mach number solution is solved with the neighboring lower Mach number solution as an initial guess using the strategy as introduced in Section 3.2. We consider three levels of mesh sizes. The results are shown in Fig. 6.13.

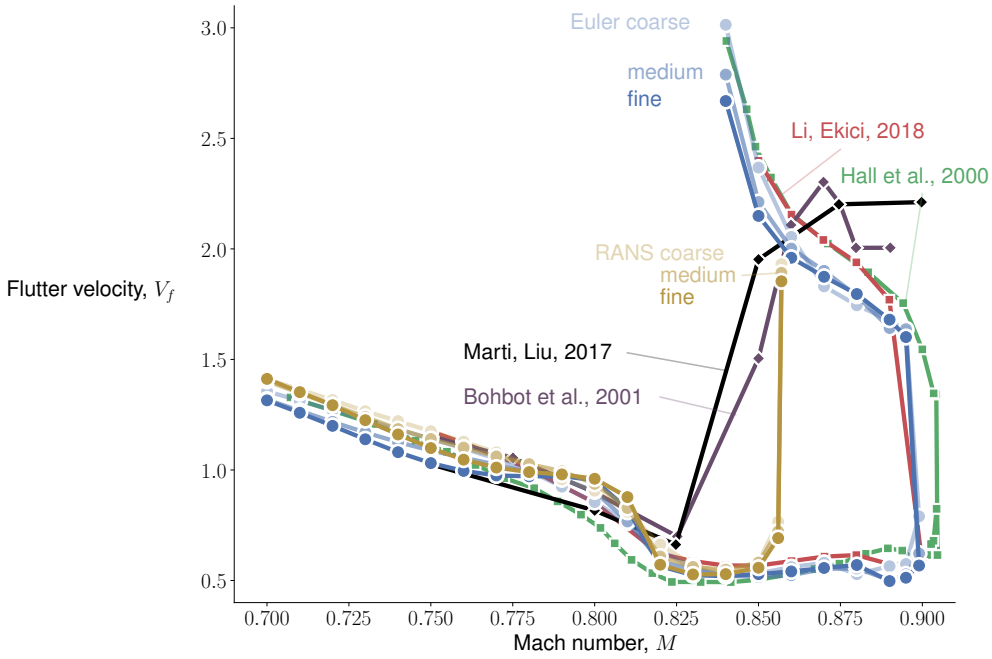


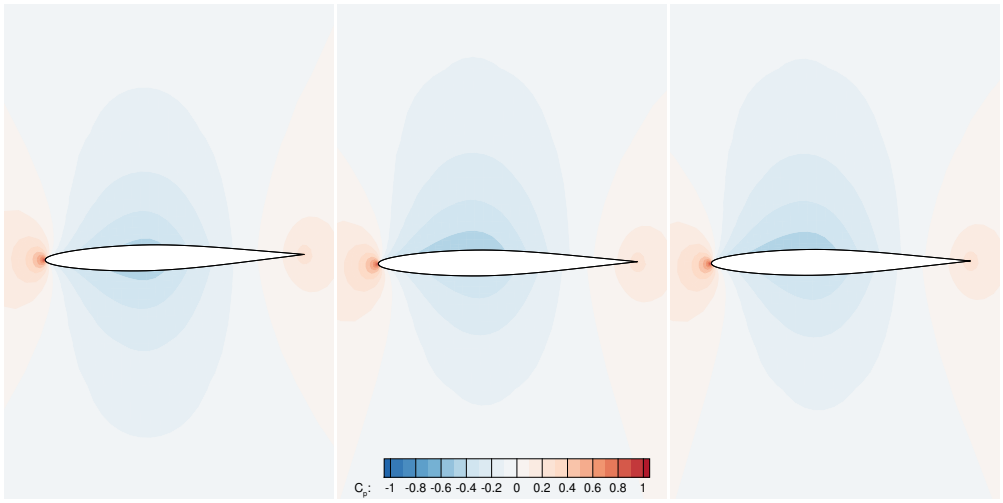
Figure 6.13: NACA 64A010 flutter boundary compared with Euler results from Li and Ekici [71] and Hall et al. [43] and RANS results (with SA turbulence model) Bohbot et al. [14] and Marti and Liu [94]

From Fig. 6.13, we see that the proposed CNK solver finds the transonic dip. The computed inviscid solution matches results published in the literature, especially in the transonic dip region. Within the upper branch, in the range of $M \in [0.83, 0.9]$, our method tends to underpredict the flutter boundary. Around $M = 0.8$, our method tends to overpredict the flutter boundary compared with other methods. This may be partly due to the different meshes used in different papers. The viscous solution shows a more pronounced dip in the transonic regime than the reference time-accurate solution of the RANS equations with SA turbulence model by Bohbot et al. [14], Marti and Liu [94]. The reference solution by Bohbot et al. [14], Marti and Liu [94] are relatively sparsely sampled in the critical range, $M = 0.825 - 0.85$, making direct comparison difficult. For Mach numbers beyond 0.86, our solver has difficulty converging the residual. To extend the solution to higher Mach numbers is part of our future work plans.

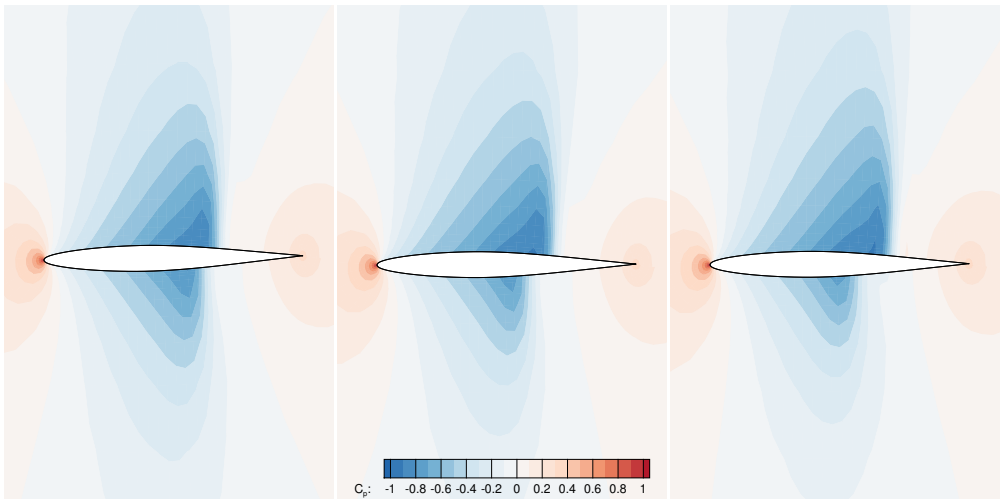
6.1.4 Preconditioner performance study

We now compare the performance of different preconditioners presented in Section 3.3. For the inviscid case, we show the convergence history for the $M = 0.83$ flutter case with an initial guess at $M = 0.7$. The prescribed pitching motion magnitude is set to $0.5°$. We use the medium mesh with 192×64 cells and three time-instances.

The viscous case proves to be more challenging to converge. For instance, we cannot solve for the state variables at $M = 0.83$ using the solution at $M = 0.7$ in a similar manner as the inviscid case. Instead, we present the convergence history solving the equations at $M = 0.72$ using the solution at $M = 0.7$ as the initial guess. We tried several Mach numbers between 0.72 and 0.83, but they failed to converge. We use $0.5°$ as the prescribed pitching motion magnitude and the coarse mesh (136×32 cells) in this study. Three time-instances are used.



(a) Snapshots at $M = 0.7$



(b) Snapshots at $M = 0.83$

Figure 6.14: Solutions at $M = 0.7$ and $M = 0.83$ with medium mesh 192×64 using Euler model. The former solution is used to warm-start the latter solution.

For the inviscid flow case, the initial guess and the final solution flow field snapshots are shown in Fig. 6.14(a) and Fig. 6.14(b), respectively. The initial guess is a subsonic case without any shock waves, but the final solution does exhibit shock waves. The initial flutter speed index is 1.32, and the final solution is 0.56. The relative difference of the flutter speed index is 58%. For the viscous

flow case, the initial flutter boundary is 1.42, and the final solution is 1.30. It is reduced by 8.45%.

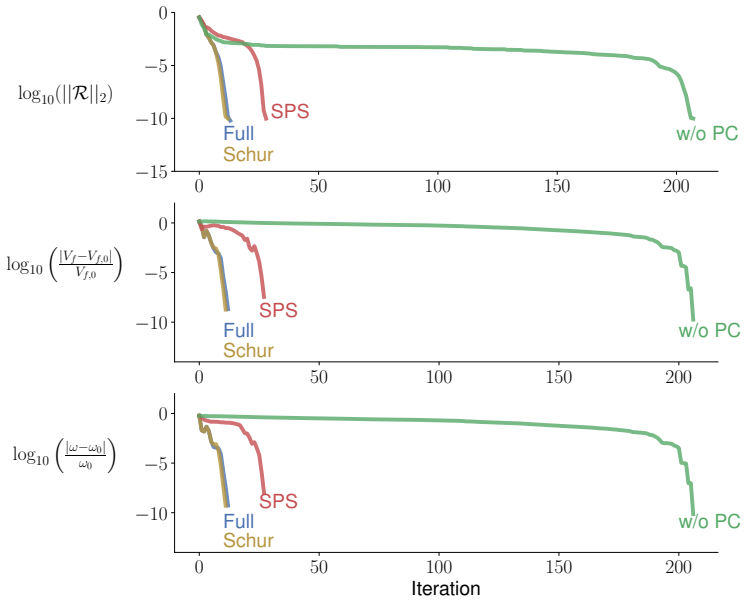


Figure 6.15: Convergence history for solving the $M = 0.83$ flutter boundary with an initialization of the $M = 0.7$ Euler solution.

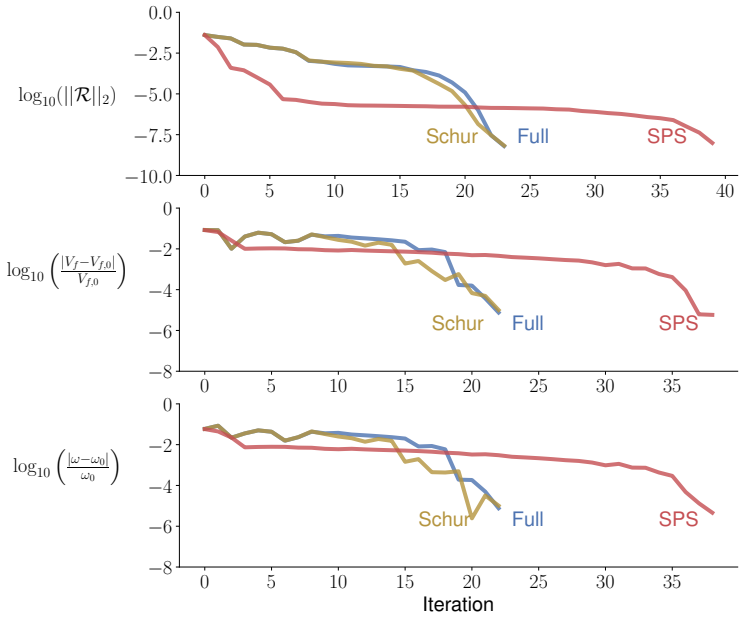


Figure 6.16: Convergence history for solving the $M = 0.72$ flutter boundary with an initialization of the $M = 0.7$ RANS solution.

The convergence history for the inviscid flow model is shown in Fig. 6.15. With full preconditioner (by direct inversion) or with Schur preconditioner, the residual is reduced by about 10 orders in around 15 steps. Using 4 cores, the direct inversion and Schur preconditioner takes about 95.18 sec and 98.33 sec, respectively. For the SPS preconditioner, more iterations are required (30 steps) using 300.21 sec. However, if no preconditioner is used, it takes more than 200 steps to converge the simulation in about 3011.27 sec. The convergence history for the viscous flow model is shown in Fig. 6.16. We find that if no preconditioner is applied the residual will not converge. Using a full or Schur preconditioner will give a similar performance with a wall-time of 54.30 and 65.99 sec, respectively. The SPS preconditioner performs the worst requiring close to double the number of iterations with a wall-time of 261.11 sec.

For our current problem, Schur and direct inversion are the best performing preconditioners. The computational costs are similar because the preconditioning matrices computed by these two

methods are identical. The computational costs forming these preconditioners are negligible compared with the CFD preconditioning. The difference in the convergence history may be attributed to the fact that the PETSc performance is not deterministic. Using the SPS preconditioner, or simply not using any preconditioner, will increase the solution computational cost or worse, not converge. However, for a more complex FEM model, there may be a trade-off between the SPS and Schur, but that remains to be explored.

6.2 Wing results

6.2.1 Model description

6.2.1.1 Structural model

The model set up is based on the "weakened model 3" from Advisory Group for Aerospace Research and Development (AGARD) report by Yates [147] At first, we conduct conversion of the dimensionless structural mode. In the AGARD 445.6 case, the modes are given as displacements at points. The generalized mass matrix, $\tilde{\mathbf{\Phi}}^{\intercal}\mathbf{M}\tilde{\mathbf{\Phi}}$, in the original AGARD report is normalized to give unit mass in English units ($\text{lbf} \cdot \text{sec}^2 \cdot \text{in}^{-1}$). Here the original matrix can be written in SI units

as,

$$\begin{aligned}
\tilde{\mathbf{\Phi}}^{\intercal}\mathbf{M}\tilde{\mathbf{\Phi}} &= \begin{bmatrix} 1\text{lbf sec}^2\,\text{in}^{-1} & & \\ & \ddots & \\ & & 1\text{lbf sec}^2\,\text{in}^{-1} \end{bmatrix}, \\
&= \begin{bmatrix} \frac{1\text{slug ft}}{\text{sec}^2}\,\text{sec}^2\,\text{in}^{-1} & & \\ & \ddots & \\ & & \frac{1\text{slug ft}}{\text{sec}^2}\,\text{sec}^2\,\text{in}^{-1} \end{bmatrix}, \\
&= \begin{bmatrix} 12\text{slug} & & \\ & \ddots & \\ & & 12\text{slug} \end{bmatrix}, \\
&= \begin{bmatrix} 175.127\text{kg} & & \\ & \ddots & \\ & & 175.127\text{kg} \end{bmatrix}.
\end{aligned} \tag{6.1}$$

However, in this work, we require this to be dimensionless from such that.

$$\mathbf{\Phi}^{\intercal}\frac{\mathbf{M}}{m_0}\mathbf{\Phi} = \mathbf{I}. \tag{6.2}$$

We thus seek to find a scaling factor c such that,

$$(c\sqrt{m_0}\tilde{\mathbf{\Phi}}^{\intercal})\frac{\mathbf{M}}{m_0}(c\sqrt{m_0}\tilde{\mathbf{\Phi}}) = \mathbf{I}. \tag{6.3}$$

Thus by expanding

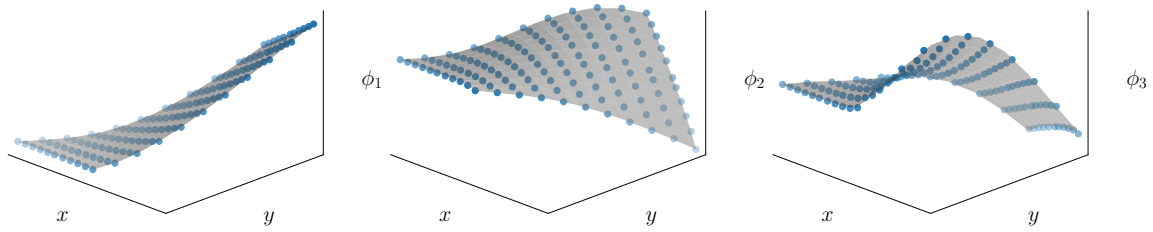
$$(c\sqrt{m_0}\tilde{\mathbf{\Phi}}^{\intercal})\frac{\mathbf{M}}{m_0}(c\sqrt{m_0}\tilde{\mathbf{\Phi}}) = c^2\tilde{\mathbf{\Phi}}^{\intercal}\mathbf{M}\tilde{\mathbf{\Phi}} = c^2\begin{bmatrix} 175.127\text{kg} & & \\ & \ddots & \\ & & 175.127\text{kg} \end{bmatrix} = \mathbf{I}. \tag{6.4}$$

we can find the coefficient to be $c = 1/\sqrt{175.127} = 0.075565$. Thus, to obtain a dimensionless form we use the following scaling,

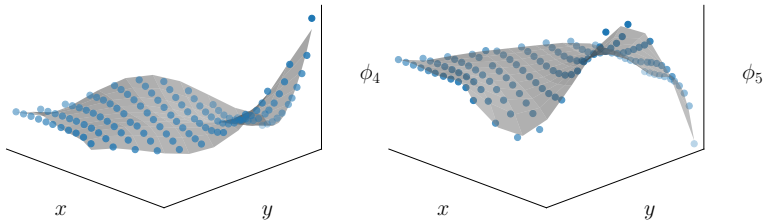
$$\mathbf{\Phi} = 0.075565\sqrt{m_0}\tilde{\mathbf{\Phi}}. \tag{6.5}$$

where m_0 is the initial weight of the wing.

The first 5 mode shapes $\mathbf{\Phi}$ are shown in Fig. 6.17. We use scikit-learn [109] to construct a 4th order polynomial approximation $\hat{\mathbf{\Phi}}$ for each structural mode. In Figure 6.17, $\hat{\mathbf{\Phi}}$ is shown as gray surfaces which demonstrate acceptable fit with respect to the structural mode shapes, $\mathbf{\Phi}$, shown as blue dots. We then use $\hat{\mathbf{\Phi}}$ to evaluate aerodynamic nodal displacements as given in Eq. (2.51).



(a) 1st mode (1st bending mode), $\omega_{r,1} = 60.31$ rad/sec **(b) 2nd mode (1st torsion mode), $\omega_{r,2} = 239.80$ rad/sec** **(c) 3rd mode (2nd bending mode), $\omega_{r,3} = 303.78$ rad/sec**



(d) 4th mode (2nd torsion mode), $\omega_{r,4} = 575.19$ rad/sec **(e) 5th mode (3rd bending mode), $\omega_{r,5} = 742.13$ rad/sec**

Figure 6.17: First 5 modes of AGARD 445.6 case weakened mode 3 [147]. The coordinates of the blue points are from the AGARD report. The gray surfaces are a polynomial regression of those blue points.

Note that in the AGARD report [147], a sixth mode is also included but is ignored here. This is because the sixth mode is a lateral motion (in-plane) mode and its z direction displacement is no longer the dominant motion, which is in contrary to our transfer class assumptions. Further, other work has indicated that the sixth mode is insignificant to flutter boundary prediction [73].

6.2.1.2 Aerodynamic model

We generate the geometry based on AGARD report [147] using the open-source package *pyLayout*[3] which is an inhouse built geometry engine. The wing planform is shown in Fig. 6.18 and the detailed geometry parameters are given in Table 6.9. The wing airfoil cross section is a NACA 65A004.

