Aerodynamic Optimization Using High-fidelity Computational Fluid Dynamics

Hamidreza Karbasian

A thesis
In the Department
of
Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Mechanical Engineering) at
Concordia University
Montréal, Québec, Canada

August 2021

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Hamidreza Karbasian**

Entitled: **Aerodynamic Optimization Using High-fidelity Computational Fluid Dynamics**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Arash Mohammadi _____ Chair

Dr. Charles Audet _____ External Examiner

Dr. Biao Li _____ Examiner

Dr. Marius Paraschivoiu _____ Examiner

Dr. Carole El Ayoubi _____ Examiner

Dr. Brian Vermeire _____ Supervisor

Approved _____

Dr. Ivan Contreras,
Graduate Program Director

08/24/2021 _____

Dr. Mourad Debbabi,
Gina Cody School of Engineering and Computer Science

Abstract

Aerodynamic Optimization Using High-fidelity Computational Fluid Dynamics

Hamidreza Karbasian, Ph.D.
Concordia University, 2021

In this study, we demonstrate the ability to perform large-scale PDE-constrained optimizations using Large Eddy Simulation (LES). We first outline the challenges associated with performing gradient-based optimization using LES, specifically chaotic divergence of the sensitivity functions. We then demonstrate that shape optimization using LES and Mesh Adaptive Direct Search Method (MADS) is feasible for aerodynamic design. Next, we introduce a Dynamic Polynomial Approximation (DPA) procedure, which allows the high-order solution polynomial representation used by the flow solver to be increased, or decreased, depending on the poll size being used by MADS. This allows rapid convergence towards the optimal design space using lower-fidelity simulations, followed by an automatic transition to higher-fidelity simulations when close to the optimal design point. Additionally, this study proposes a new physics-constrained data-driven approach for sensitivity analysis and uncertainty quantification of large-scale chaotic dynamical systems. Unlike conventional sensitivity analysis, the proposed approach can manipulate the unsteady sensitivity function (i.e., tangent) for PDE-constrained optimizations. In this new approach, high-dimensional governing equations from physical space are transformed into an unphysical space (i.e., Hilbert space) to develop a closure model in the form of a Reduced-Order Model (ROM). Afterward, a new data sampling approach is proposed to build a data-driven approach for this framework. To compute sensitivities, Least-Squares Shadowing (LSS) minimization is applied to the ROM. It is shown that the proposed approach can capture sensitivities for large-scale chaotic dynamical systems, where Finite Difference (FD) approximations fail. Therefore, we expect that implementing the proposed optimization approach can be applied to large-scale chaotic problems, such as turbulent flows, and this approach significantly reduces computational cost and data storage requirements.

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Brian Vermeire, for his support, help and guidance throughout this research study. He showed me the way and continuously encouraged me to proceed with this project. Moreover, he is a role model for his students, and I am honoured to be his student.

Also, I would like to acknowledge our laboratory fellows. I should thank my friends Siavash, Ramin, Mohsen, Jie, and other friends who helped me with technical stuff and notable hints throughout this research study. A special salute belongs to Carlos for his support and dedication.

I would love to thank Dr. Javad Abolfazli Esfahani for his continuous support in each chapter of my life. His dedication has always been with me, and I am still leveraging many valuable things from his guidance. Moreover, it was my pleasure to meet Dr. Mahdi Bahadori, who showed me how to love all creatures. He showed me how a person could fly in the soar of happiness.

I want to acknowledge the financial support from the National Science and Engineering Research Council of Canada (NSERC) as well as school of graduate study at Concordia University for awarding fellowships.

Last but not least, I sincerely would like to thank my supportive wife, Zahra, whose endless love, patience, encouragement inspired me to complete this academic milestone. I should also remark on the littlest supportive, Nina, whose existence pushed me never to stop in this journey and do my best for a better future.

من به سرچشمه خورشید نه خود بردم راه

ذره ای بودم و مهر تو مرا بالا برد

I could not reach the source of light, and your true love raised me up!