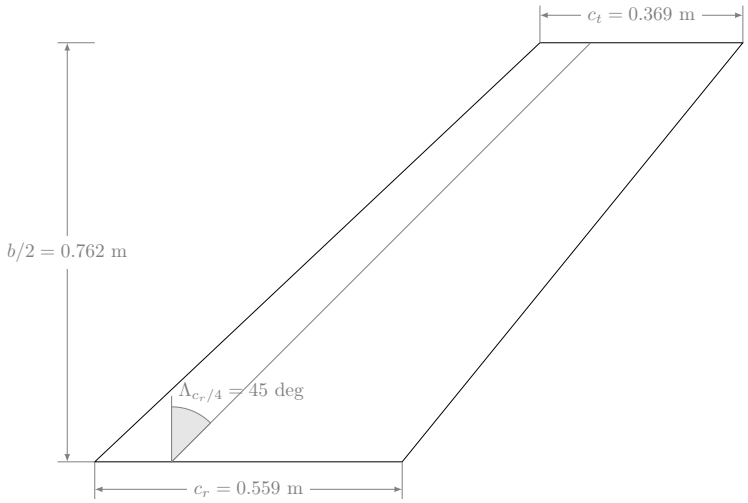


Figure 6.18: Geometry of AGARD 445.6 case

The surface mesh is generated by ICEM [31]. We then apply the open-source package *pyHyp*[4], an inhouse hyperbolic mesh generator, to generate the volume mesh from the surface mesh. The mesh we use for the work is a "O" mesh as shown in Fig. 6.19. For the inviscid mesh, there are 186 mesh points around the airfoil, 128 mesh points orthogonal to the surface of the wing, and 49

[3] https://github.com/mdolab/pylayout.git
[4] https://github.com/mdolab/pyhyp.git

Table 6.9: AGARD 445.6 wing geometric properties

Description	Symbol	Value	Unit
Sweep	$\Lambda_{c_r/4}$	45	deg
Aspect ratio	AR	1.65	-
Taper ratio	λ	0.66	-
Semi span	$b/2$	0.762	m
Root chord	c_r	0.559	m
Tip chord	c_t	0.369	m
Area	A	0.353	m^2

mesh points in the spanwise direction. The thickness of the first layer of mesh around the wing is 10^{-3} m. For the viscous mesh, the mesh has the same topology. But the thickness of the first layer of mesh is 10^{-6} m which is much smaller compared with the inviscid mesh to resolve the boundary layer. For the viscous mesh, the last 1% of the wing has been chopped off and the rest of the wing has been scaled up to give the same chord length. The tip of the wing is rounded which makes it easier for the solver to reduce the residual.

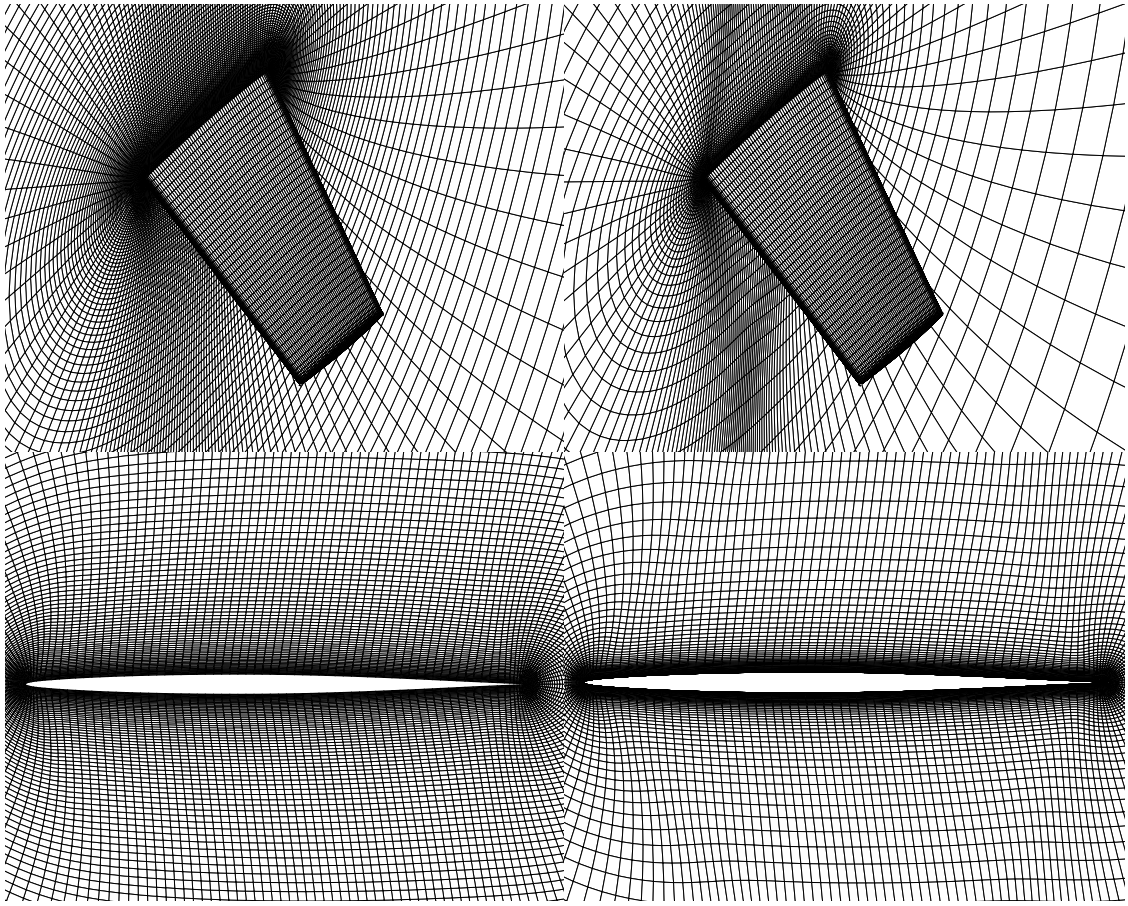


Figure 6.19: CFD mesh used in this study. Inviscid mesh shown in the left, and viscous grid shown in the right.

6.2.2 Flutter boundary results

In this section, we compute the flutter boundary of the proposed method and compare it to experimental and other CFD results. The Mach numbers chosen here to compute the flutter boundary are the same as used in the AGARD report [147]. For each Mach number, the flow density is given in Table 6.10. With a V_f given, the triplet (T_∞, p_∞, M) is fully determined that is described earlier in Section 2.3.5.

The flutter boundary is computed in the following manner. At first, we compute the flutter onset

Table 6.10: Density for each point from the flutter boundary [147]

M	density (slug/ft^3)
0.499	0.000830
0.678	0.000404
0.901	0.000193
0.954	0.000123
1.072	0.000107
1.141	0.000152

velocity at $M = 0.499$. We then use this result to initialize the neighboring states for $M = 0.654$ as we expect the solution to be close. We continue with this initialization strategy and then obtain the whole flutter boundary shown in Fig. 6.20. The transonic dip due to the nonlinear dynamics around $M = 0.954$ is captured. Three time-instances are considered for the flutter boundary computations.

In the figure, we compare current result using different numbers of mode shapes with experimental results by Yates [147] and numerical results by Li and Ekici [74] who solved the Euler and RANS equations with a harmonic balance approach. Our results are consistent with results by Li and Ekici [74]. The results match especially well at the Euler transonic dip. We also observe that both simulation results cannot make a prediction close enough to experimental results. the flutter boundary at $M = 1.072, 1.141$ very accurately. However, the viscous results are much more closer to the experimental results compared with the inviscid results indicating that viscosity is playing a critical role for these two cases. This is an indication of the importance of high-fidelity models.

Our solutions using an inviscid flow model are shown in Figs. 6.21 to 6.23 at three different Mach numbers, $M = 0.499$, 0.954, and 1.141. Each of these figures contains snapshots of the pressure coefficient (C_p) distribution from three different time instances. At $M = 0.499$, the flow speeds up in the mid chord and slows down at the tip. At $M = 0.954$, right at the transonic dip, we can see a shock wave forming at the trailing edge. And at $M = 1.141$, there is one shock wave ahead of the leading edge of the wing and there is another shock wave formed along the trailing edge similar to that of $M = 0.954$. From the figures, we can see that the prescribed motion is

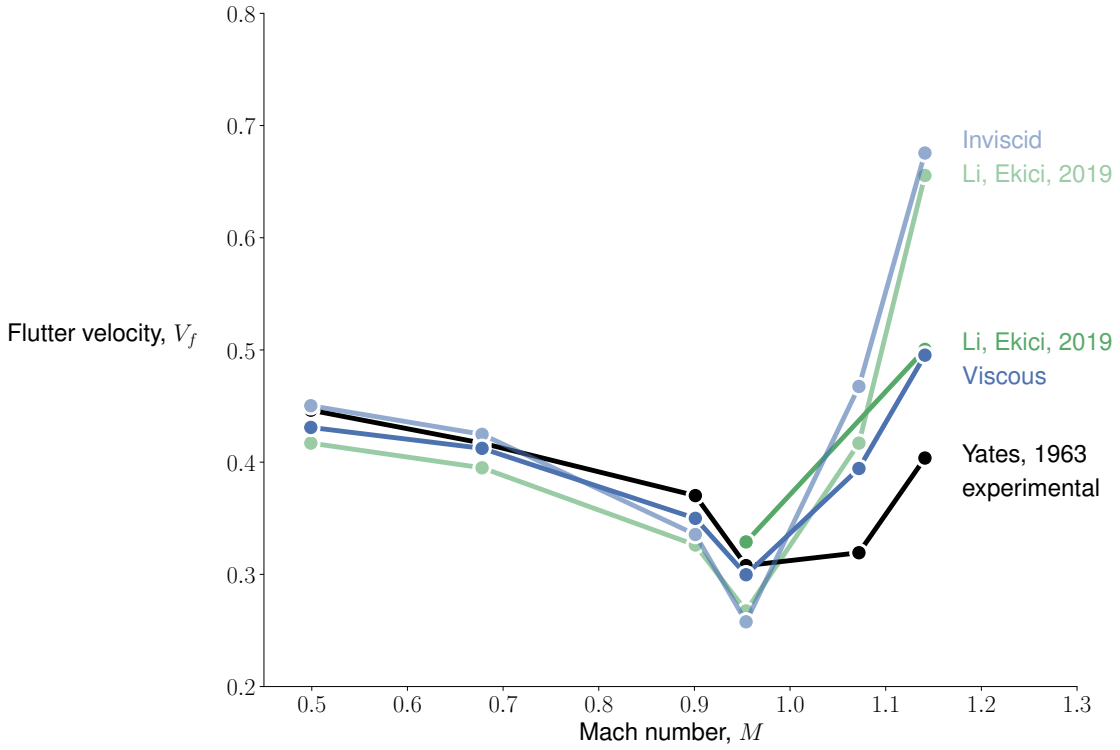


Figure 6.20: AGARD 445.6 flutter boundary with different structural modes considered. Current results match better with numerical results by [74] than with experimental results by Yates [147].

insignificant as expected.

Our solutions using a viscous flow model are shown in Figs. 6.24 to 6.26 at three different Mach numbers, $M = 0.499$, 0.954, and 1.141. The flow fields look similar to those from inviscid simulations. The mesh used for this study is coarsened by a factor of 2 in all directions.

6.2.3 LCO results

The proposed method can also be used to predict the LCO behavior using larger prescribed motion amplitudes. We studied the LCO responses of the wing under different Mach numbers using Euler and RANS flow models under different prescribed motion magnitude as shown in Fig. 6.27. The same flow conditions are used here as those used in the flutter boundary prediction Table 6.10. Different from the flutter boundary prediction, we consider five time-instances to

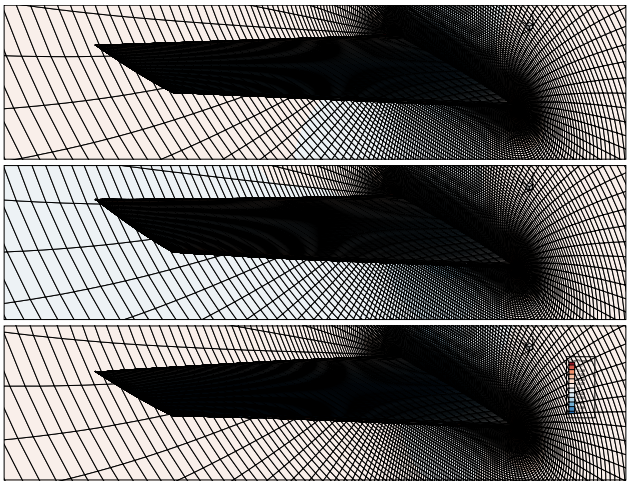


Figure 6.21: C_p distribution at $M = 0.499$ with an inviscid flow model. From top to bottom are three different time instances at the flutter point.

capture additional higher frequency components induced by a larger prescribed motion magnitude.

The LCO responses are shown in Fig. 6.28. It is observed that most LCOs are subcritical for the Euler result besides the one at $M = 1.141$ which has a supercritical response for prescribed motion magnitude below 0.06 after that the response is becoming subcritical. Most LCOs with RANS flow model also have slightly subcritical responses. And similar to the Euler results, the LCO at $M = 1.141$ seems also to be initially supercritical and then transit to subcritical responses with an increased prescribed motion magnitude. The Euler mesh has been coarsened once, and the RANS mesh has been coarsened twice for the study.

The flow field at $M = 0.954$ with prescribed motion magnitude $\eta_{\text{1 st mode}} = 0.1$ is shown in Fig. 6.29. The formation and dissipation of the shock wave are shown in this case. This demonstrates that the proposed method can capture flow nonlinearity caused by shock wave motions.

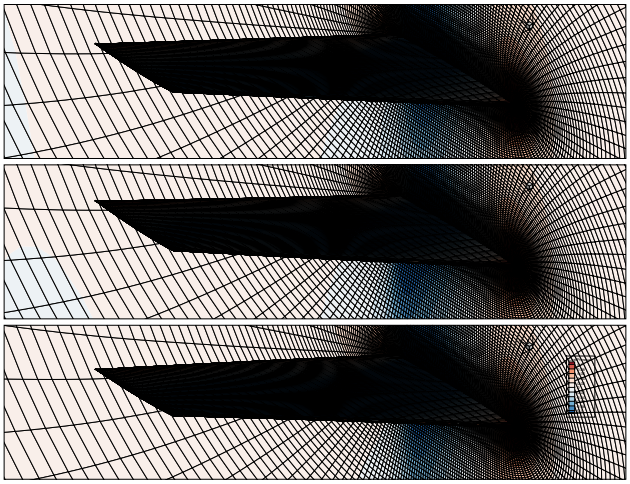


Figure 6.22: C_p distribution at $M = 0.954$ with an inviscid flow model. From top to bottom are three different time instances at the flutter point.

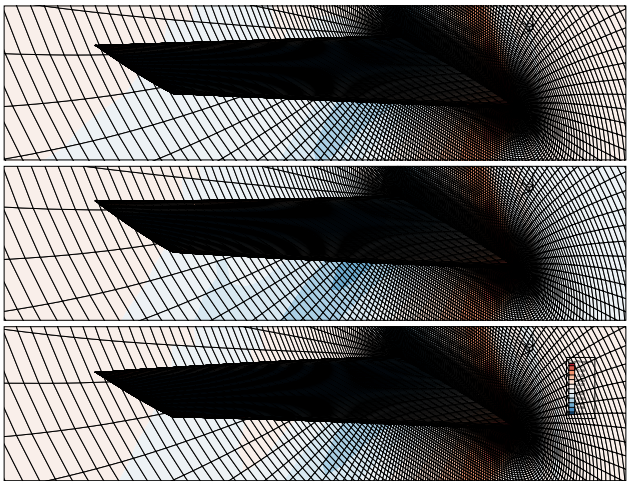


Figure 6.23: C_p distribution at $M = 1.141$ with an inviscid flow model. From top to bottom are three different time instances at the flutter point.

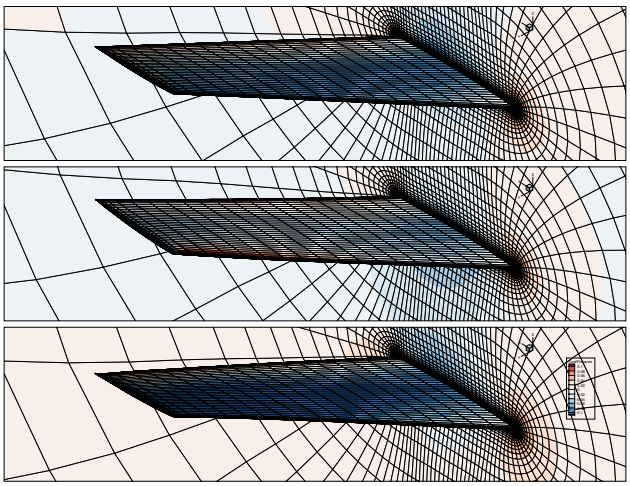


Figure 6.24: C_p distribution at $M = 0.499$ with a viscous flow model. From top to bottom are three different time instances at the flutter point.

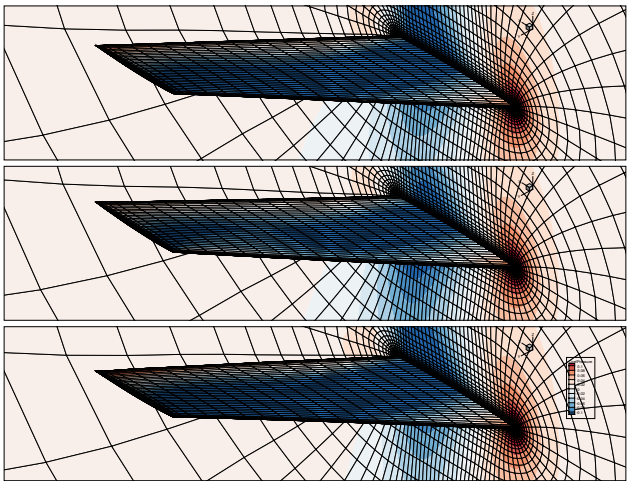


Figure 6.25: C_p distribution at $M = 0.954$ with a viscous flow model. From top to bottom are three different time instances at the flutter point.

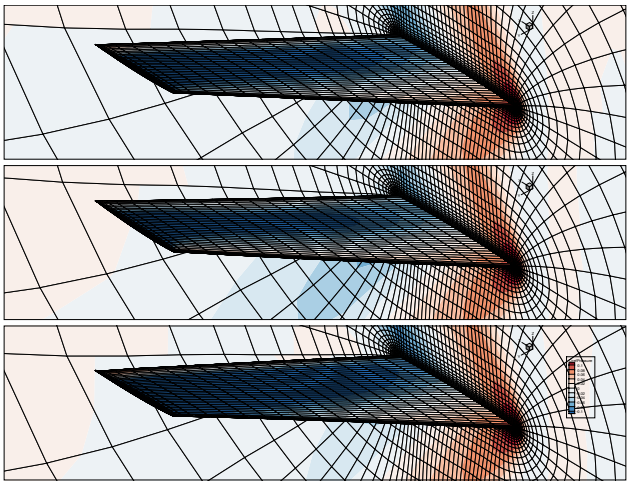


Figure 6.26: C_p distribution at $M = 1.141$ with a viscous flow model. From top to bottom are three different time instances at the flutter point.

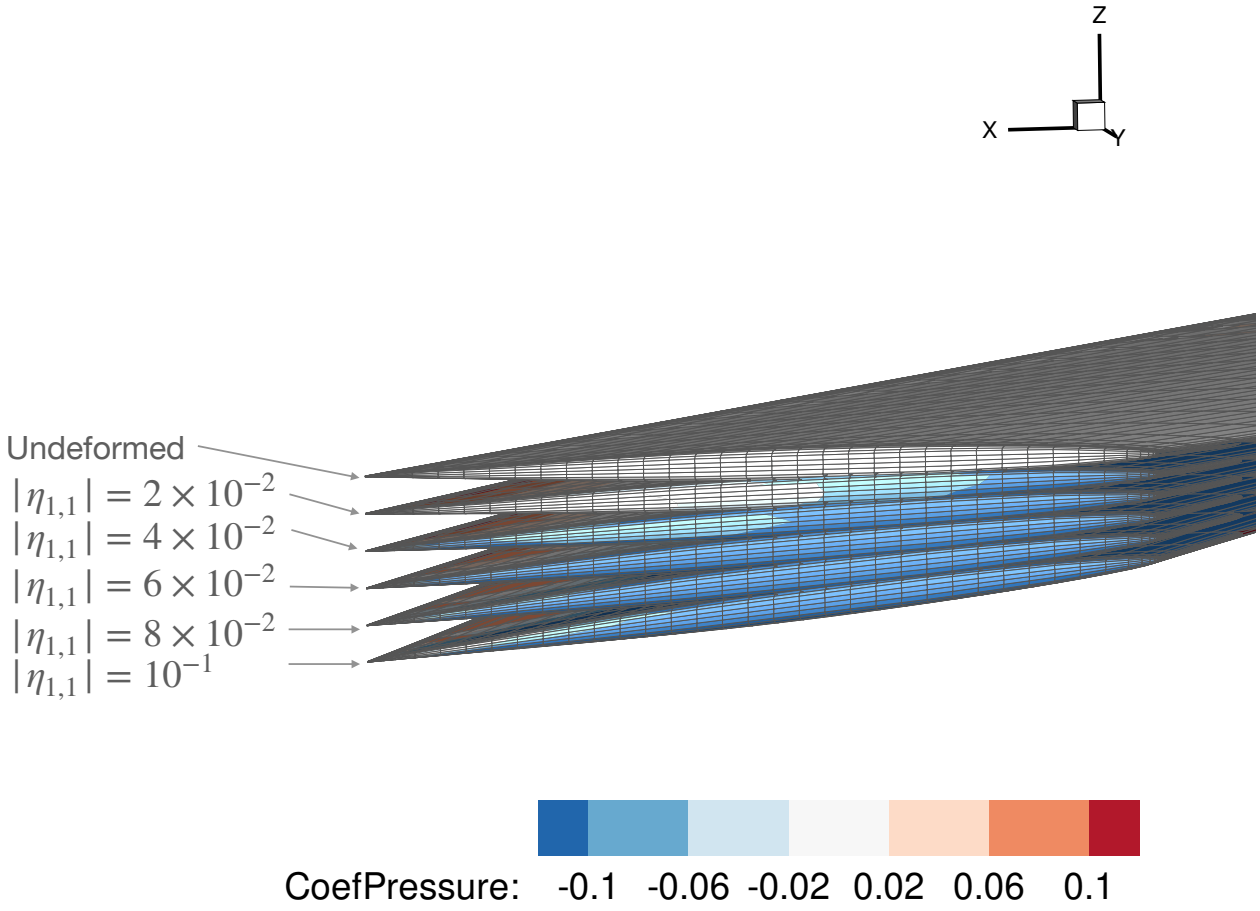


Figure 6.27: 5^{th} time instance of LCO responses for AGARD 445.6 at $M = 0.954$ with different prescribed motion magnitude

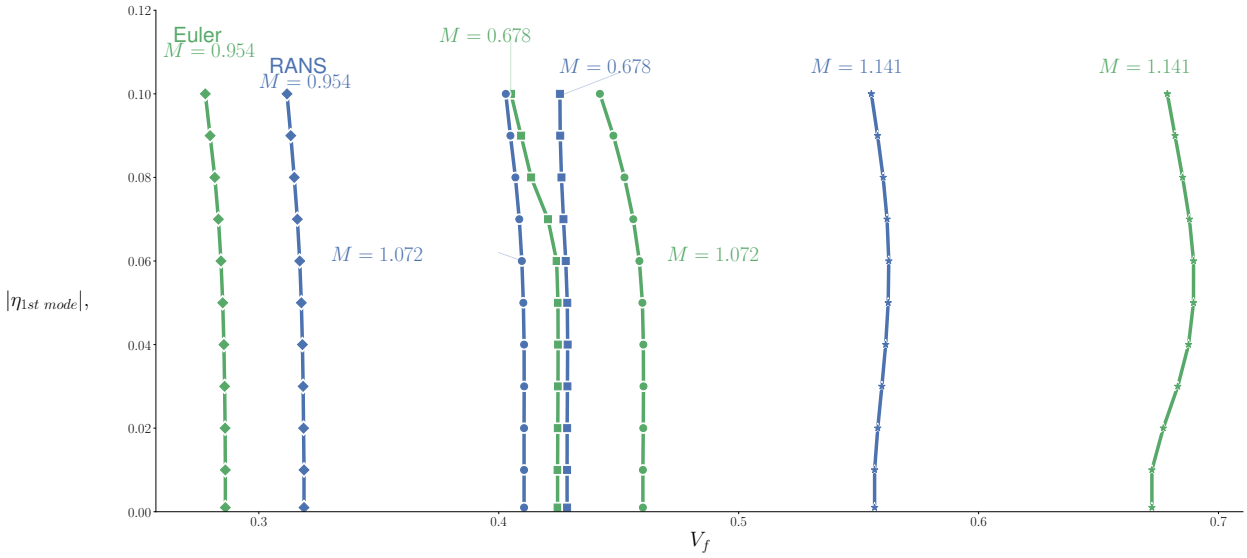


Figure 6.28: LCO behavior for AGARD 445.6 at $M = 1.072$

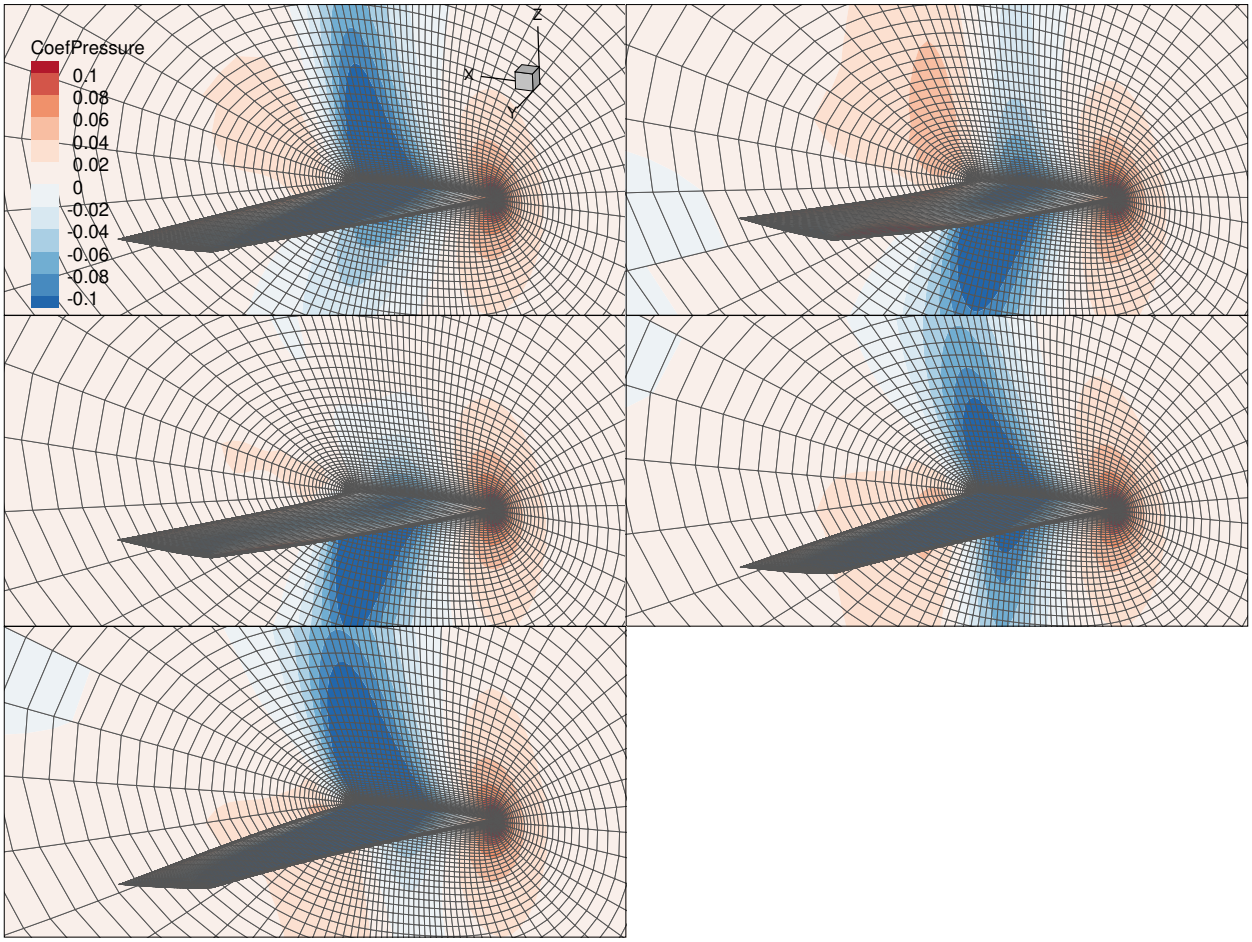


Figure 6.29: LCO behavior for AGARD 445.6 at $M = 0.954$

CHAPTER 7

Aerodynamic Shape Optimization for LCO Speed

In the chapter, we present results for LCO speed optimization. The chapter is organized as follows: In Section 7.1, we present results airfoil aerodynamic shape optimization result. Then, in Section 7.2, we present wing aerodynamic shape optimization result. For each section, besides optimization results, we also discuss the adjoint solution performance and adjoint derivative accuracy. More details about the adjoint method are presented earlier in Chapter 4.

7.1 Airfoil results

7.1.1 ADjoint solution performance

We consider the adjoint solution for variables LCO speed index, V_f, and average lift coefficient, $\overline{C}_l$ defined as

$$\overline{C}_l = \frac{1}{n} \sum_{i=1,n} C_l^i, \tag{7.1}$$

where C_l^i is a lift coefficient from the i^{th} time-instance and n is the total number of time-instances. For this test, we select $n = 3$. The flight conditions are described in Chapter 6 in Section 6.1.2, and the detailed structural set up are given in Table 6.7. In addition, the prescribed motion is set to be 0.1° and the Mach number is set as M $= 0.825$. A medium Euler mesh is used (see Fig. 6.1 for more details about the mesh).

The residual convergence history of the adjoint equations for LCO speed index, V_f, and average lift coefficient C_l Eq. (4.3) are shown in Fig. 7.1. The x axis is the number of iterations taken by generalized minimal residual (GMRES) method and the y axis is the logarithm of the adjoint equation residual. A relative residual decrease of 10^{-8} is achieved for both cases. During the solution of both equations, there is an initial slow convergence state. Later, for V_f after about 60 iterations and for $\overline{C}_l$ after about 30 iterations, both methods enter to a linear convergence rate regime similar to those reported by Kenway et al. [62], Shi et al. [121].

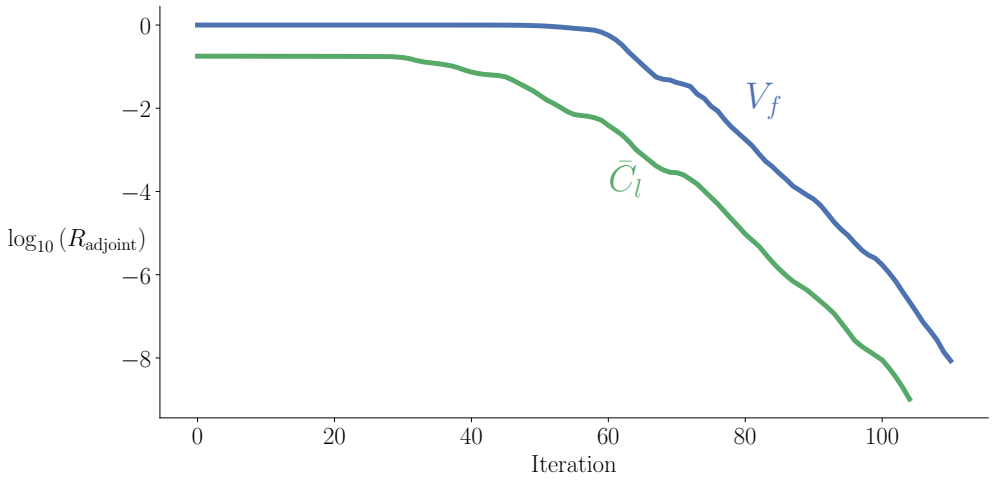


Figure 7.1: Adjoint equation residual convergence history for V_f and $\overline{C}_l$. A 10^{-8} relative residual convergence criterion is enforced.

The simulation is conducted using two cores, one for structure and the other for aerodynamics, using an Intel i7-4790L CPU @ 4.00 GHz. The computational times are reported in Table 7.1 with a primal solution time with 84.99 sec.

Table 7.1: Solution time of V_f and $\overline{C}_l$ adjoint equations using two cores (one for structure and the other for aerodynamic) with a medium Euler mesh with 3 time-instances.

Function of interest	Number of iteration	Time (sec)
V_f	110	262.89
$\overline{C}_l$	104	244.00

7.1.2 Derivative verification

We want to test the LCO speed sensitivity with respect to the design variables, i.e. dV_f/dx solved by ADjoint method describe in Chapter 4 with the FD method. The case setup is described in Section 7.1.1. We consider both geometric design variables and angle-of-attack (AOA). The geometric design variables are the y coordinates of 8 free-form deformation (FFD) points for the verification case as shown in 7.2.

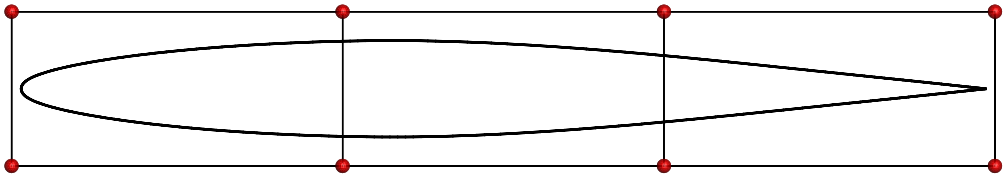


Figure 7.2: FFD box for the adjoint test

We compute the derivatives by solving the coupled adjoint equation Eq. (4.3). It is tested against the FD method with four different step sizes $10^{-5}, 10^{-6}, 10^{-7}$ and 10^{-8}. And the step with the minimal difference from the adjoint method has been shown in the table. We find that the relative error is between orders of 10^{-3} to 10^{-6} for all the design variables. And most of the design variables have a relative error in the order of 10^{-5} for the test. That is about the best performance achievable in a comparison with the FD method due to the truncation error as shown by Martins et al. [97], Shi et al. [121]. A more accurate test of the adjoint method can be achieved using the CS method [97].

7.1.3 LCO speed index optimization

We conduct an optimization to maximize the LCO speed index. The problem is set up for the demonstration purpose. For a more realistic case, the LCO speed index should be a constraint rather than an objective function. We have the "y" coordinates of FFD points and AOA as our design variables. The body fitted FFD is shown in Figure 7.3 with 16 points. We constrain the upper

Table 7.2: Verification of coupled-adjoint gradients for the airfoil case

Var	Coupled adjoint	FD	Rel. error	h_{opt}
1	-0.49878545	-0.49880140	3.20×10^{-5}	10^{-8}
2	0.48993934	0.48985941	-1.63×10^{-4}	10^{-7}
3	0.02804556	0.02808902	1.55×10^{-3}	10^{-8}
4	1.86143706	1.86150342	3.56×10^{-5}	10^{-6}
5	-0.98933053	-0.98931430	-1.64×10^{-5}	10^{-8}
6	2.03503684	2.03484926	-9.22×10^{-5}	10^{-7}
7	-1.25761856	-1.25766695	3.85×10^{-5}	10^{-5}
8	-1.67100493	-1.67109631	5.47×10^{-5}	10^{-7}
AOA	-0.06127731	-0.06127724	-1.16×10^{-6}	10^{-6}

and lower bounds ($0.01c$ and $-0.01c$ respectively where c is the chord length) of displacement of the "y" coordinates of FFD points. Since the FFD points at the leading edge and the trailing edge are forced to be symmetric with respect to the chord line, only two points out of the four points are independent. Thus, in total, we have 14 geometric design variables and 1 design variable for AOA. We also constrain the area, S, be between S_0 and $10S_0$, where S_0 is the initial area, i.e.,

$$S_0 \leq S \leq 10S_0. \tag{7.2}$$

Additionally, we set a constraint for the time averaged lift coefficient $\overline{C}_l$ defined as Eq. (7.1) to be

$$\overline{C}_l = 0.3. \tag{7.3}$$

And the bounds for AOA are defined as

$$0^\circ \leq \text{AOA} \leq 10^\circ. \tag{7.4}$$

The optimization problem is summarized in Table 7.3. We use SNOPT [36] as the optimizer that has a python interface from pyOptsparse [110, 144].

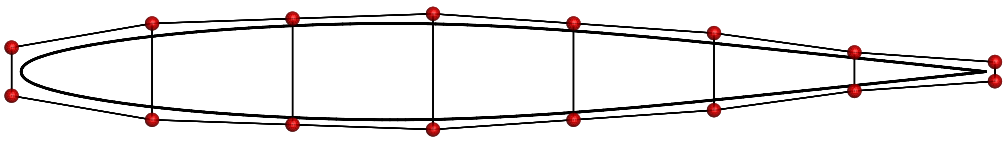


Figure 7.3: FFD box for the aerodynamic shape optimization

Table 7.3: Aerodynamic shape optimization problem

	Function/variable	Description	Quantity
maximize	V_f	LCO speed index	
w.r.t	y	FFD control points y coordinates	14
	AOA	angle of attack	1
s.t.	$-0.01c \leq y \leq 0.01c$	bounds on FFD control points y coordinates	14
	$0^\circ \leq \text{AOA} \leq 10^\circ$	bounds AOA y coordinates	1
	$\overline{C}_l = 0.3$	lift constraint	1
	$S_0 \leq S \leq 10S_0$	area constraint	1

The flight condition is similar with that used in Section 6.1.3 for the case at

$$M = 0.85, \tag{7.5}$$

within the transonic dip. We adjust the initial AOA for the optimization problem to be

$$\text{AOA} = 3.77^\circ \tag{7.6}$$

to match the average lift constraint. AOA is subsequently updated by the optimizer during the optimization process. A medium Euler mesh is used as described earlier in Fig. 6.1. The prescribed motion magnitude is set to be 1° and three time-instances are used.