To the best father, Mahdi, who illuminates the universe,
To my lovely wife Zahra,
& To my daughter Nina

Contents

List of Figures

List of Tables

List of Symbols

The following list describes symbols used later within the body of this thesis:

$\acute{\mathcal{Q}}_v$	Matrix of generalized coordinates for unsteady sensitivity solution
$\acute{\mathcal{Q}}_w$	Matrix of generalized coordinates for unsteady adjoint
$\acute{\mathbf{X}}_{adj}$	unsteady adjoint dataset
$\acute{\mathbf{X}}_{sens}$	unsteady sensitivity dataset
$\acute{\tilde{\mathbf{\Phi}}}_v$	Reduced-order basis for unsteady sensitivity solution
$\acute{\tilde{\mathbf{\Phi}}}_w$	Reduced-order basis for unsteady adjoint solution
$\acute{m}_p$	Total number of spamles for unsteady sensitivity solutions.
$\acute{r}_{sens}$	Rank of ROB for unsteady sensitivity
α_{eff}	Effective angle of attack (AOA).
α_{lss}	Weighting factor of time dilation
β	Weighting factor
χ	Computaitonal space
χ'	Referential computational space
Δt	Discrete time-step
Δ_m	Mesh size parameter

Δ_p Poll size parameter

δ_v Kronecker delta

$\ell(\xi)$ Nodal basis functions

η Time dilation

γ Ratio of specific heat

Γ^+ Counter clock-wise Vortex

Γ^- Clock-wise Vortex

$\hat{\lambda}$ Lagrange multiplier of adjoint solution

$\hat{\mathcal{R}}$ Vector of residual in ROM (backward solution)

$\hat{\boldsymbol{r}}$ Residual of the backward sensitivity function

$\hat{\tilde{\boldsymbol{\Psi}}}$ Test basis function for adjont equation

Λ Lagrange multiplier

λ Largrange multiplier in Hilbert space

$\mathbb{F}$ Actual function in Hilbert space

$\mathbb{N}_{p'}$ Set of time intervals for unsteady sensitivity solutions in *building* phase.

$\mathbb{N}_p$ Set of time intervals for steady-state sensitivity solutions in *building* phase.

$\mathbb{N}_q$ Set of time intervals in *shaping* phase.

$\mathbb{N}_s$ Set of time intervals in *hyper-reduction* phase.

$\mathbb{P}$ Mapping function

$\mathbb{T}$ Time domain

$\boldsymbol{\Phi}$ Trial basis function

$\boldsymbol{\Phi}'$ Fine-scale trial basis function

$\mathbf{\Psi}$	Test basis function
$\mathbf{\Psi}'$	Fine-scale test basis function
$\mathbf{\Theta}_v$	Linear operator in *shaping* phase (forward solution)
$\mathbf{\Theta}_w$	Linear operator in *shaping* phase (backward solution)
$\mathcal{B}$	Primary matrix of KKT system
$\mathcal{C}$	Secondary matrix of KKT system
$\mathcal{D}$	Vector of KKT system
$\mathcal{E}$	Time dilation of the generalized coordinates
$\mathcal{F}$	flux in ROM
$\mathcal{I}$	Identity matrix
$\mathcal{J}$	Objective function
$\mathcal{K}$	Koopman operator
$\mathcal{L}$	Lagrange function
$\mathcal{M}$	Mesh points in design space
$\mathcal{P}$	Trial points in design space
$\mathcal{Q}_u$	Matrix of generalized coordinates for state
$\mathcal{Q}_v$	Matrix of generalized coordinates for steady-state sensitivity solution
$\mathcal{Q}_w$	Matrix of generalized coordinates for steady-state adjoint solution
$\mathcal{R}$	Vector of residual in ROM (forward solution)
$\mathcal{S}$	Set of design parameter
$\mathcal{W}$	Weighting function in Petrov-Galerkin approach
$\mathcal{X}$	Kinematics of a wing section