The initial and final values of the functions of interest are summarized in Table 7.4. It is observed that the optimized shape has V_f at 0.85823 compared with the initial value at 0.61702. The objective function has increased by 39.09%. The constraint violation for $\overline{C}_l$ has increased a bit but is within the feasibility range set for the optimizer. For the area constraint, similar with

drag optimization result by He et al. [51], the optimizer choose to keep the current area instead of increasing it.

Table 7.4: Function values with baseline and optimized aerodynamic shapes

Function	Baseline	Optimized
V_f	0.61702	0.85823
$\overline{C}_l$	0.30003	0.30065
S	S_0	$1.00657S_0$

A detailed optimization history is shown in Fig. 7.4. V_f starts with a quick jump, it then keeps increasing at a constant rate, and it finally plateaus. The violation of $\overline{C}_l$ constraint increases rapidly at the beginning and decreases slowly afterward. The airfoil cross-section area decreases initially to almost the lower bound but later increases moderately to $1.00657S_0$. The optimizer has taken over 500 steps for this case to the final optimized solution. This is because we have limited the maximum step size for the optimizer to make sure that the solver converges for most points. We set the maximum step to be 10^{-5} for SNOPT. We observe that if no such limit is set, the optimization may fail at the first dozens of steps with a lower V_f. The failure is caused by some unphysical oscillatory geometry.

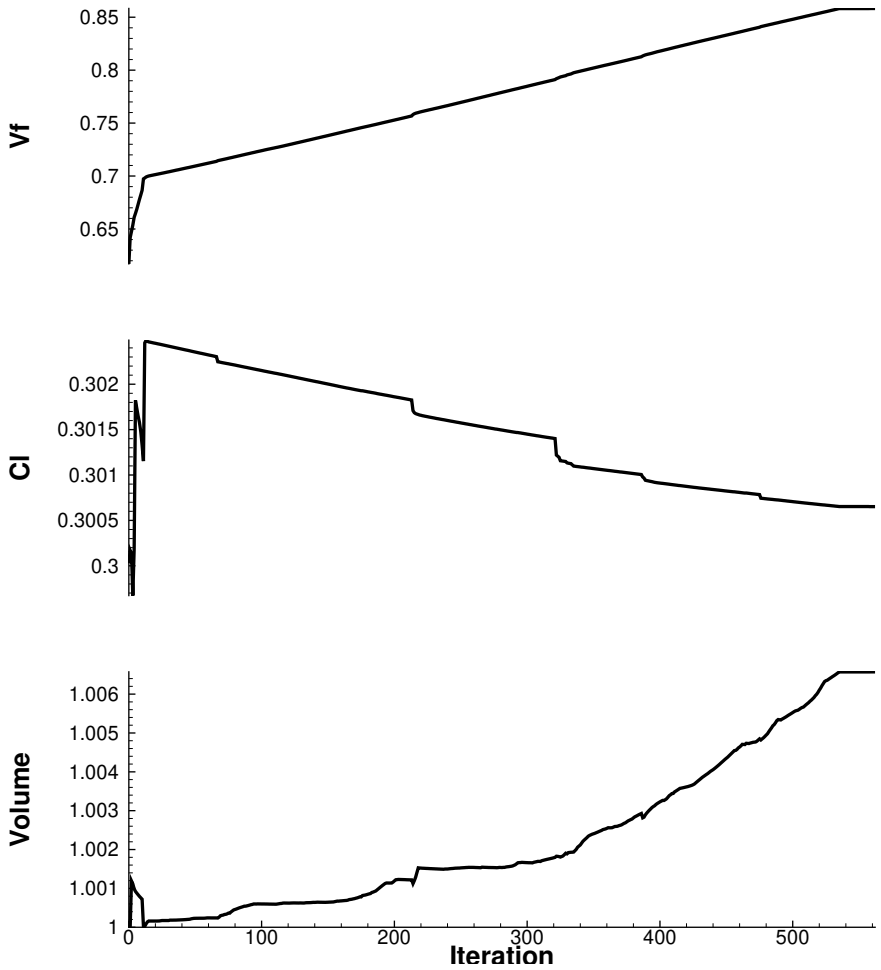


Figure 7.4: V_f optimization history

The baseline and optimized airfoils are shown in Fig. 7.5. The corresponding fields of the coefficient of pressure are shown in Fig. 7.6. And the surface pressure coefficient distribution is presented in Fig. 7.7. There is a kink created at the quarter chord in the pressure side of the airfoil. This may be related to the two shock waves generated in the pressure side as indicated by Figs. 7.6 and 7.7. For a drag minimization problem, we usually observe that the optimizer attempts to reduce the strength of the shock wave or even remove it [51] to reduce the pressure drag. However, for the current optimization problem, it seems that the strength of the existing shock wave has been

strengthened at the suction side, and at the same time, shock waves are created at the pressure side. The double-shock wave pattern has been observed multiple times when we conduct LCO speed optimization under different Mach numbers. But at this point, the benefit of such flow structures can not be explained. The optimization of the LCO speed adopted by this research and several research efforts in the field Prasad et al. [112], Thomas and Dowell [135] are more about verifying the algorithm and implementation than about generating physical meaningful geometries. Additional development is required for more physically sound optimization problems such as drag minimization with an LCO speed constraint.

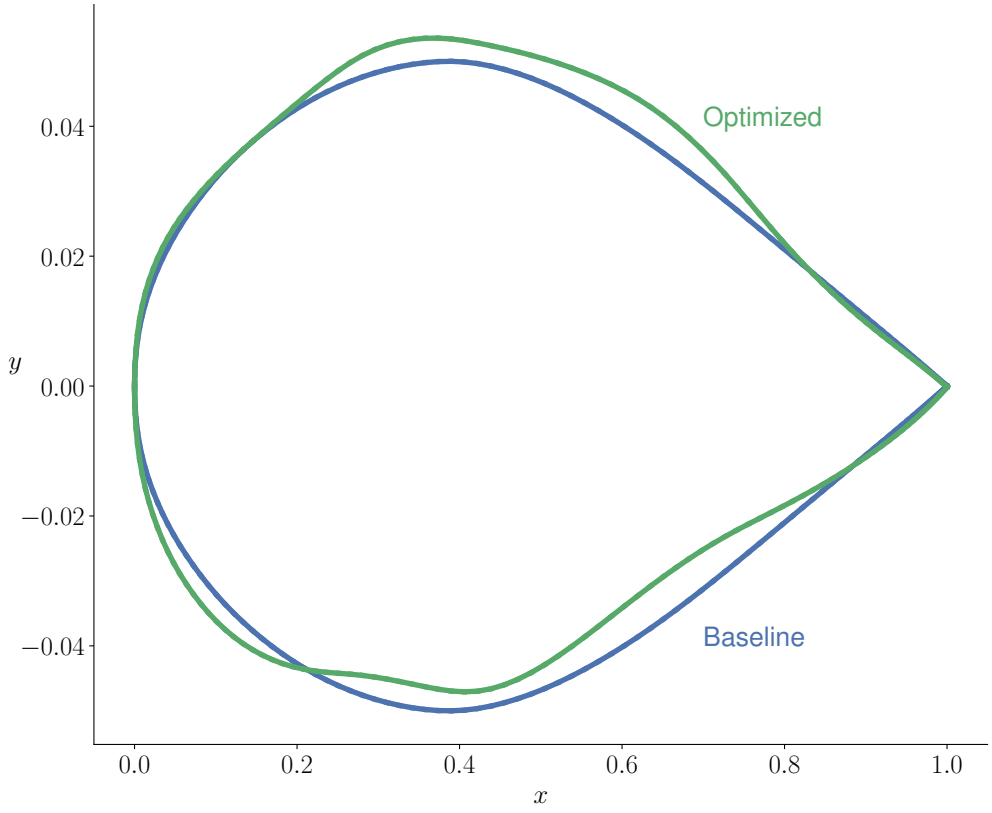


Figure 7.5: Geometry of baseline and optimized airfoil

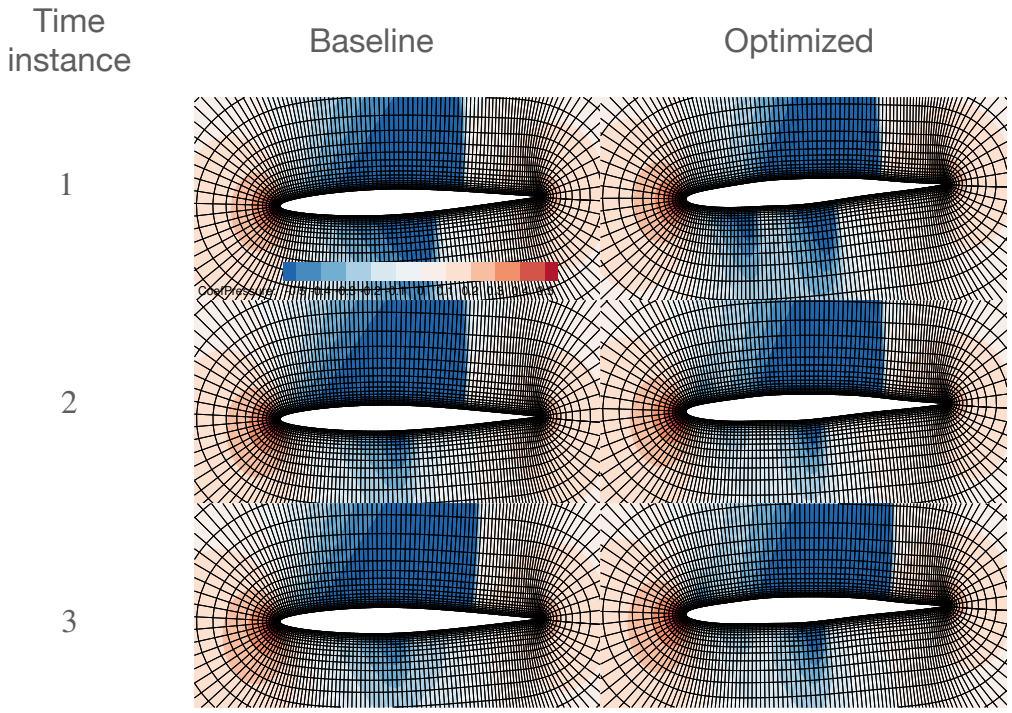


Figure 7.6: C_p distributions for the baseline (left) and the optimized (right) airfoils at different time-instances.

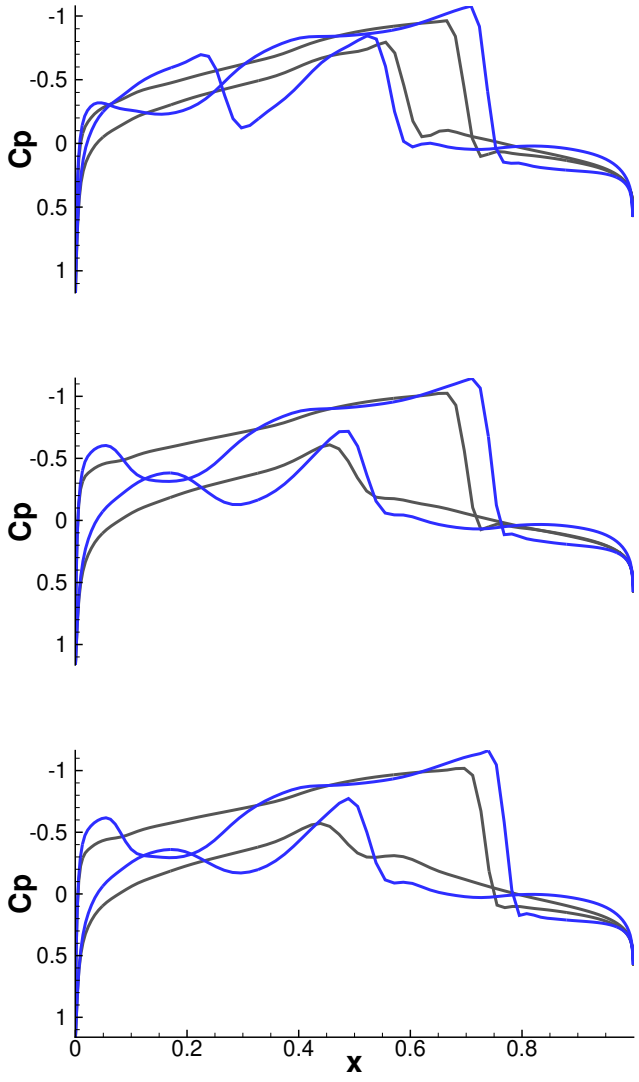


Figure 7.7: C_p distributions on the surface of the baseline (black) and optimized (blue) airfoils at different time-instances.

7.2 Wing results

7.2.1 ADjoint solution performance

Similar with the airfoil case, we study the adjoint solver performance for the LCO speed, V_f, and average lift coefficient, $\overline{C}_l$ defined in Eq. (7.1). In this study, we use the medium Euler mesh by coarsening the Euler mesh from Fig. 6.19 by a factor of 2 in all directions. The setup of the case is similar to that Section 6.2.2 at $M = 0.954$ besides we pick a relatively large AOA at 5° to make the problem more general and thus to expose any potential derivative error hidden in a symmetrical wing case. Three time-instances are considered in this test.

The computations are completed using a high-performance parallel computer with nodes composed of two 3.0 GHz Intel Xeon Gold processors, for a total of 36 cores and 180 GB memory per node. Out of all the cores, we partition it in such a way that one is dedicated to the structural component and the rest are dedicated to the aerodynamic component. The computational time for an LCO analysis is about 387.89 sec. The adjoint solution times for V_f and $\overline{C}_l$ are shown in Table 7.5. The convergence criterion for both variables is a reduction of residual by 8 orders compared with the initial residual. We observe that the iterations taken by $\overline{C}_l$ is about 8% more compared with that of V_f. This seems to be related to a longer initial slow convergence region as shown in Fig. 7.8. Similar to the airfoil test case, we observe a pattern with a slow convergence region at the beginning and a linear convergence region towards the end.

Table 7.5: Solution time of V_f and $\overline{C}_l$ adjoint equations using 36 cores (one for structure and the other for aerodynamic) with a medium Euler mesh with 3 time-instances.

Function of interest	Number of iteration	Time (sec)
V_f	127	363.30
$\overline{C}_l$	136	397.25

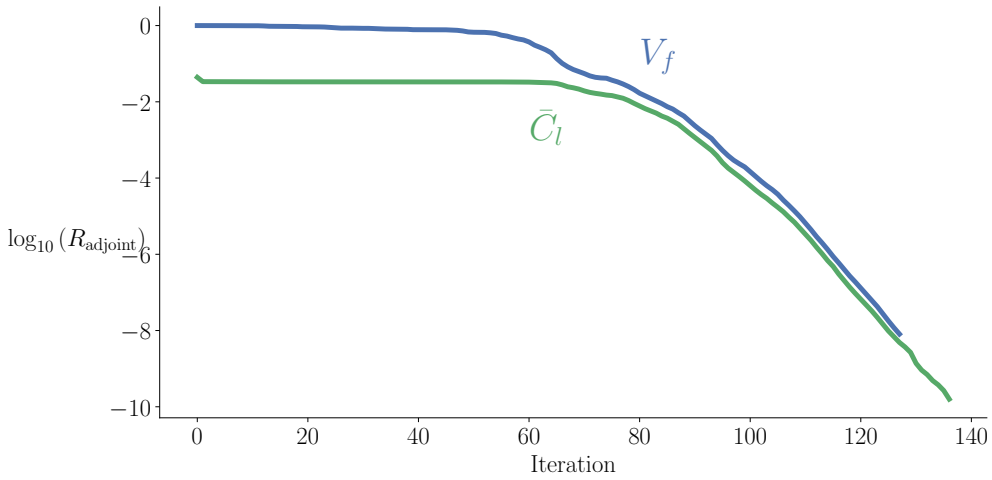


Figure 7.8: Adjoint equation residual convergence history for V_f and $\overline{C}_l$. 10^{-8} residual convergence is enforced.

7.2.2 Derivative verification

We verified the derivative for the airfoil case in Section 7.1.2. However, the test case for a wing is more complicated and needs a separate verification. At first, the CFD component is more involving. The airfoil case is a pseudo-2D test case meaning that we use one layer of hexahedron meshes for the airfoil instead of using 2D quadrilaterals. However, there are no fluxes through the surface of the airfoil due to symmetry. The wing case involves the nonzero flux of mass, momentum, and energy in the third coordinate. In addition, boundary conditions are set in different manners. And finally, the load and displacement transfer schemes are different for the two cases. Thus, we need to verify the more complicated time-spectral aeroelastic adjoint in a separate test.

The case setup is identical to that described earlier in Section 7.2.1. We consider geometric variables of FFD box and AOA. We use a coarse FFD box with 18 FFD points that able to move in z direction as shown in Fig. 7.9 to save some FD computation time. We compare the derivative computed with the adjoint method Eq. (4.3) and that computed by FD method. For FD method, we consider 4 different step sizes, $10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$ and the one with most digits match with the adjoint method is shown in Table 7.6. We observe that most of the derivatives have a relative

error of 10^{-6} between the adjoint and the FD method. There are four derivatives with 10^{-5}, and four with 10^{-7} relative error, respectively. This is the first time the LCO speed derivative can be computed with such accuracy for a wing case using CFD tools. Overall, it is likely that it is most digits match can be expected for a comparison with FD method due to the truncation error as detailed by Martins et al. [97]. A more accurate comparison with the CS method [97] may give more confidence to the derivative accuracy.

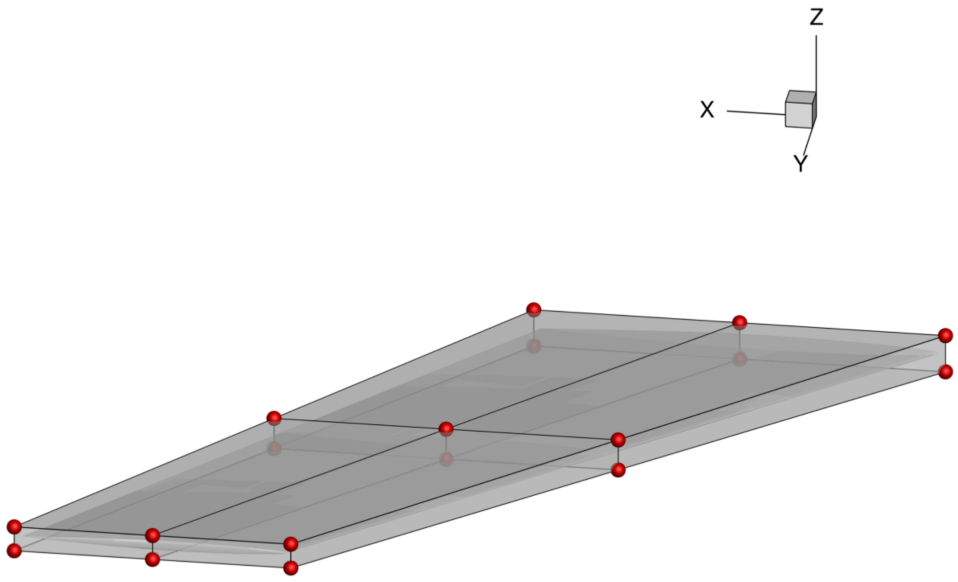


Figure 7.9: FFD box for the adjoint test

7.2.3 LCO speed optimization

With the derivative information obtained by the ADjoint method verified above, we are ready to conduct an aerodynamic shape optimization for the wing. Similar to the airfoil optimization problem we have set up earlier, the objective function is the LCO speed index. It is maximized to avoid LCO. The constraints include the time-averaged lift coefficient, $\overline{C}_L = 0.3$, the volume of

Table 7.6: Verification of coupled-adjoint gradients for the wing case.

Var	Coupled adjoint	FD	Rel. error	h_{opt}
1	0.10714487	0.10714620	1.24×10^{-5}	10^{-7}
2	−0.24528040	−0.24528059	7.75×10^{-7}	10^{-6}
3	−0.81450993	−0.81450864	-1.58×10^{-6}	10^{-7}
4	−0.39010121	−0.39010195	1.90×10^{-6}	10^{-7}
5	−0.09141936	−0.09142020	9.19×10^{-6}	10^{-7}
6	0.09613774	0.09613788	1.46×10^{-6}	10^{-6}
7	−0.06384253	−0.06384292	6.11×10^{-6}	10^{-6}
8	−0.07267708	−0.07267748	5.50×10^{-6}	10^{-7}
9	0.59039067	0.59039223	2.64×10^{-6}	10^{-7}
10	−0.11932325	−0.11932292	-2.77×10^{-6}	10^{-6}
11	0.16195491	0.16195787	1.83×10^{-5}	10^{-7}
12	−0.51788331	−0.51788325	-1.16×10^{-7}	10^{-6}
13	0.73402951	0.73403319	5.01×10^{-5}	10^{-7}
14	0.25486670	0.25486890	8.63×10^{-6}	10^{-8}
15	−0.04201696	−0.04201401	-7.02×10^{-5}	10^{-7}
16	−0.23273019	−0.23273112	4.00×10^{-6}	10^{-7}
17	0.10436176	0.10436169	-6.71×10^{-7}	10^{-7}
18	0.54089810	0.54089828	3.33×10^{-7}	10^{-7}
AOA	0.00962494	0.00962503	8.69×10^{-6}	10^{-6}

the wing shall be between V_0 and $10V_0$ where V_0 is the baseline volume of the wing, and a lower bound on thickness. There are also bounds on AOA and FFD points displacements. The problem setup is described in Table 7.7.

We use the same mesh and operation condition as that from Section 7.2.1 except for the AOA. We start with an AOA which gives a C_L close to the specified C_L. The Mach number is specified as $M = 0.954$ which is in the transonic dip as shown Fig. 6.20. We use a denser FFD box compared with the one used in Section 7.2.2 to give better control of the surface geometry. The FFD box with 50 FFD points is shown in Fig. 7.10.

The optimization history of the functions of the LCO speed, V_f, average lift coefficient, $\overline{C}_L$, and volume, V are shown in Fig. 7.11. V_f almost increases in a monotonic way instead of several points. $\overline{C}_L$ starts with a value that violates the constraint. Shortly into the optimization process, it

Table 7.7: Aerodynamic shape optimization problem

	Function/variable	Description	Quantity
maximize	V_f	LCO speed index	
w.r.t	y	FFD control points y coordinates	50
	AOA	angle of attack	1
s.t.	-0.02 m $\leq y \leq 0.02$ m	bounds on FFD control points y coordinates	50
	$0^\circ \leq$ AOA $\leq 10^\circ$	bounds AOA y coordinates	1
	$\overline{C}_L = 0.3$	lift constraint	1
	$t \geq 0.75t_0$	thickness constraint	25
	$V_0 \leq V \leq 10V_0$	volume constraint	1
	$y_{\text{upper}} = -y_{\text{lower}}$	symmetric leading/trailing edge constraints	10

takes a spike and decreases later to about 0.325. Later, $\overline{C}_L$ stays at a value of around 0.35 until for the last 20 iterations. Finally, it decreases to a value satisfying the lift constraint within the numerical tolerance. The volume constraint is almost perfectly satisfied through the optimization. The initial and final values of the functions are given in Table 7.8. V_f has increased quite significantly by about 118%. This indicates that the time-spectral aeroelastic adjoint solver is indeed a useful tool for the LCO speed derivative computation and optimization. This is the first optimization of V_f with respect to geometric design variables for a wing test case using time-spectral CFD method.

Table 7.8: Function values with baseline and optimized aerodynamic shapes

Function	Baseline	Optimized
V_f	2.78040×10^{-1}	6.04802×10^{-1}
$\overline{C}_l$	3.15069×10^{-1}	3.00165×10^{-1}
V	V_0	$1.00000V_0$

The initial and optimized flow field is shown in Fig. 7.12. We observe that after optimization the wing gets a wavy shape as illustrated by the right panel of Fig. 7.12. At slice B, the shock wave strength seems to be increased. This is the opposite of what has been observed for a drag minimization problem where the shock wave strength is usually decreased to reduce wave drag. The strengthened shock wave seems to be similar to what we have observed in te previous airfoil shape optimization problem Fig. 7.6. Besides, by taking a close look at the lift distribution on the

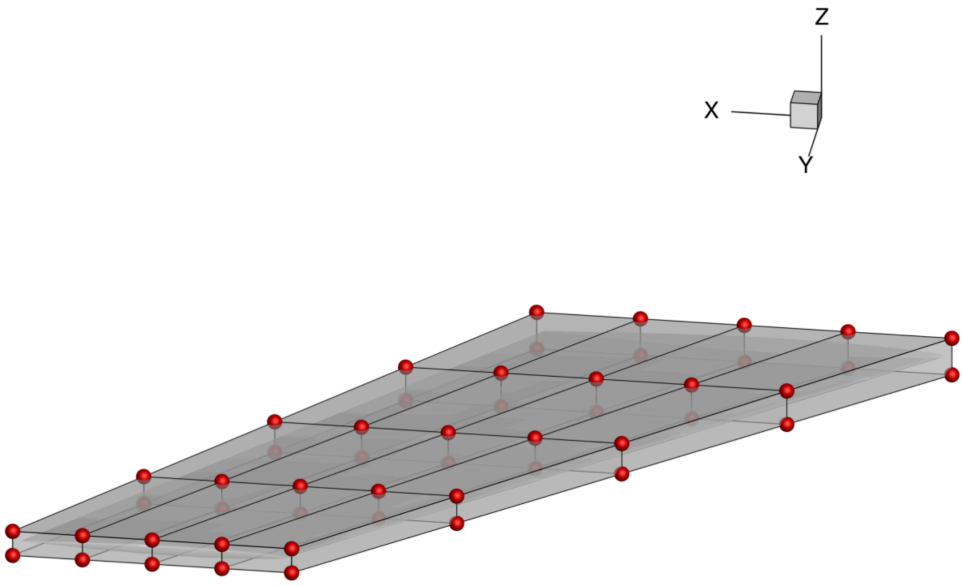


Figure 7.10: FFD box for the LCO speed optimization

left panel of Fig. 7.12, before optimization, it seems the lift has a relatively uniform distribution in the spanwise direction except that it has a bump in the middle. However, the optimized wing shifts more load in board. Shifting the load in board and reduce the load at the tip is favorable in the sense that it reduces the structural vibration at the tip. It seems that the load shift could be attributed to the shock waves created on the suction side.

The aerodynamic shape generated here is not realistic. This may be due to the unphysical problem setup–there is no wing designed to maximize the LCO speed. The optimizer may take advantage of the fact and creates features like shock waves to increase V_f. A better way of formulating the problem is to make the LCO speed as a constraint and use drag as the objective function.

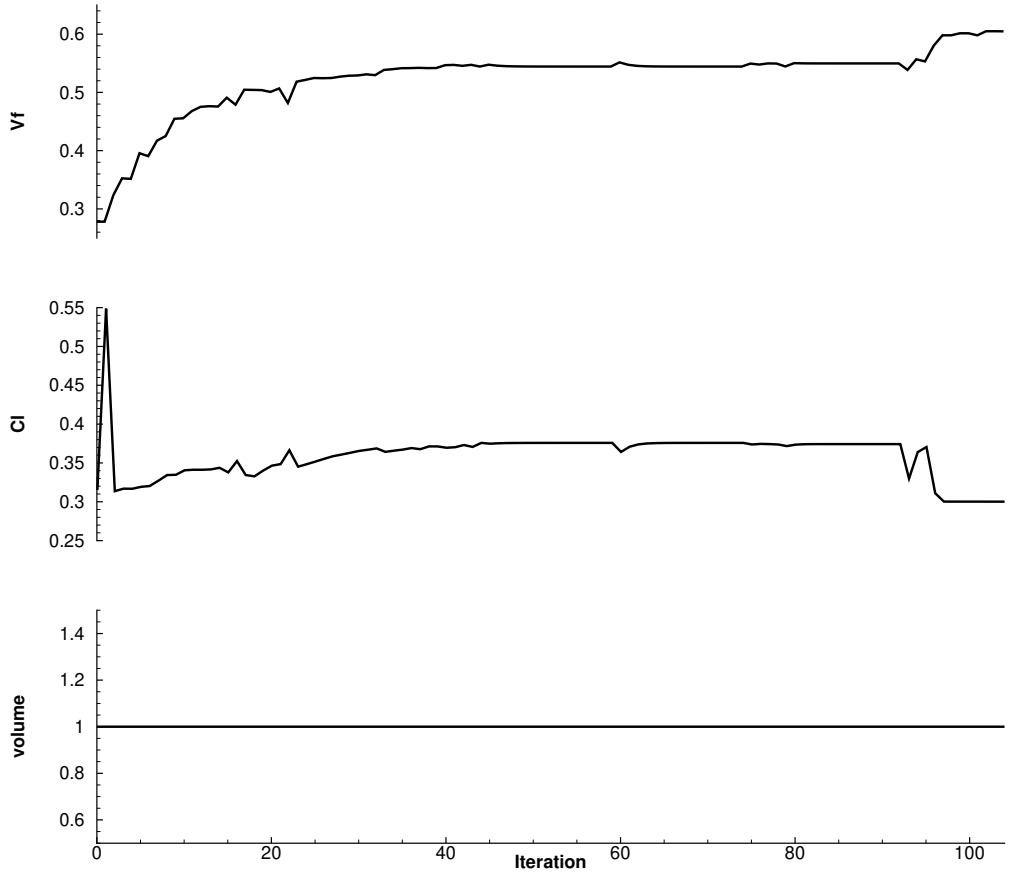


Figure 7.11: V_f optimization history

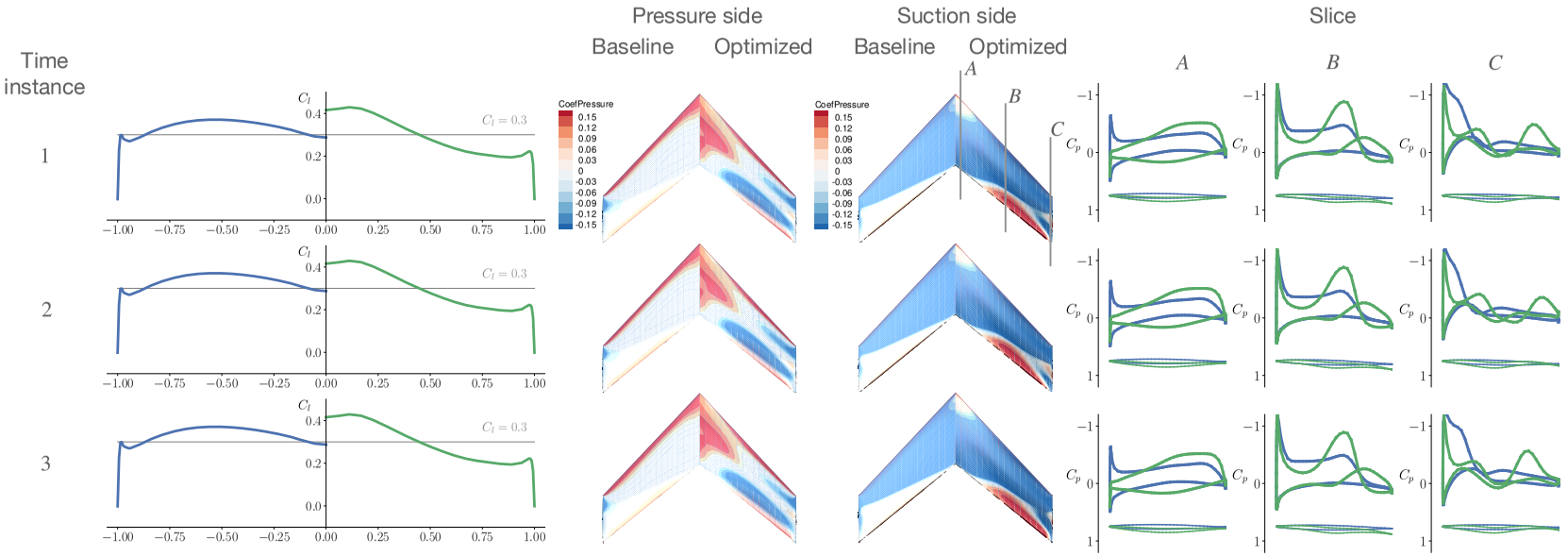


Figure 7.12: LCO speed optimization results. The block on the left shows the lift coefficient distribution. The block in the middle shows C_p distribution on pressure (left) and suction (right) sides. The block on the right shows C_p distribution over airfoil cross-sections at slices A to C. Baseline (blue) and optimized (green) results are shown. Row 1 to row 3 correspond with time-instance 1 to 3.

CHAPTER 8

Conclusion

In this work, we developed new computational methods to make it possible to perform aerodynamic shape optimization to suppress LCO. The new methods include CNK method for LCO simulation, the coupled ADjoint method for the LCO speed derivative computation, and two RAD-based formulae for structural mode and natural frequency derivative computation.

One challenge of LCO optimization using CFD tools is their high computational cost. To address this issue, we developed a Jacobian-free Newton–Krylov solver for both two-dimensional airfoil and three-dimensional wing. The LCO response is captured by time-spectral aeroelastic equation composed of the prescribed motion equations, CSD equations, and CFD equations. We considered both Euler and RANS equations for the CFD equations. The solver is more efficient compared with the segregated approach in the literature because each Newton step is cheaper to evaluate. The segregated method needed $\mathcal{O}(N_{\text{CSD}} \times n \times N_{\text{Newton}})$ CFD evaluations for each solution, where N_{CSD} is the number of DOFs in the CSD model, n is the number of time-instances, and N_{Newton} is the number of Newton steps. Our CNK method only requires that the CFD residual be driven to the machine precision twice: once at a warm start stage, and the other during the time-spectral aeroelastic equations solution.

We compared our results with those from the literature and our time-accurate solvers for (1) Time-spectral aerodynamic only simulations, (2) flutter boundary simulations, and (3) LCO simulations. We found that our results had an overall good match with those from the literature.

Since the preconditioners play a key role in the Krylov subspace solver performance, we developed three preconditioning strategies: (1) Direct inversion preconditioner, (2) Schur complement based preconditioner, and (3) SPS preconditioner. The direct inversion and Schur complement based preconditioners yield the same preconditioning matrix. However, the Schur complement based preconditioner obtains the preconditioning matrix by manipulating the block structure of the matrix. The SPS preconditioner is a block diagonal approximation of the Schur preconditioner. It is less accurate but cheaper to evaluate. We found that for the airfoil test case, direct inversion and Schur complement based preconditioners have the best performance. But when a more complicated FEM model is used, a trade-off between the direct inversion (or Schur complement based) preconditioner and the SPS preconditioner is anticipated.

We explored the impact of GCL on LCO simulation. We proposed a volume rate computation formula that satisfies GCL. GCL was considered to be critical for time-accurate CFD. However, in this study, we found that for the airfoil test case within a pitching range of $[-2^\circ, 2^\circ]$, the impact of GCL on V_f is no more than 0.5% for both Euler and RANS equations. Thus, it is not a critical factor that has to be taken into account for the time-spectral method-based LCO simulation in the motion magnitude range given above.

Another challenge of LCO optimization using CFD tools is that we need to evaluate the derivative of the LCO speed, V_f, with respect to a large number of design variables. To address this challenge, we proposed the use of the coupled adjoint method that is a monolithic way to compute the derivative. We based our work on previous research in the literature, which used a two-level adjoint formulation. The coupled adjoint is cheaper to compute compared to the two-level adjoint. This is because, for each coupled adjoint solution, we only need to solve one slightly bigger linear system compared to solving $\boldsymbol{\mathcal{O}}(N_{\text{CSD}} \times n)$ linear systems where N_{CSD} is the structural DOF and n is the number of time-instances.

The derivatives computed using the adjoint method were verified against those computed using the FD method for both two-dimensional and three-dimensional problems. We considered both

geometric design variables using FFD box and AOA. Most of the design variables have a relative error of 10^{-5} and 10^{-6} for two-dimensional airfoil and three-dimensional wing test cases, respectively. This is the first time the LCO speed derivative can be computed with such accuracy for a wing case using time-spectral method-based CFD tools.

Using the CNK solver and the coupled ADjoint solver developed in this work, we were able to conduct aerodynamic shape optimization of V_f for both a two-dimensional airfoil case and a three-dimensional wing case. The constraints taken into consideration include the time-averaged lift constraint, volume (area) constraints, and thickness constraints. We were able to find configurations to increase V_f by 39% and 118% for a two-dimensional airfoil case and a three-dimensional wing case, respectively. Thanks to the efficient solution method and derivative computation method developed in this work, we conducted the first optimization of V_f with respect to geometric design variables for a wing test case using the time-spectral method-based CFD method.

We noticed that the optimized configurations had unrealistic features, such as wavy patterns and strengthened shock waves. These features were used by the optimizer to shift the load in board. These features were not useful for drag reduction aerodynamic or aerostructural optimization problems because these features usually correspond with high wave drags. Thus, an alternative drag optimization with V_f constraint may be a more physical and well-posed problem to be explored in the future.

Finally, to extend the adjoint method to an aerostructural optimization problem, we addressed a computational bottleneck related to the structural modes and natural frequency derivative computation. If it is not handled carefully, this computation may have a cost of the order of the number of design variables. We developed two formulations based on RAD to reduce it to one single computation We named the methods: (1) the modal method, and (2) the improved modal method. The modal method only requires evaluating a handful of matrix-vector products. The improved modal method is more accurate compared with the modal method, but it requires equation solutions. We showed that the additional equations to be solved for the improved modal method are

elastic equations and the direct solution method can be leveraged to solve the equations efficiently.

We verified the proposed methods by implementing a reverse Lanczos iteration and the adjoint of an Euler–Bernoulli beam test case. Within about 10 iterations, the modal method and the improved modal method reduce the error by about 2 and 6 orders, respectively.

To conclude, we developed computational methods to make aerodynamic shape optimization to suppress LCO practical for wing cases, and the RAD formulae for the mode shapes and natural frequencies are likely to be useful for the future aerostructural optimization.