$\mathfrak{C}$ Covariant Lyapunov Exponent

$\mathfrak{E}$ Eigenvalue

$\mathfrak{L}$ Lyapunov Exponent (LE)

$\mathscr{D}$ Design space

$\mathscr{H}$ Hilbert space

$\mathscr{L}$ Linear space

$\mathscr{N}$ Non-linear space

$\mathscr{P}$ Physical space

μ Kinematic viscosity

μ_d Dynamic viscosity (μ/ρ)

$\nabla.\mathcal{H}$ Hamiltonian dynamical system

$\mathcal{V}$ Volume

ν Normalized vector in MADS

Ω Source term

$\overline{\boldsymbol{u}}$ Vector of reference state

$\overline{\boldsymbol{v}_{ss}}$ Vector of reference steady-state sensitivity solution

$\overline{\boldsymbol{v}_{us}}$ Vector of reference unsteady sensitivity solution

$\overline{\boldsymbol{v}}$ Vector of reference sensitivity solution

$\overline{\boldsymbol{w}_{ss}}$ Vector of reference steady-state adjoint solution

$\overline{\boldsymbol{w}_{us}}$ Vector of reference unsteady adjoint solution

$\overline{C_{L,rms}}$ Root-Mean-Square of Lift coefficient

$\overline{C}_{L,target}$ Desired Lift coefficient in optimization

ϕ	Vector of trial basis function
ϕ_x	Vector of trial basis function for $x-$momentum
ψ	Vector of test basis function
ρ	Density
ρ_∞	Free-stream density
τ	Time transformation
τ_v	Viscous stress tensor
$\boldsymbol{f}$	flux
$\boldsymbol{f}_s$	Perturbed flux
$\boldsymbol{f}_c^{'}$	Corrected Riemann flux
$\boldsymbol{f}_i$	Inviscid Navier-Stokes flux
$\boldsymbol{f}_i^{'\delta_m}$	Elementwise polynomial representation of the flux
$\boldsymbol{f}_l^{'}$	Riemann flux of the left-hand side of the element
$\boldsymbol{f}_r^{'}$	Riemann flux of the right-hand side of the element
$\boldsymbol{f}_v$	Viscous Navier-Stokes flux
$\boldsymbol{h}$	Vector of sensitivity solution of generalized coordinates.
$\boldsymbol{h}'$	Fine-scale sensitivity function of generalized coordinates
$\boldsymbol{q}$	Generalized coordinates
$\boldsymbol{q}'$	Fine-scale generalized coordinates
$\boldsymbol{r}$	Residual vector
$\boldsymbol{r}_{gn}$	Cost function in Gauss-Newton method
$\boldsymbol{u}$	Vector of state

$\boldsymbol{u}'$ Vector of fine-scale state

$\boldsymbol{u}_s$ Vector of perturbed state

$\boldsymbol{u}_i^{'\delta_m}$ Elementwise polynomial representation of the solution