Bibliography

[1] Howard M. Adelman and Raphael T. Haftka. Sensitivity analysis of discrete structural systems. *AIAA Journal*, 24(5):823–832, 1986. doi:10.2514/3.48671.

[2] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press. ISBN 0-89791-412-0.

[3] John D. Anderson. *Fundamentals of Aerodynamics*. McGraw–Hill, 1991.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools for Scientific Computing*, pages 163–202. Birkhäuser Press, 1997. doi:10.1007/978-1-4612-1986-6_8.

[5] Robert E. Bartels and Bret Stanford. Economical Unsteady High Fidelity Aerodynamics in a Structural Optimization with a Flutter Constraint. In *35th AIAA Applied Aerodynamics Conference*, Denver, CO, June 5–9 2017. doi:10.2514/6.2017-4358.

[6] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[7] Atilim Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(1):5595—5637, January 2018.

[8] Joseph A. Beck, Jeffrey M. Brown, Onome E. Scott-Emuakpor, Emily B. Carper, and Alex A. Kaszynski. Modal expansion method for eigensensitivity calculations of cyclically symmetric bladed disks. *AIAA Journal*, 56(10):4112–4120, October 2018. doi:10.2514/1.j057322.

[9] Marc Benoit and Siva Nadarajah. On the geometric conservation law for the non linear frequency domain and time-spectral methods. *Computer Methods in Applied Mechanics and Engineering*, 355:690–728, October 2019. doi:10.1016/j.cma.2019.04.002.

[10] Philip S. Beran, Bret K. Stanford, and Kevin G. Wang. Fast Prediction of Flutter and Flutter Sensitivities. In *17th International Forum on Aeroelasticity and Structural Dynamics (IFASD)*, Como, Italy, June 25–28 2017. doi:10.2514/6.2017-1350.

[11] Michael L. Bernard and Alien J. Bronowicki. Modal expansion method for eigensensitivity with repeated roots. *AIAA Journal*, 32(7):1500–1506, July 1994. doi:10.2514/3.12221.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] Raymond L. Bisplinghoff and Holt Ashley. *Principles of Aeroelasticity*. Dover Publications, Inc., 1975.

[14] Julien Bohbot, Julien Garnier, Stephane Toumit, and Denis Darracq. Computation of the flutter boundary of an airfoil with a parallel navier-stokes solver. In *39th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, January 2001.

[15] Mohamed Amine Bouhlel, Sicheng He, and Joaquim R. R. A. Martins. Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes. *Structural and Multidisciplinary Optimization*, 61:1363–1376, March 2020. doi:10.1007/s00158-020-02488-5.

[16] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. DOVER Publications, Inc., 2000.

[17] Timothy R. Brooks, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Benchmark aerostructural models for the study of transonic aircraft wings. *AIAA Journal*, 56(7):2840–2855, July 2018. doi:10.2514/1.J056603.

[18] Timothy R. Brooks, Joaquim R. R. A. Martins, and Graeme J. Kennedy. High-fidelity aerostructural optimization of tow-steered composite wings. *Journal of Fluids and Structures*, 88:122–147, July 2019. doi:10.1016/j.jfluidstructs.2019.04.005.

[19] Peter N. Brown and Youcef Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, May 1990. doi:10.1137/0911026.

[20] P. C. Chen, Zhichao Zhang, and Eli Livne. Design-Oriented Computational Fluid Dynamics-Based Unsteady Aerodynamics for Flight-Vehicle Aeroelastic Shape Optimization. *AIAA Journal*, 53(12):3603–3619, 2015. doi:10.2514/1.J054024.

[21] Wengang Chen, Weiwei Zhang, Yilang Liu, and Jiaqing Kou. Accelerating the convergence of steady adjoint equations by dynamic mode decomposition. *Structural and Multidisciplinary Optimization*, 2020. doi:10.1007/s00158-020-02531-5.

[22] Seongim Choi and Anubhav Datta. Cfd prediction of rotor loads using time-spectral method and exact fluid-structure interface. 2008. doi:10.2514/6.2008-7325.

[23] S.S. Davis. Naca 64a010 (nasa ames model) oscillatory pitching. *AGARD Report 702, AGARD,Dataset 2.*, January 1982.

[24] Earl H. Dowell. *A Modern Course in Aeroelasticity*. Springer International Publishing, 2015. doi:10.1007/978-3-319-09453-3.

[25] Ariel Drachinsky and Daniella E. Raveh. Modal rotations: A modal-based method for large structural deformations of slender bodies. *AIAA Journal*, pages 1–15, May 2020. doi:10.2514/1.j058899.

[26] J. Driver and D. W. Zingg. Numerical aerodynamic optimization incorporating laminar-turbulent transition prediction. *AIAA Journal*, 45(8):1810–1818, 2007. doi:10.2514/1.23569.

[27] Paul S. Dwyer and M. S. Macphail. Symbolic matrix derivatives. *The Annals of Mathematical Statistics*, 19(4):517–534, May 1948.

[28] Stanley C. Eisenstat and Homer F. Walker. Choosing the forcing terms in an inexact newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, January 1996. doi:10.1137/0917003.

[29] Kivanc Ekici and Kenneth C. Hall. Harmonic balance analysis of limit cycle oscillations in turbomachinery. *AIAA Journal*, 49(7):1478–1487, July 2011. doi:10.2514/1.j050858.

[30] Benjamin Emerson, Tim Lieuwen, and Matthew P. Juniper. Local stability analysis and eigenvalue sensitivity of reacting bluff-body wakes. *Journal of Fluid Mechanics*, 788:549–575, January 2016. doi:10.1017/jfm.2015.724.

[31] Mark A. Finlayson. *ANSYS ICEM CFD User's Manual*. ANSYS, Inc., Canonsburg, PA, November 2013.

[32] R. L. Fox and M. P. Kapoor. Rates of change of eigenvalues and eigenvectors. *AIAA Journal*, 6(12):2426–2429, December 1968. doi:10.2514/3.5008.

[33] Michael I. Friswell and Sondipon Adhikari. Derivatives of complex eigenvectors using Nelson's method. *AIAA Journal*, 38(12):2355–2357, December 2000. doi:10.2514/2.907.

[34] Amin Ghadami, Carlos E.S. Cesnik, and Bogdan I. Epureanu. Model-less forecasting of hopf bifurcations in fluid-structural systems. *Journal of Fluids and Structures*, 76:1–13, 2018. doi:10.1016/j.jfluidstructs.2017.09.005.

[35] Mike Giles. An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation. Technical Report 08/01, Oxford University Computing Laboratory, Parks Road, Oxford, UK, January 2008.

[36] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal of Optimization*, 12(4):979–1006, 2002. doi:10.1137/S1052623499350013.

[37] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3 edition, 1996.

[38] Yiming Gong and Weiwei Zhang. Efficient aeroelastic solution based on time-spectral fluid–structure interaction method. *AIAA Journal*, 57(7):3014–3025, July 2019. doi:10.2514/1.j057628.

[39] Arathi Gopinath and Antony Jameson. Time spectral method for periodic unsteady computations over two- and three- dimensional bodies. Technical report, 2005.

[40] Andreas Griewank. *Evaluating Derivatives*. SIAM, Philadelphia, 2000.

[41] Raphael T. Haftka and Zafer Gürdal. *Elements of Structural Optimization*. Kluwer, 3rd edition, 1993.

[42] Gustavo L. O. Halila, Joaquim R. R. A. Martins, and Krzysztof J. Fidkowski. Adjoint-based aerodynamic shape optimization including transition to turbulence effects. *Aerospace Science and Technology*, (107):1–15, December 2020. doi:10.1016/j.ast.2020.106243.

[43] Kenneth C. Hall, Jeffrey P. Thomas, and Earl H. Dowell. Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows. *AIAA Journal*, 38(10):1853–1862, October 2000. doi:10.2514/2.867.

[44] Kenneth C. Hall, Jeffrey P. Thomas, and W. S. Clark. Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40(5):879–886, 2002. doi:10.2514/2.1754.

[45] L. Hascoët and V Pascual. TAPENADE 2.1 user's guide. Technical report 300, INRIA, 2004. URL `https://hal.inria.fr/inria-00069880/document`.

[46] Laurent Hascoet and Valérie Pascual. The Tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software*, 39(3):20:1–20:43, May 2013. ISSN 0098-3500. doi:10.1145/2450153.2450158.

[47] Ping He, Alton J. Luder, Charles A. Mader, Kevin J. Maki, and Joaquim R. R. A. Martins. A time-spectral adjoint approach for aerodynamic shape optimization under periodic wakes. In *AIAA SciTech Forum*, Orlando, FL, January 2020. AIAA. doi:10.2514/6.2020-2114.

[48] Ping He, Charles A. Mader, Joaquim R. R. A. Martins, and Kevin J. Maki. DAFoam: An open-source adjoint framework for multidisciplinary design optimization with OpenFOAM. *AIAA Journal*, 58(3), March 2020. doi:10.2514/1.J058853.

[49] Sicheng He, Eirikur Jonsson, Charles A. Mader, and Joaquim R. R. A. Martins. A coupled Newton–Krylov time spectral solver for flutter prediction. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Kissimmee, FL, January 2018. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2018-2149.

[50] Sicheng He, Eirikur Jonsson, Charles A. Mader, and Joaquim R. R. A. Martins. A coupled Newton–Krylov time-spectral solver for wing flutter and LCO prediction. In *AIAA Aviation Forum*, Dallas, TX, June 2019. doi:10.2514/6.2019-3549.

[51] Xiaolong He, Jichao Li, Charles A. Mader, Anil Yildirim, and Joaquim R. R. A. Martins. Robust aerodynamic shape optimization—from a circle to an airfoil. *Aerospace Science and Technology*, 87:48–61, April 2019. doi:10.1016/j.ast.2019.01.051.

[52] Daning Huang and Peretz P. Friedmann. An integrated aerothermoelastic analysis framework for predicting the response of composite panels. In *15th Dynamics Specialists Conference*. American Institute of Aeronautics and Astronautics, January 2016. doi:10.2514/6.2016-1090.

[53] Koji Isogai. On the transonic-dip mechanism of flutter of a sweptback wing. *AIAA Journal*, 17(7):793–795, July 1979. doi:10.2514/3.61226.

[54] Anthony Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, September 1988. doi:10.1007/BF01061285.

[55] Eirikur Jonsson, Gaetan K. W. Kenway, Graeme J. Kennedy, and Joaquim R. R. A. Martins. Development of flutter constraints for high-fidelity aerostructural optimization. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Denver, CO, June 2017. doi:10.2514/6.2017-4455. AIAA 2017-4455.

[56] Eirikur Jonsson, Charles A. Mader, Graeme J. Kennedy, and Joaquim R. R. A. Martins. Computational modeling of flutter constraint for high-fidelity aerostructural optimization. In *2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, San Diego, CA, January 2019. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2019-2354.

[57] Eirikur Jonsson, Cristina Riso, Christopher A. Lupp, Carlos E. S. Cesnik, Joaquim R. R. A. Martins, and Bogdan I. Epureanu. Flutter and post-flutter constraints in aircraft design optimization. *Progress in Aerospace Sciences*, 109:100537, August 2019. doi:10.1016/j.paerosci.2019.04.001.

[58] Farid Kachra and Siva K. Nadarajah. Aeroelastic solutions using the nonlinear frequency-domain method. *AIAA Journal*, 46(9):2202–2210, September 2008. doi:10.2514/1.27602.

[59] Graeme J. Kennedy and Joaquim R. R. A. Martins. A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures. *Finite Elements in Analysis and Design*, 87:56–73, September 2014. doi:10.1016/j.finel.2014.04.011.

[60] Graeme J. Kennedy, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Towards gradient-based design optimization of flexible transport aircraft with flutter constraints. In *Proceedings of the 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, June 2014. doi:10.2514/6.2014-2726. AIAA 2014-2726.

[61] Gaetan K. W. Kenway and Joaquim R. R. A. Martins. Buffet onset constraint formulation for aerodynamic shape optimization. *AIAA Journal*, 55(6):1930–1947, June 2017. doi:10.2514/1.J055172.

[62] Gaetan K. W. Kenway, Graeme J. Kennedy, and Joaquim R. R. A. Martins. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations. *AIAA Journal*, 52(5):935–951, May 2014. doi:10.2514/1.J052255.

[63] Gaetan K. W. Kenway, Charles A. Mader, Ping He, and Joaquim R. R. A. Martins. Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, 110: 100542, October 2019. doi:10.1016/j.paerosci.2019.05.002.

[64] Gaetan K.W. Kenway, Graeme. J. Kennedy, and Joaquim R. R. A. Martins. A CAD-free approach to high-fidelity aerostructural optimization. In *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, number AIAA 2010-9231, Fort Worth, TX, September 2010. doi:10.2514/6.2010-9231.

[65] Denis B. Kholodar, Jeffrey P. Thomas, Earl H. Dowell, and Kenneth C. Hall. Parametric study of flutter for an airfoil in inviscid transonic flow. *Journal of Aircraft*, 40(2):303–313, March 2003. doi:10.2514/2.3094.

[66] Jan F Kiviaho, Kevin Jacobson, Marilyn J Smith, and Graeme Kennedy. Application of a time-accurate aeroelastic coupling framework to flutter-constrained design optimization. In *2018 Multidisciplinary Analysis and Optimization Conference*, page 2932, 2018. doi:10.2514/6.2018-2932.

[67] Dana A Knoll and David E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004. doi:10.1016/j.jcp.2003.08.010.

[68] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, August 2012. doi:10.1007/s00158-012-0763-y.

[69] Tae Hee Lee. Adjoint method for design sensitivity analysis of multiple eigenvalues and associated eigenvectors. *AIAA Journal*, 45(8):1998–2004, August 2007. doi:10.2514/1.25347.

[70] Hang Li and Kivanc Ekici. Revisiting the One-shot method for modeling limit cycle oscillations: Extension to two-degree-of-freedom systems. *Aerospace Science and Technology*, 69:686–699, 2017. doi:10.1016/j.ast.2017.07.037.

[71] Hang Li and Kivanc Ekici. A novel approach for flutter prediction of pitch–plunge airfoils using an efficient one-shot method. *Journal of Fluids and Structures*, 82:651–671, October 2018. doi:10.1016/j.jfluidstructs.2018.08.012.

[72] Hang Li and Kivanc Ekici. Improved one-shot approach for modeling viscous transonic limit cycle oscillations. *AIAA Journal*, 56(8):3138–3152, August 2018. doi:10.2514/1.j056969.

[73] Hang Li and Kivanc Ekici. Aeroelastic modeling of a three-dimensional wing using the harmonic-balance-based one-shot method. In *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, January 2019. doi:10.2514/6.2019-0607.

[74] Hang Li and Kivanc Ekici. Aeroelastic modeling of the AGARD 445.6 wing using the harmonic-balance-based one-shot method. *AIAA Journal*, 57(11):4885–4902, November 2019. doi:10.2514/1.j058363.

[75] Yingqian Liao, Sicheng He, Joaquim R. R. A. Martins, and Yin Lu Young. Hydrostructural optimization of generic composite hydrofoils. In *AIAA SciTech Forum*, Orlando, FL, January 2020. AIAA. doi:10.2514/6.2020-0164.

[76] K. B. Lim, J. L. Junkins, and B. P. Wang. Re-examination of eigenvector derivatives. *Journal of Guidance, Control, and Dynamics*, 10(6):581–587, November 1987. doi:10.2514/3.20259.

[77] R. M. Lin and M. K. Lim. Complex eigensensitivity-based characterization of structures with viscoelastic damping. *The Journal of the Acoustical Society of America*, 100(5):3182–3191, November 1996. doi:10.1121/1.417202.

[78] R. M. Lin and T. Y. Ng. Frequency response functions and modal analysis of general non-viscously damped dynamic systems with and without repeated modes. *Mechanical Systems and Signal Processing*, 120:744–764, April 2019. doi:10.1016/j.ymssp.2018.10.032.

[79] R. M. Lin and T. Y. Ng. Prediction of mistuning effect of bladed disks using eigensensitivity analysis. *Engineering Structures*, 212:110416, June 2020. doi:10.1016/j.engstruct.2020.110416.

[80] R. M. Lin, J. E. Mottershead, and T. Y. Ng. A state-of-the-art review on theory and engineering applications of eigenvalue and eigenvector derivatives. *Mechanical Systems and Signal Processing*, 138:106536, April 2020. doi:10.1016/j.ymssp.2019.106536.

[81] F Liu, J Cai, Y Zhu, HM Tsai, and AS F. Wong. Calculation of wing flutter by a coupled fluid-structure method. *Journal of Aircraft*, 38(2):334–342, 2001. doi:10.2514/2.2766.

[82] Zhongsheng Liu, Suhuan Chen, Min Yu, and Youqun Zhao. Contribution of the truncated modes to eigenvector derivatives. *AIAA Journal*, 32(7):1551–1553, July 1994. doi:10.2514/3.12228.

[83] Edward Luke, Eric Collins, and Eric Blades. A fast mesh deformation method using explicit interpolation. *Journal of Computational Physics*, 231(2):586–601, January 2012. ISSN 0021-9991. doi:10.1016/j.jcp.2011.09.021.

[84] Christopher A. Lupp and Carlos E. S. Cesnik. A gradient-based flutter constraint including geometrically nonlinear deformations. In *2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, San Diego, California, 2019. AIAA. doi:10.2514/6.2019-1212.

[85] Christopher A. Lupp, Carlos E.S. Cesnik, Philip Beran, Joshua Deaton, and David Easterling. Including geometrical nonlinear flutter constraints in high fidelity aircraft optimization. In *International Forum on Aeroelasticity and Structural Dynamics 2019*, Savannah, Georgia, 2019.

[86] Zhoujie Lyu and Joaquim R. R. A. Martins. RANS-based aerodynamic shape optimization of a blended-wing-body aircraft. In *21st AIAA Computational Fluid Dynamics Conference*, San Diego, CA, July 2013. doi:10.2514/6.2013-2586.

[87] Zhoujie Lyu, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. RANS-based aerodynamic shape optimization investigations of the Common Research Model wing. In *Proceedings of the AIAA Science and Technology Forum and Exposition (SciTech)*, National Harbor, MD, January 2014. doi:10.2514/6.2014-0567. AIAA 2014-0567.

[88] Zhoujie Lyu, Zelu Xu, and Joaquim 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, July 2014. ICCFD8-2014-0203.

[89] Andrew C. Madden, Matthew P. Castanier, and Bogdan I. Epureanu. Mistuning identification of blisks at higher frequencies. *AIAA Journal*, 49(6):1299–1302, June 2011. doi:10.2514/1.j050427.

[90] Charles A. Mader and Joaquim R. R. A. Martins. Derivatives for time-spectral computational fluid dynamics using an automatic differentiation adjoint. *AIAA Journal*, 50(12): 2809–2819, December 2012. doi:10.2514/1.J051658.

[91] Charles A. Mader, Joaquim R. R. A. Martins, Juan J. Alonso, and Edwin van der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal*, 46(4):863–873, April 2008. doi:10.2514/1.29123.