$\boldsymbol{v}$ Sensitivity solution

$\boldsymbol{w}$ Vector of adjoint solution

$\mathbf{A}$ Weighting matrix

$\mathbf{C}$ Constraint function

$\mathbf{C}_{bound}$ Bound constraint function

$\mathbf{C}_{geom.}$ Geometrical constraint function

$\mathbf{D}$ Matrix of directions in design space

$\mathbf{H}$ House-holder matrix

$\mathbf{K}$ Weighting matrix

$\mathbf{U}$ Matrix of states

$\mathbf{V}_{ss}$ Matrix of steady-state sensitivity solutions

$\mathbf{V}_{us}$ Matrix of unsteady sensitivity solutions

$\mathbf{W}_{ss}$ Matrix of steady-state adjoint solutions

$\mathbf{W}_{us}$ Matrix of unsteady adjoint solutions

$\mathbf{X}_{adj}$ Steady-state adjoint dataset

$\mathbf{X}_{sens}$ Steady-state sensitivity dataset

$\mathbf{X}_{state}$ State dataset

$\mathrm{x}(t)$ Steamwise motion

x_{cp} Center of pressure

$\mathrm{y}(t)$ Plunging motion

y_0 Plunging amplitude

$\mathrm{z}(t)$ Spanwise motion

$\theta(t)$ Pitching motion

θ_0 Pithing amplitude

θ_{corr} scaling factor of a manifold

θ_m Mean pitching angle

θ_s Shift angle between the plunging and pitching motions

$\tilde{\hat{\boldsymbol{\Phi}}}_u$ Reduced-order basis for state

$\tilde{\hat{\boldsymbol{\Phi}}}_v$ Reduced-order basis for steady-state sensitivity solution

$\tilde{\hat{\boldsymbol{\Phi}}}_w$ Reduced-order basis for steady-state adjoint solution

$\tilde{\boldsymbol{\Phi}}$ Coarse-scale trial basis function

$\tilde{\boldsymbol{\Psi}}$ Coarse-scale test basis function

$\tilde{\boldsymbol{a}}$ Coarse-scale adjoint solution in Hilbert space

$\tilde{\boldsymbol{a}}$ Generalized coordinates of the adjoint solution

$\tilde{\boldsymbol{a}}_c$ Corrected adjoint solution of the generalized coordinates

$\tilde{\boldsymbol{h}}$ Coarse-scale sensitivity function of generalized coordinates

$\tilde{\boldsymbol{h}}_c$ Corrected sensitivity of the generalized coordinates

$\tilde{\mathbf{q}}$ Coarse-scale generalized coordinates

$\tilde{\mathbf{u}}$ Vector of coarse-scale state

ξ Index of time intervals

ξ Nodal basis point

$\xi_{i,p}$	Set of nodal basis point
$\xi_{p'}^{(n)}$	Index of time interval in $\mathbb{N}_{p'}$
$\xi_{p}^{(n)}$	Index of time interval in $\mathbb{N}_{p}$
$\xi_{q}^{(n)}$	Index of time interval in $\mathbb{N}_{q}$
$\xi_{s}^{(n)}$	Index of time interval in $\mathbb{N}_{s}$
ζ	Weighting factor
B_l	Lower limits of bounds
B_u	Upper limits of bounds
c	Chord length
C_p	Pressure coefficient
c_p	Specific heat at constant pressure
c_v	Specific heat at constant volume
C_L	Lift coefficient
C_n	Normal force coefficient
C_P	Power coefficient
C_t	Tangential force coefficient
$d\tau$	Time transformation step
dt	Time interval
e	Total energy
E_p	Propulsive efficiency
f_s	Frequency
f_a	Flapping frequency

F_D Drag

F_L Lift

F_n Normal force

F_T Thrust

F_t Tangential force

h Component of vector of sensitivity solution

k Number of steps in time integrator

k_a Reduced frequency

l_c Characteristic length

m_q Number of samples in *shaping* phase

M_∞ Free-stream Mach number

m_{samp} Number of sample intervals

m_k Total number of T_s

m_p Total number of steady-state samples

m_u Total number of time step

n_+ list of indices at which LEs are positive

n_0 list of indices at which LEs are zero

n_D Number of direction defined in design space

n_d Number of design sets

n_e Number of elements in domain

n_s Nuumber of design parameters

n_u Dimension of state

n_{dp}	Total number of design sets
n_l	Number of leading LEs
n_w	Total number of the objective functions
p	Pressure
p_∞	Ambient pressure
p_s	Polynomial degree
Pr	Prandtl number
q_v	Heat flux
r	Rank
r_h	Truncation rank for hyper-reduction
r_{sens}	Rank of ROB for steady-state sensitivity solution
r_{state}	Rank of ROB for state
Re	Reynolds number
s	Design parameter
St	Strouhal number
t	Time
t^*	Convective time
T_s	Time interval for calculation of LEs
u_i	Velocity component
u_∞	Free-streem velocity
x_i	Space direction
$x_{i,p}$	Set of nodal solution point

Chapter 1

Introduction

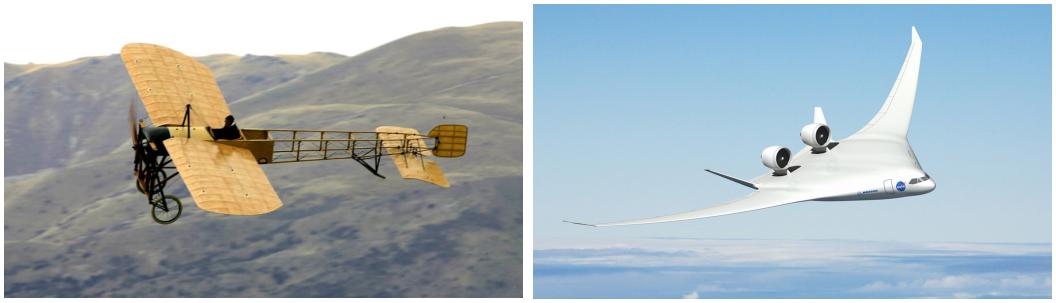
1.1 Overview

The aerospace industry can be considered as one of the highest-impact fields in history, significantly changing human behaviour over the last century. Air transportation has paved the way for easier and faster travel worldwide. With the advent of powerful computational tools in the fields of control systems, avionics, and conceptual aircraft design, we see continuous improvements in air transportation every year. As we reflect on the history of aircraft design over time, aircraft topology has changed significantly. These changes arise as modifications to the fuselage, wings, stabilizers, and related systems. For instance, engineers design aircraft wings that produce higher lift, reduced drag, and to perform more efficiently at high-speeds. Moreover, the fuselage is designed to reduce drag, or in next-generation designs, to produce additional lift. Figure 1a shows a classical aircraft, the Bleriot XI, built in 1909 and restored in 2006 to fly in the Wanaka International Airshow in Lake Wanaka, New Zealand [1]. Figure 1b shows a next-generation commercial aircraft proposed by NASA in 2012 [2]. Notable changes between these two aircraft imply a continuous optimization process performed by engineers over the past century to improve aircraft performance. To accelerate this process, advanced optimization tools are now required to develop the next-generation of more efficient aircraft.

By 2030 there will be an estimated global demand for approximately 27,000 new commercial aircraft [127]. These aircraft must comply with current and forthcoming

[1] Wanaka International Airshow - http://www.warbirdsoverwanaka.com

[2] Greener aircraft - https://www.nasa.gov/topics/aeronautics/



(a) Classical aircraft (b) Future aircraft

Figure 1: History of aircraft technologies - past and future.

emission regulations, which require a $20 - 25\%$ reduction in fuel consumption [127]. Hence, aerodynamic optimization of next-generation cleaner aircraft remains a significant challenge in the aerospace community [128]. This is because aerodynamic optimization is particularly difficult due to non-linear coupling between the design parameters, such as the aircraft shape, and objective functions, such as lift and drag coefficients. In general terms, aerodynamic optimization involves finding the optimal set of design parameters that minimize the chosen objective function. Hence, aerodynamic optimization applies to a wide range of applications because of the general nature of the problem and the range of different aircraft operating requirements. Choosing a suitable optimization strategy is dependent on the behaviour of the objective function with respect to these design parameters [151]. Furthermore, optimization strategies should efficiently utilize available computational resources to minimize the computational cost of the optimization process [138]. It is worth mentioning that optimization with multiple objective functions, or multiple design parameters, is known as multi-objective or multi-disciplinary optimization, respectively. Often, for aerodynamics applications, multi-disciplinary optimization is of interest [101, 129, 142]. Additionally, time-averaged objective functions are used in unsteady flow problems, where mean quantities such as lift and drag coefficients, are of interest. Optimization is usually an iterative process. The design parameters are updated at each step based on the constraints and information from previously obtained objective functions. It is important to note that optimization strategies do not generally guarantee reaching the global optimum, as they may also stagnate at local optima [51, 112, 123, 130].