[92] Charles A. Mader, Gaetan K. W. Kenway, Anil Yildirim, and Joaquim R. R. A. Martins. ADflow—an open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 2020. doi:10.2514/1.I010796.

[93] Charles Alexander Mader. *Stability-Constrained Aerodynamic Shape Optimization with Applications to Flying Wings*. PhD thesis, University of Toronto, August 2012.

[94] Ferran Marti and Feng Liu. Multiple equilibrium points of airfoil flutter in viscous flow. In *55th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, January 2017. doi:10.2514/6.2017-1647.

[95] Joaquim R. R. A. Martins and John T. Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal*, 51(11): 2582–2599, November 2013. doi:10.2514/1.J052184.

[96] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2021. (to be published).

[97] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, September 2003. doi:10.1145/838250.838251.

[98] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, May 2004. doi:10.2514/1.11478.

[99] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62, March 2005. doi:10.1023/B:OPTE.0000048536.47956.62.

[100] D. J. Mavriplis and Z. Yang. Time spectral method for periodic and quasi-periodic unsteady computations on unstructured meshes. *Mathematical Modelling of Natural Phenomena*, 6 (3):213–236, 2011. doi:10.1051/mmnp/20116309.

[101] M McMullen, Antony Jameson, and J Alonso. Application of a non-linear frequency domain solver to the euler and navier-stokes equations. In *40th AIAA Aerospace Sciences Meeting & Exhibit*, page 120, 2002.

[102] Matthew McMullen. *The Application of Non-Linear Frequency Domain Methods to the Euler and Navier-Stokes Equations*. PhD thesis, Stanford University, March 2003. URL citeseer.ist.psu.edu/mcmullen03application.html.

[103] Thomas. P. Minka. Old and new matrix algebra useful for statistics. 2000.

[104] Nathan Mundis and Dimitri Mavriplis. Quasi-periodic time spectral method for aeroelastic flutter analysis. 2013. doi:10.2514/6.2013-638.

[105] Durbha V. Murthy and Raphael T. Haftka. Derivatives of eigenvalues and eigenvectors of a general complex matrix. *International Journal for Numerical Methods in Engineering*, 26 (2):293–311, 1988. doi:10.1002/nme.1620260202.

[106] Uwe Naumann. *The Art of Differentiating Computer Programs—An Introduction to Algorithmic Differentiation*. SIAM, 2011.

[107] Richard B. Nelson. Simplified calculation of eigenvector derivatives. *AIAA Journal*, 14(9): 1201–1205, September 1976. doi:10.2514/3.7211.

[108] Max M. J. Opgenoord, Mark Drela, and Karen E. Willcox. Influence of transonic flutter on the conceptual design of next-generation transport aircraft. *AIAA Journal*, pages 1–15, March 2019. doi:10.2514/1.j057302.

[109] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Rettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[110] Ruben E. Perez, Peter W. Jansen, and Joaquim R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, January 2012. doi:10.1007/s00158-011-0666-3.

[111] Michael G. H. Piotrowski and David W. Zingg. Smooth local correlation-based transition model for the spalart–allmaras turbulence model. *AIAA Journal*, pages 1–19, 2020. doi:10.2514/1.j059784.

[112] Rachit Prasad, Hyunsoon Kim, and Seongim Choi. Flutter related design optimization using the time spectral and coupled adjoint method. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, January 2018. doi:10.2514/6.2018-0101.

[113] Rachit Prasad, Hyunsoon Kim, Seongim Choi, and Seulgi Yi. High fidelity prediction of flutter/LCO using time spectral method. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, January 2018. doi:10.2514/6.2018-0459.

[114] Donya Ramezanian, Dimitri Mavriplis, and Behzad R. Ahrabi. An order N log N parallel solver for time-spectral problems. *Journal of Computational Physics*, 411:109319, June 2020. doi:10.1016/j.jcp.2020.109319.

[115] Cristina Riso, Amin Ghadami, Carlos E. S. Cesnik, and Bogdan I. Epureanu. Data-driven forecasting of postflutter responses of geometrically nonlinear wings. *AIAA Journal*, 58(6): 2726–2736, June 2020. doi:10.2514/1.j059024.

[116] Carl S. Rudisill and Yee-Yeen Chu. Numerical methods for evaluating the derivatives of eigenvalues and eigenvectors. *AIAA Journal*, 13(6):834–837, June 1975. doi:10.2514/3.60449.

[117] Yousef Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993. doi:10.1137/0914028.

[118] Peter J. Schmid and Dan S. Henningson. *Stability and Transition in Shear Flows*. Springer New York, 2001. doi:10.1007/978-1-4613-0185-1.

[119] Ney Secco, Gaetan K. W. Kenway Ping He, Charles A. Mader, and Joaquim R. R. A. Martins. Efficient mesh generation and deformation for aerodynamic shape optimization. *AIAA Journal*, 2020. (In press).

[120] Yayun Shi, Raphael Gross, Charles A. Mader, and Joaquim R. R. A. Martins. Transition prediction based on linear stability theory with the RANS solver for three-dimensional configurations. In *Proceedings of the AIAA Aerospace Sciences Meeting, AIAA SciTech Forum*, Kissimmee, FL, January 2018. doi:10.2514/6.2018-0819.

[121] Yayun Shi, Charles A. Mader, Sicheng He, Gustavo L. O. Halila, and Joaquim R. R. A. Martins. Natural laminar-flow airfoil optimization design using a discrete adjoint approach. *AIAA Journal*, 58(11):4702–4722, 2020. doi:10.2514/1.J058944.

[122] Yukiko S. Shimizu and Krzysztof Fidkowski. Output-based error estimation for chaotic flows using reduced-order modeling. In *2018 AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 2018. doi:10.2514/6.2018-0826.

[123] Philippe Spalart and Steven Allmaras. A One-Equation Turbulence Model for Aerodynamic Flows. *La Recherche Aerospatiale*, 1:5–21, 1994.

[124] Bret Stanford and Philip Beran. Direct flutter and limit cycle computations of highly flexible wings for efficient analysis and optimization. *Journal of Fluids and Structures*, 36:111–123, jan 2013. doi:10.1016/j.jfluidstructs.2012.08.008.

[125] Bret Stanford, Carol D. Wieseman, and Christine Jutte. Aeroelastic Tailoring of Transport Wings Including Transonic Flutter Constraints. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Kissimmee, FL, January 2015. doi:10.2514/6.2015-1127.

[126] Weihua Su and Carlos E. S. Cesnik. Strain-based analysis for geometrically nonlinear beams: A modal approach. *Journal of Aircraft*, 51(3):890–903, May 2014. doi:10.2514/1.c032477.

[127] Praneeth Reddy Sudalagunta, Cornel Sultan, Rakesh K. Kapania, Layne T. Watson, and Pradeep Raj. Accurate computing of higher vibration modes of thin flexible structures. *AIAA Journal*, 54(5):1704–1718, May 2016. doi:10.2514/1.j054428.

[128] Weihan Tang, Seunghun Baek, and Bogdan I. Epureanu. Reduced-order models for blisks with small and large mistuning and friction dampers. *Journal of Engineering for Gas Turbines and Power*, 139(1), September 2016. doi:10.1115/1.4034212.

[129] Pierre-Olivier Tardif and Siva Nadarajah. Three-dimensional aeroelastic solutions via the nonlinear frequency-domain method. *AIAA Journal*, 55(10):3553–3569, October 2017. doi:10.2514/1.j054849.

[130] Jeffrey Thomas and Earl Dowell. Discrete adjoint method for aeroelastic design optimization. In *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 2298, 2014. doi:10.2514/6.2014-2298.

[131] Jeffrey Thomas and Earl Dowell. A fixed point iteration approach for harmonic balance based aeroelastic computations. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, January 2018. doi:10.2514/6.2018-1446.

[132] Jeffrey Thomas and Earl H. Dowell. Methodology for numerically stabilizing a harmonic balance based aeroelastic solution approach. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, January 2020. doi:10.2514/6.2020-1446.

[133] Jeffrey Thomas and Earl H. Dowell. Discrete adjoint constrained design optimization approach for unsteady transonic aeroelasticity and buffet. In *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, 2020. doi:10.2514/6.2020-3137.

[134] Jeffrey Thomas, Earl Dowell, and Kenneth C Hall. Discrete adjoint method for nonlinear aeroelastic sensitivities for compressible and viscous flows. In *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1860, 2013. doi:10.2514/6.2013-1860.

[135] Jeffrey P Thomas and Earl H Dowell. Discrete adjoint approach for nonlinear unsteady aeroelastic design optimization. *AIAA Journal*, pages 1–9, 2019. doi:10.2514/1.J057504.

[136] Jeffrey P. Thomas, Earl H. Dowell, and Kenneth C. Hall. A harmonic balance approach for modeling three-dimensional nonlinear unsteady aerodynamics and aeroelasticity. In *5th International Symposium on Fluid Structure International, Aeroeslasticity, and Flow Induced Vibration and Noise*. ASME, 2002. doi:10.1115/imece2002-32532.

[137] Jeffrey P. Thomas, Earl H. Dowell, and Kenneth C. Hall. Nonlinear inviscid aerodynamic effects on transonic divergence, flutter, and limit-cycle oscillations. *AIAA Journal*, 40(4): 638–646, apr 2002. doi:10.2514/2.1720.

[138] P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, October 1979. ISSN 0001-1452. doi:10.2514/3.61273.

[139] Sebastian Timme. Global instability of wing shock-buffet onset. *Journal of Fluid Mechanics*, 885, January 2020. doi:10.1017/jfm.2019.1001.

[140] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, March 2011. doi:10.1109/mcse.2011.37.

[141] B. P. Wang. Improved approximate methods for computing eigenvector derivatives in structural dynamics. *AIAA Journal*, 29(6):1018–1020, June 1991. doi:10.2514/3.59945.

[142] Qiqi Wang, David Gleich, Amin Saberi, Nasrollah Etemadi, and Parviz Moin. A monte carlo method for solving unsteady adjoint equations. *Journal of Computational Physics*, 227(12):6184–6205, 2008. doi:10.1016/j.jcp.2008.03.006.

[143] Qiqi Wang, Rui Hu, and Patrick Blonigan. Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014. doi:10.1016/j.jcp.2014.03.002.

[144] Neil Wu, Gaetan Kenway, Charles A. Mader, John Jasa, and Joaquim R. R. A. Martins. pyOptSparse: a Python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, 5(54):2564, October 2020. doi:10.21105/joss.02564.

[145] Shenren Xu, Sebastian Timme, and Kenneth J Badcock. Enabling off-design linearised aerodynamics analysis using Krylov subspace recycling technique. *Computers & Fluids*, 140:385–396, 2016. doi:10.1016/j.compfluid.2016.10.018.

[146] W. Yao and S. Marques. A harmonic balance method for nonlinear fluid structure interaction problems. *Computers & Structures*, 201:26–36, May 2018. doi:10.1016/j.compstruc.2018.02.003.

[147] E.C. Yates. Agard standard aeroelastic configuration for dynamic response, candidate configuration i—-wing 445.6. *NASA TM-100492*, 1987.

[148] Anil Yildirim, Gaetan K. W. Kenway, Charles A. Mader, and Joaquim R. R. A. Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397:108741, November 2019. ISSN 0021-9991. doi:10.1016/j.jcp.2019.06.018.

[149] De-Wen Zhang and Fu-Shang Wei. Computation of eigenvector derivatives with repeated eigenvalues using a complete modal space. *AIAA Journal*, 33(9):1749–1753, September 1995. doi:10.2514/3.12723.

[150] Zhichao Zhang, P. C. Chen, Shuchi Yang, Zhicun Wang, and Qiqi Wang. Unsteady Aerostructure Coupled Adjoint Method for Flutter Suppression. *AIAA Journal*, 53(8):2121–2129, 2015. doi:10.2514/1.J053495.

[151] Zhichao Zhang, Ping-Chih Chen, Qiqi Wang, Zhiqiang Zhou, Shuchi Yang, and Zhicun Wang. Adjoint based structure and shape optimization with flutter constraints. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2016. doi:10.2514/6.2016-1176.

Appendices

APPENDIX A

CSD Equations Example

The example is related with Section 2.2.

Consider,

$$\begin{bmatrix} \mathbf{M} & & \\ & \mathbf{M} & \\ & & \mathbf{M} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_1 \\ \ddot{\mathbf{u}}_2 \\ \ddot{\mathbf{u}}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{K} & & \\ & \mathbf{K} & \\ & & \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} = \frac{V_f^2}{\pi} \begin{bmatrix} \bar{\mathbf{f}}_1 \\ \bar{\mathbf{f}}_2 \\ \bar{\mathbf{f}}_3 \end{bmatrix}, \tag{A.1}$$

where the subscript denotes the time instance (e.g., $\mathbf{u}_2$ indicates displacement from the second instance) and $\bar{\mathbf{f}}_i$ is defined as $[-C_{l,i}, 2C_{m,i}]^\top$ for the i^{th} time instance. Permute the history vector,

$$\mathbf{Q} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} = \mathbf{Q} \begin{bmatrix} \frac{h_1}{b} \\ \alpha_1 \\ \frac{h_2}{b} \\ \alpha_2 \\ \frac{h_3}{b} \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \frac{h_1}{b} \\ \frac{h_2}{b} \\ \frac{h_3}{b} \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}, \tag{A.2}$$

where the permutation matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A.3}$$

With the permutation matrix $\mathbf{Q}$, we conduct spectral differentiation twice to get the second derivative (e.g., for the pitching mode: $\ddot{\boldsymbol{\alpha}}^n \approx \mathbf{D}(\omega)(\dot{\boldsymbol{\alpha}}^n) \approx \mathbf{D}(\omega)^2 \boldsymbol{\alpha}^n$). Then we have:

$$\begin{bmatrix} \mathbf{M} & & \\ & \mathbf{M} & \\ & & \mathbf{M} \end{bmatrix} \mathbf{Q}^\top \begin{bmatrix} \mathbf{D}(\omega)^2 & & \\ & \mathbf{D}(\omega)^2 & \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} + \begin{bmatrix} \mathbf{K} & & \\ & \mathbf{K} & \\ & & \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} = \frac{V_f^2}{\pi} \begin{bmatrix} \bar{\mathbf{f}}_1 \\ \bar{\mathbf{f}}_2 \\ \bar{\mathbf{f}}_3 \end{bmatrix}. \tag{A.4}$$

APPENDIX B

Derivation of Equations from Chapter 5

B.1 Dot product identity

The dot product identity originally proposed by Minka [103] is the key for the derivation of the RAD formulations. This identity is also known as a "*dot product test*" [46], which is an indication that the forward and reverse codes are consistent. Let $\mathbf{Q}$ be an input matrix and $\mathbf{H}$ be an output matrix dependent on $\mathbf{Q}$. For $\mathbf{H}(\mathbf{Q})$, the identity is

$$\mathrm{Tr}(\overline{\mathbf{H}}^{\intercal}\dot{\mathbf{H}}) = \mathrm{Tr}(\overline{\mathbf{Q}}^{\intercal}\dot{\mathbf{Q}}), \tag{B.1}$$

where

$$\begin{aligned} \dot{\mathbf{H}} &= \frac{\partial \mathbf{H}}{\partial \mathbf{Q}}\dot{\mathbf{Q}}, \\ \overline{\mathbf{Q}} &= \frac{\partial \mathbf{H}}{\partial \mathbf{Q}}^{\intercal}\overline{\mathbf{H}}. \end{aligned} \tag{B.2}$$

The equality is verified as follows

$$\begin{aligned} &\mathrm{Tr}(\overline{\mathbf{H}}^{\intercal}\dot{\mathbf{H}}) \\ =&\mathrm{Tr}\left(\overline{\mathbf{H}}^{\intercal}\left(\frac{\partial \mathbf{H}}{\partial \mathbf{Q}}\dot{\mathbf{Q}}\right)\right) && \text{(by the first equation from Eq. (B.2))} \\ =&\mathrm{Tr}\left(\left(\overline{\mathbf{H}}^{\intercal}\frac{\partial \mathbf{H}}{\partial \mathbf{Q}}\right)\dot{\mathbf{Q}}\right) && \text{(by reordering the multiplication sequence)} \\ =&\mathrm{Tr}\left(\overline{\mathbf{Q}}^{\top}\dot{\mathbf{Q}}\right) && \text{(by the second equation from Eq. (B.2)).} \end{aligned} \tag{B.3}$$

Using the identity and by matching terms, we can derive RAD formulations based on corresponding FAD formulations.

To demonstrate how to apply Eq. (B.1), consider a linear equation for simplicity

$$\mathbf{H}(\mathbf{Q}) = \mathbf{A}\mathbf{Q}, \tag{B.4}$$

where $\mathbf{A}$ is a constant matrix. Differentiating Eq. (B.4), we obtain the FAD term

$$\dot{\mathbf{H}} = \mathbf{A}\dot{\mathbf{Q}}, \tag{B.5}$$

and by inserting into Eq. (B.1) we obtain,

$$\text{Tr}(\overline{\mathbf{H}}^\intercal \mathbf{A}\dot{\mathbf{Q}}) = \text{Tr}(\overline{\mathbf{Q}}^\intercal \dot{\mathbf{Q}}). \tag{B.6}$$

Since the equation holds true for arbitrary $\dot{\mathbf{Q}}$, comparing the LHS and RHS we conclude that

$$\overline{\mathbf{Q}} = \mathbf{A}^\top \overline{\mathbf{H}}, \tag{B.7}$$

which is the RAD result.

This equation can be generalized to multiple inputs and multiple outputs by summing up the input product traces on one side of the equation and the output on the other. For the generalized eigenvalue problem, we have inputs, $\mathbf{K}, \mathbf{M}$, and outputs, $\mathbf{\Lambda}, \mathbf{\Phi}$, we obtain

$$\text{Tr}(\overline{\mathbf{\Lambda}}^\intercal \dot{\mathbf{\Lambda}}) + \text{Tr}(\overline{\mathbf{\Phi}}^\intercal \dot{\mathbf{\Phi}}) = \text{Tr}(\overline{\mathbf{K}}^\intercal \dot{\mathbf{K}}) + \text{Tr}(\overline{\mathbf{M}}^\intercal \dot{\mathbf{M}}). \tag{B.8}$$

This expression is the foundation to derive the RAD formulations for the modal method, presented in the next section, and the improved modal method presented in Section 5.3.3. This process was proposed by Minka [103] and was used by Giles [35] to derive a series of very useful RAD results.