The derivative of an objective function is zero at local optima for unconstrained optimizations. However, there are exceptions to this statement when constraints are

applied. Therefore, finding derivatives of the objective function with respect to each design parameter can be used to identify locations of these local optima in the design space, but we should always check the derivatives during the optimization method to ensure the existence of any exceptions. In engineering design, sensitivity analysis has been widely used to improve designs or understand to developing better control systems. Sensitivity analysis finds the derivative of the objective function, as an output of the system, with respect to perturbed inputs. Conventional sensitivity analysis, such as the adjoint method, is widely used for steady-state problems [127]. However, it breaks down when applied to time-averaged quantities in chaotic systems, due to the chaotic dependence of turbulent flows on initial conditions (commonly referred to as the *Butterfly Effect*) [18]. Hence, computing these sensitivities becomes challenging. This is because small perturbations to the initial flow field result in chaotic divergence of the instantaneous flow, and introduces noise in the objective function under finite time-averaging. Several methods have been proposed to overcome this problem [139]. However, each approach has several shortcomings, such as expensive computational costs, and the unreliability of the computed results with high uncertainties.

1.2 Literature Review

Optimization arises in a wide range of applications from engineering design [15, 116, 121] to control systems [78, 85]. Increasing computational power has recently enabled the application of optimization techniques for industrial scale problems. These problems are usually governed by large non-linear Partial Differential Equations (PDEs) with suitable initial and boundary conditions, which are often referred to as Full-Order Models (FOMs). These FOMs arise in computational mechanics applications, such as Computational Fluid Dynamics (CFD) [94, 98, 105]. In practice, these FOMs act as a constraint during optimization, referred to as PDE-constrained optimization. This type of optimization can be prohibitively expensive, since it requires one or more high-fidelity FOM solutions per design iteration [144, 149]. Solving multiple large-scale FOMs using high-fidelity solvers in the context of CFD, such as Large-Eddy Simulation (LES) and Direct Numerical Simulation (DNS), can be prohibitively expensive [86]. Therefore, the majority of optimizations are performed using lower-fidelity models, such as the Reynolds-Averaged Navier-Stokes (RANS)

approach.

In aerodynamic design, optimization can be categorized into two distinct methods, gradient-based and gradient-free optimization. Gradient-based optimization requires the sensitivity of the objective function with respect to the design parameters. Given this, the optimizer is then guided towards a superior design using this sensitivity via gradient-based techniques [35, 68, 107]. Gradient-based optimization using the adjoint is widely used in aerospace design, since it has a relatively low computational cost when handling a large number of design parameters [84, 87, 143]. Pironneau [111] applied the adjoint technique in fluid dynamics and showed that the cost of computing sensitivities is independent of the total number of design parameters. Therefore, the adjoint is well suited for shape design optimization with a large number of design parameters. Gradient-free optimization is a relatively robust and flexible design strategy. It increases the likelihood of finding a globally optimal solution [48, 83, 84]. Gradient-free methods are often more suitable for noisy objective functions, since they do not require sensitivities. However, the main drawback of these methods is that they are relatively expensive when the number of design parameters is large. Example algorithms for gradient-free optimization include the Genetic Algorithm (GA) [51, 123, 126] and Particle Swarm Optimization (PSO) approaches [113, 140, 150]. These population-based algorithms exploit evolutionary principles to move towards a more optimal design [126]. They can handle non-linear, non-convex and non-continuous problems [56]. However, studies have shown that while the GA can quickly determine the region of optimal designs, it is then slow to find the true optimal point [151]. In PSO, an improper population distribution can lead to poor performance and a possible failure to find the optimal design [53]. Furthermore, these methods require a large number of simulations per design iteration.