B.2 Trace identities

These are several matrix identities used frequently in the paper for the trace operation[35, 103]:

$$\begin{aligned} \text{Tr}(\mathbf{AB}) &= \text{Tr}(\mathbf{BA}) \\ \text{Tr}(\mathbf{A} + \mathbf{B}) &= \text{Tr}(\mathbf{A}) + \text{Tr}(\mathbf{B}) \\ \text{Tr}(\mathbf{A}(\mathbf{B} \circ \mathbf{C})) &= \text{Tr}((\mathbf{A} \circ \mathbf{B}^\intercal)\mathbf{C}). \end{aligned} \tag{B.9}$$

B.3 Direct method for eigenvalue and eigenvector sensitivities

The direct method for derivative computation for the i^{th} eigenpair, $(\lambda_i, \boldsymbol{\phi}_i)$ requires the solution of the linear system [41],

$$\begin{bmatrix} \mathbf{M} - \mathbf{K}\lambda_i & -\mathbf{K}\boldsymbol{\phi}_i \\ 2\boldsymbol{\phi}_i^\intercal \mathbf{M} & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\phi}}_i \\ \dot{\lambda}_i \end{bmatrix} = \begin{bmatrix} -\dot{\mathbf{M}}\boldsymbol{\phi}_i + \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i \\ -\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i \end{bmatrix}. \tag{B.10}$$

B.4 Derivation of Eq. 5.19

In this section we provide the derivation for $\overline{\mathrm{M}}_i$. The derivation for $\overline{\mathbf{K}}_i$ is similar and is therefore omitted. From Eq. (5.18), we want to evaluate

$$\overline{\mathbf{M}}_i = \frac{\partial \mathbf{R}}{\partial \mathrm{M}}^{\intercal} (-\boldsymbol{\psi}_i). \tag{B.11}$$

As mentioned in the main text of the paper, we evaluate this product using RAD. Here, $-\boldsymbol{\psi}_i$ can be taken as a seed for $\mathbf{R}$, i.e., $-\boldsymbol{\psi}_i$ is an instance of $\overline{\mathbf{R}}$. We use the following identity from Eq. (B.1) for the derivation,

$$\mathrm{Tr}(\overline{\mathbf{R}}^{\intercal}\dot{\mathbf{R}}) = \mathrm{Tr}(\overline{\mathrm{M}}_i^{\intercal}\dot{\mathrm{M}}_i). \tag{B.12}$$

Before proceeding with deriving the RAD formulation, we derive the FAD expressions. We differentiate Eq. (5.11) to obtain the partial derivative of $\mathbf{R}$ with respect to M. The FAD formulation is given as

$$\dot{\mathbf{R}} = \begin{bmatrix} \dot{\mathrm{M}}_i\boldsymbol{\phi}_i \\ \boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i \end{bmatrix}. \tag{B.13}$$

Now we derive $\overline{\mathrm{M}}_i$. By taking $-\boldsymbol{\psi}_i$ as a reverse seed, we obtain

$$\mathrm{Tr}(\overline{\mathbf{R}}^{\intercal}\dot{\mathbf{R}}) = \mathrm{Tr}(-\boldsymbol{\psi}_i^{\intercal}\dot{\mathbf{R}}). \tag{B.14}$$

We now substitute in the FAD formulation Eq. (B.13),

$$\mathrm{Tr}(-\boldsymbol{\psi}_i^{\intercal}\dot{\mathbf{R}}) = \mathrm{Tr}\left(-\boldsymbol{\psi}_i^{\intercal}\begin{bmatrix} \dot{\mathrm{M}}_i\boldsymbol{\phi}_i \\ \boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i \end{bmatrix}\right). \tag{B.15}$$

Expanding the $\boldsymbol{\psi}_i = \begin{bmatrix}\boldsymbol{\psi}_{\mathbf{R}_1}^{\intercal} & \boldsymbol{\psi}_{\mathrm{R}_2}^{\intercal}\end{bmatrix}^{\intercal}$, we get

$$\mathrm{Tr}\left(-\boldsymbol{\psi}_i^{\intercal}\begin{bmatrix} \dot{\mathrm{M}}_i\boldsymbol{\phi}_i \\ \boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i \end{bmatrix}\right) = \mathrm{Tr}\left(-\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i - \boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i\right) \tag{B.16}$$

Now using the first identity from Eq. (B.9), we obtain

$$\mathrm{Tr}\left(-\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i - \boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\boldsymbol{\phi}_i\right) = \mathrm{Tr}\left(-\boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal}\dot{\mathrm{M}}_i - \boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\right), \tag{B.17}$$

and by factoring out similar terms, we obtain

$$\mathrm{Tr}\left(-\boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal}\dot{\mathrm{M}}_i - \boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\dot{\mathrm{M}}_i\right) = \mathrm{Tr}\left(\left(-\boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal} - \boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\right)\dot{\mathrm{M}}_i\right). \tag{B.18}$$

By Eq. (B.12), we can then write

$$\mathrm{Tr}\left(\left(-\boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathbf{R}_1}^{\intercal} - \boldsymbol{\phi}_i\boldsymbol{\psi}_{i,\mathrm{R}_2}\boldsymbol{\phi}_i^{\intercal}\right)\dot{\mathrm{M}}_i\right) = \mathrm{Tr}(\overline{\mathrm{M}}_i^{\intercal}\dot{\mathrm{M}}_i). \tag{B.19}$$

Since the equation holds for arbitrary $\dot{\mathrm{M}}$, comparing and matching the LHS and RHS we conclude that

$$\overline{\mathrm{M}}_i = -\left(\boldsymbol{\psi}_{i,\mathbf{R}_1} + \boldsymbol{\phi}_i \boldsymbol{\psi}_{i,\mathrm{R}_2}\right) \boldsymbol{\phi}_i^{\intercal}, \tag{B.20}$$

B.5 Derivation of Eq. 5.23

We use the identity Eq. (B.8), which is a generalized form of Eq. (B.1) for the multiple input and output matrices case, together with the FAD result Eq. (5.21) for the RAD derivation. First, using Eq. (B.8) and Eq. (5.21), we have

$$\begin{aligned}
&\mathrm{Tr}\left(\overline{\mathrm{M}}^{\intercal}\dot{\mathrm{M}}\right) + \mathrm{Tr}\left(\overline{\mathbf{K}}^{\intercal}\dot{\mathbf{K}}\right) \\
=&\mathrm{Tr}(\overline{\boldsymbol{\Lambda}}^{\intercal}\dot{\boldsymbol{\Lambda}}) + \mathrm{Tr}(\overline{\boldsymbol{\Phi}}^{\intercal}\dot{\boldsymbol{\Phi}}) \\
=&\mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda} + \boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right) \\
&+ \mathrm{Tr}\left(\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda} + \boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) - \frac{1}{2}\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right).
\end{aligned} \tag{B.21}$$

Regrouping the terms, we obtain

$$\begin{aligned}
&\mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda} + \boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right) \\
&+ \mathrm{Tr}\left(\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda} + \boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) - \frac{1}{2}\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right) \\
=&\mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) + \overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) - \frac{1}{2}\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right) \\
&+ \mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right)\right) + \overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right)\right)\right),
\end{aligned} \tag{B.22}$$

We then use the third identity from Eq. (B.9) to get

$$\begin{aligned}
&\mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) + \overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) - \frac{1}{2}\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{I} \circ \left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right)\right) \\
&+ \mathrm{Tr}\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\left(\mathbf{I} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right)\right) + \overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\left(\mathbf{F} \circ \left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right)\right)\right) \\
=&\mathrm{Tr}\left(\left(\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\right) \circ \mathbf{I}\right)\left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right) + \left(\left(\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\right) \circ \mathbf{F}^{\intercal}\right)\left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right) - \frac{1}{2}\left(\left(\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\right) \circ \mathbf{I}\right)\left(\boldsymbol{\Phi}^{\intercal}\dot{\mathrm{M}}\boldsymbol{\Phi}\right)\right) \\
&+ \mathrm{Tr}\left(\left(\left(\overline{\boldsymbol{\Lambda}}^{\intercal}\boldsymbol{\Lambda}\right) \circ \mathbf{I}\right)\left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right) + \left(\left(\overline{\boldsymbol{\Phi}}^{\intercal}\boldsymbol{\Phi}\right) \circ \mathbf{F}^{\intercal}\right)\left(-\boldsymbol{\Phi}^{\intercal}\dot{\mathbf{K}}\boldsymbol{\Phi}\boldsymbol{\Lambda}\right)\right).
\end{aligned} \tag{B.23}$$

Then, we use the first identity from Eq. (B.9) to get

$$\begin{aligned}
&\mathrm{Tr}\left(\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\left(\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)+\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\left(\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)-\frac{1}{2}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\left(\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\mathbf{\Phi}\right)\right)\\
&+\mathrm{Tr}\left(\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\left(-\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\mathbf{\Phi}\mathbf{\Lambda}\right)+\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\left(-\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\mathbf{\Phi}\mathbf{\Lambda}\right)\right)\\
=&\mathrm{Tr}\left(\mathbf{\Phi}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}+\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}-\frac{1}{2}\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\right)\\
&+\mathrm{Tr}\left(-\mathbf{\Phi}\mathbf{\Lambda}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}-\mathbf{\Phi}\mathbf{\Lambda}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\right).
\end{aligned}\tag{B.24}$$

Finally, we merge similar terms, and we obtain

$$\begin{aligned}
&\mathrm{Tr}\left(\mathbf{\Phi}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}+\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}-\frac{1}{2}\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{M}}\right)\\
&+\mathrm{Tr}\left(-\mathbf{\Phi}\mathbf{\Lambda}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}-\mathbf{\Phi}\mathbf{\Lambda}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\dot{\mathbf{K}}\right)\\
=&\mathrm{Tr}\left(\left(\mathbf{\Phi}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}+\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}-\frac{1}{2}\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{M}}\right)\\
&+\mathrm{Tr}\left(\left(-\mathbf{\Phi}\mathbf{\Lambda}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}-\mathbf{\Phi}\mathbf{\Lambda}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{K}}\right),
\end{aligned}\tag{B.25}$$

which is equal to

$$\begin{aligned}
&\mathrm{Tr}\left(\left(\mathbf{\Phi}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}+\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}-\frac{1}{2}\mathbf{\Phi}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{I}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{M}}\right)\\
&+\mathrm{Tr}\left(\left(-\mathbf{\Phi}\mathbf{\Lambda}\left(\overline{\mathbf{\Lambda}}^{\intercal}\mathbf{\Lambda}\right)\mathbf{\Phi}^{\intercal}-\mathbf{\Phi}\mathbf{\Lambda}\left(\left(\overline{\mathbf{\Phi}}^{\intercal}\mathbf{\Phi}\right)\circ\mathbf{F}^{\intercal}\right)\mathbf{\Phi}^{\intercal}\right)\dot{\mathbf{K}}\right),\\
=&\mathrm{Tr}\left(\overline{\mathbf{M}}^{\intercal}\dot{\mathbf{M}}\right)+\mathrm{Tr}\left(\overline{\mathbf{K}}^{\intercal}\dot{\mathbf{K}}\right).
\end{aligned}\tag{B.26}$$

Since the equality holds for arbitrary $\dot{\mathbf{M}}$, and $\dot{\mathbf{K}}$, comparing the LHS and RHS we conclude that

$$\begin{aligned}
\overline{\mathbf{M}}&=\mathbf{\Phi}\left(\mathbf{\Lambda}\overline{\mathbf{\Lambda}}+\mathbf{F}\circ\left(\mathbf{\Phi}^{\intercal}\overline{\mathbf{\Phi}}\right)-\frac{1}{2}\mathbf{I}\circ\left(\mathbf{\Phi}^{\intercal}\overline{\mathbf{\Phi}}\right)\right)\mathbf{\Phi}^{\intercal},\\
\overline{\mathbf{K}}&=-\mathbf{\Phi}\left(\mathbf{\Lambda}\overline{\mathbf{\Lambda}}+\mathbf{F}\circ\left(\mathbf{\Phi}^{\intercal}\overline{\mathbf{\Phi}}\right)\right)\mathbf{\Lambda}\mathbf{\Phi}^{\intercal}.
\end{aligned}\tag{B.27}$$

B.6 Derivation of Eq. 5.27

The derivation of the truncation error Eq. (5.27) is as follows. The full basis RAD result given in Eq. (5.24) can be written as a combination of the reduced basis denoted by $\hat{(\,)}$, and the truncated

basis denoted by $(\tilde{\ })$ as,

$$
\begin{aligned}
\overline{\mathbf{M}} =& \begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{\Lambda}} & 0\\ 0 & \tilde{\mathbf{\Lambda}}\end{bmatrix}\begin{bmatrix}\overline{\hat{\mathbf{\Lambda}}} & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{F}} & \tilde{\mathbf{F}}_1\\ \tilde{\mathbf{F}}_2 & \tilde{\mathbf{F}}_3\end{bmatrix} \circ \left(\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\begin{bmatrix}\overline{\hat{\mathbf{\Phi}}} & 0\end{bmatrix}\right)\right)\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\\
&- \frac{1}{2}\begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{I}} & 0\\ 0 & \tilde{\mathbf{I}}\end{bmatrix} \circ \left(\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\begin{bmatrix}\overline{\hat{\mathbf{\Phi}}} & 0\end{bmatrix}\right)\right)\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix},\\
=& \begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}} & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{F}} & \tilde{\mathbf{F}}_1\\ \tilde{\mathbf{F}}_2 & \tilde{\mathbf{F}}_3\end{bmatrix} \circ \begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}} & 0\\ \tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}} & 0\end{bmatrix}\right)\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\\
&- \frac{1}{2}\begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{I}} & 0\\ 0 & \tilde{\mathbf{I}}\end{bmatrix} \circ \begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}} & 0\\ \tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}} & 0\end{bmatrix}\right)\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix},\\
=& \begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}} & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{F}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right) & 0\\ \tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right) & 0\end{bmatrix}\right)\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\\
&- \frac{1}{2}\begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{I}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right) & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix},\\
=& \hat{\mathbf{\Phi}}\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}}\hat{\mathbf{\Phi}}^\intercal + \left(\hat{\mathbf{\Phi}}\left(\hat{\mathbf{F}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^\intercal + \tilde{\mathbf{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^\intercal\right) - \frac{1}{2}\hat{\mathbf{\Phi}}\left(\hat{\mathbf{I}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\hat{\mathbf{\Phi}}^\intercal\right),\\
=& \left(\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}} + \hat{\mathbf{F}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^\intercal - \frac{1}{2}\hat{\mathbf{\Phi}}\left(\hat{\mathbf{I}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\hat{\mathbf{\Phi}}^\intercal\right)\right) + \tilde{\mathbf{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^\intercal,\\
=& \overline{\hat{\mathbf{M}}} + \Delta\overline{\mathbf{M}}.
\end{aligned}
\tag{B.28}
$$

Comparing the last line with Eq. (5.25), the expression for the $\Delta\overline{\mathbf{M}}$ truncation error is

$$
\Delta\overline{\mathbf{M}} = \tilde{\mathbf{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Phi}}^\intercal. \tag{B.29}
$$

We conduct a similar derivation for $\Delta\overline{\mathbf{K}}$. The full basis RAD expression for $\mathbf{K}$ is,

$$
\begin{aligned}
\overline{\mathbf{K}} =& -\begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\mathbf{\Lambda} & 0\\ 0 & \tilde{\mathbf{\Lambda}}\end{bmatrix}\begin{bmatrix}\overline{\hat{\mathbf{\Lambda}}} & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{F}} & \tilde{\mathbf{F}}_1\\ \tilde{\mathbf{F}}_2 & \tilde{\mathbf{F}}_3\end{bmatrix} \circ \left(\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix}\begin{bmatrix}\overline{\hat{\mathbf{\Phi}}} & 0\end{bmatrix}\right)\right)\begin{bmatrix}\mathbf{\Lambda} & 0\\ 0 & \tilde{\mathbf{\Lambda}}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix},\\
=& -\begin{bmatrix}\hat{\mathbf{\Phi}} & \tilde{\mathbf{\Phi}}\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}} & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}\hat{\mathbf{F}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right) & 0\\ \tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right) & 0\end{bmatrix}\right)\begin{bmatrix}\hat{\mathbf{\Lambda}} & 0\\ 0 & \tilde{\mathbf{\Lambda}}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{\Phi}}^\intercal\\ \tilde{\mathbf{\Phi}}^\intercal\end{bmatrix},\\
=& -\hat{\mathbf{\Phi}}\left(\hat{\mathbf{\Lambda}}\overline{\hat{\mathbf{\Lambda}}} + \hat{\mathbf{F}} \circ \left(\hat{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^\intercal + \left(-\tilde{\mathbf{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^\intercal\right),\\
=& \overline{\hat{\mathbf{K}}} + \Delta\overline{\mathbf{K}},
\end{aligned}
\tag{B.30}
$$

Comparing this last line with Eq. (5.25), the expression for the truncation error $\Delta\overline{\mathbf{K}}$ is

$$
\Delta\overline{\mathbf{K}} = -\tilde{\mathbf{\Phi}}\left(\tilde{\mathbf{F}}_2 \circ \left(\tilde{\mathbf{\Phi}}^\intercal\overline{\hat{\mathbf{\Phi}}}\right)\right)\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}}^\intercal. \tag{B.31}
$$

This concludes our derivation for the truncation error presented in Eq. (5.27).

B.7 Derivation of Eq. 5.28

The following FAD derivation is due to Lim et al. [76] and Wang [141]:

$$
\begin{aligned}
\dot{\boldsymbol{\phi}}_i &= \sum_{l=1,l\neq i}^{n} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right), \\
&= \sum_{l=1,l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} \\
&\quad + \sum_{l=r+1}^{n} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right), \\
&\approx \sum_{l=1,l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} \\
&\quad + \sum_{l=r+1}^{n} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right), \\
&= \sum_{l=1,l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} \\
&\quad - \sum_{l=1}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} + \sum_{l=1}^{n} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} \\
&\quad - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right), \\
&= \sum_{l=1,l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} \\
&\quad - \sum_{l=1}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} + \sum_{l=1}^{n} \boldsymbol{\phi}_l\lambda_l\boldsymbol{\phi}_l^\intercal(-\dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{1}{\lambda_i} \\
&\quad - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right), \\
&= \sum_{l=1,l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i-\lambda_l} \\
&\quad - \sum_{l=1}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} + \mathbf{K}^{-1}(-\dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{1}{\lambda_i} \\
&\quad - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right),
\end{aligned}
\tag{B.32}
$$

where $i = 1, \ldots, r$. From the second to third equality we have used the assumption that $\lambda_i \gg \lambda_l$ where $l = r+1, \ldots, n$. Thus,

$$\frac{\lambda_l}{\lambda_i - \lambda_l} \approx \frac{\lambda_l}{\lambda_i}. \tag{B.33}$$

In the last equality of Eq. (B.32), we use the following result,

$$\mathbf{K}^{-1} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\intercal. \tag{B.34}$$

which can be obtained by manipulating Eq. (5.1)

From the above derivation, we have shown that

$$\begin{aligned}\dot{\boldsymbol{\phi}}_i = &\sum_{l=1, l\neq i}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i - \lambda_l} \\ &- \sum_{l=1}^{r} \boldsymbol{\phi}_l(-\boldsymbol{\phi}_l^\intercal \dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \boldsymbol{\phi}_l^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{\lambda_l}{\lambda_i} + \mathbf{K}^{-1}(-\dot{\mathbf{K}}\boldsymbol{\phi}_i\lambda_i + \dot{\mathbf{M}}\boldsymbol{\phi}_i)\frac{1}{\lambda_i} - \frac{1}{2}\boldsymbol{\phi}_i\left(\boldsymbol{\phi}_i^\intercal \dot{\mathbf{M}}\boldsymbol{\phi}_i\right).\end{aligned} \tag{B.35}$$