From a fluid dynamics perspective, resolving the entire turbulent energy cascade is computationally expensive. This approach, called Direct Numerical Simulation (DNS), is primarily used for some scientific applications. In contrast, the Reynolds Averaged Navier-Stokes (RANS) approach solves for a statistical time-averaged flow, and models all the turbulent length and time scales. RANS is computationally faster, but has limited accuracy due to its reliance on approximate turbulence models. Situated between DNS and RANS, Large-Eddy Simulation (LES) is an approach that resolves large-scale turbulent length scales, and models the smallest dissipative scales.

Figure 2 shows several scale-resolving simulations performed using LES. LES is a promising approach for engineering applications since it provides accurate results, but is computationally less expensive than DNS [16]. Hence, LES could become an essential tool for aerospace CFD, most notably for flows around aircraft, wings, or within jet engines. RANS solvers often fail to simulate relevant flow physics associated with these geometries. This is because flow and boundary layer features, such as transition and separation points, are strongly affected by chaotic non-linear interactions. Therefore, in many engineering designs, such as PDE-constrained optimization, RANS struggles with inaccuracy of relevant flow physics. This inaccuracy brings high uncertainty, leading to poor design. Therefore, scale-resolving simulations play an essential role in engineering design and optimization, since they provide more accurate flow physics for designs. However, there is, as yet, no suitable framework for efficient and reliable optimization using LES.

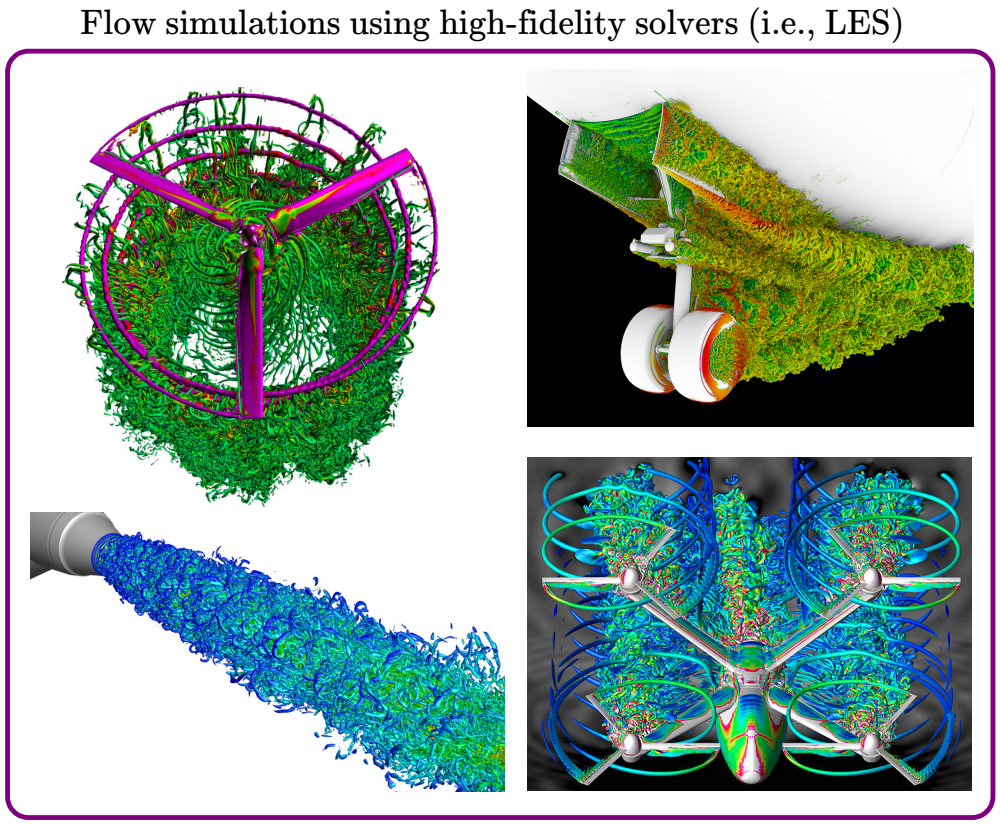


Figure 2: Different scale-resolving simulations using LES.

To alleviate the prohibitive cost of PDE-constrained optimization using FOMs,

such as LES, a wide range of studies have developed surrogate modelling techniques [80, 88, 89]. Surrogate models of non-linear PDEs are usually referred to as Reduced Order Models (ROMs) [26, 60, 104]. Dimensionality reduction using ROMs has become an attractive approach, and employed successfully in optimal control [14, 76], optimization [121, 148], and sensitivity analysis [27, 148, 149]. Optimization combined with surrogate models introduces a new approach called ROM-constrained optimization. Generally speaking, ROMs can be embedded within machine learning frameworks [23, 119, 132], and are based on a projection of FOMs from physical space into a lower-dimensional model space (i.e., Hilbert space, Eigenspace, Banach space, etc.) and vice versa [119, 132, 133]. This projection typically employs trial basis functions. These trial basis functions typically use matrix representations, which define the Reduced-Order Basis (ROB). For example, the ROB can be the eigenvectors or bases of a dataset obtained via Proper Orthogonal Decomposition (POD). ROMs leveraging machine learning features are usually constructed by collecting data initially obtained from FOMs [24, 25, 93]. After dimensionality reduction, other machine learning techniques can be applied to develop new models that can predict the behaviour of the dynamical system. Depending on the system's complexity, or non-linearity, hidden layers can be used to build deep learning algorithms.

1.3 Motivation

Figure 3 is provided to illustrate the current state of aerodynamic optimization. RANS, Unsteady RANS (URANS), and potential flow solvers are low-fidelity and well-established for analysis with manually predefined geometry and boundary conditions. These can be coupled with forward/backward sensitivity functions for design and optimization purposes. Higher-fidelity simulations, such as DNS, LES, and Detached Eddy Simulation (DES), are now becoming more prevalent, due to algorithmic improvements and advances in computing power.

However, sensitivity functions obtained using LES or DNS are unconditionally unstable due to chaotic divergence. Therefore, optimization using high-fidelity CFD simulations is currently impossible. Therefore, aerospace design and existing optimization techniques are currently limited to low-fidelity CFD. Figure 4 displays example of aerodynamic designs performed using PDE-constrained optimization. RANS

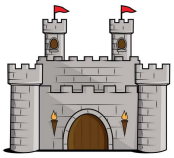
Computational Tools *Flow Solvers*	Analysis *Single simulation with predefined geometry and boundary conditions*	Design *Multiple simulations with a large number of design parameters*
Low Fidelity RANS, URANS Potential flow	 Trivial	 Challenging
High Fidelity DNS, LES, DES	 Challenging	 Currently impossible

Figure 3: High-fidelity and low-fidelity simulations for analysis and design.

approach is required to stabilize the sensitivity functions of these large-scale optimizations. To clarify how a non-linear problem leads to divergence of the sensitivity function, Figure 5 is provided for the flow past a circular cylinder at the Reynolds number of 150. Sensitivity of the state vector with respect to the Reynolds number is provided. Figure 5a shows the instantaneous sensitivity of the $x-$momentun with respect to the Reynolds number at the beginning of the analysis. Figure 5b shows the moment instability errors propagate through the domain, due to divergence.

In optimization, chaotic dynamical systems usually yield noisy design spaces for a short period evaluation of the objective function, and these noises contaminate the sensitivities. Optimization in this design space could be convex or non-convex, with a certain level of noise. Figure 6 shows different approaches for optimization including benefits and limitations. Red boxes show the contribution of this work, and all acronyms will be defined later in the following chapters. Popular approaches for optimization are divided into two distinct categories. The first category is gradient-free methods, for which sensitivities are not required during the optimization procedure. These methods are computationally inexpensive when the number of design parameters is small. Genetic Algorithms (GA) [36, 145] and Heuristic Search methods, such as General Pattern Search (GPS) [22] and Mesh Adaptive Direct Search (MADS)

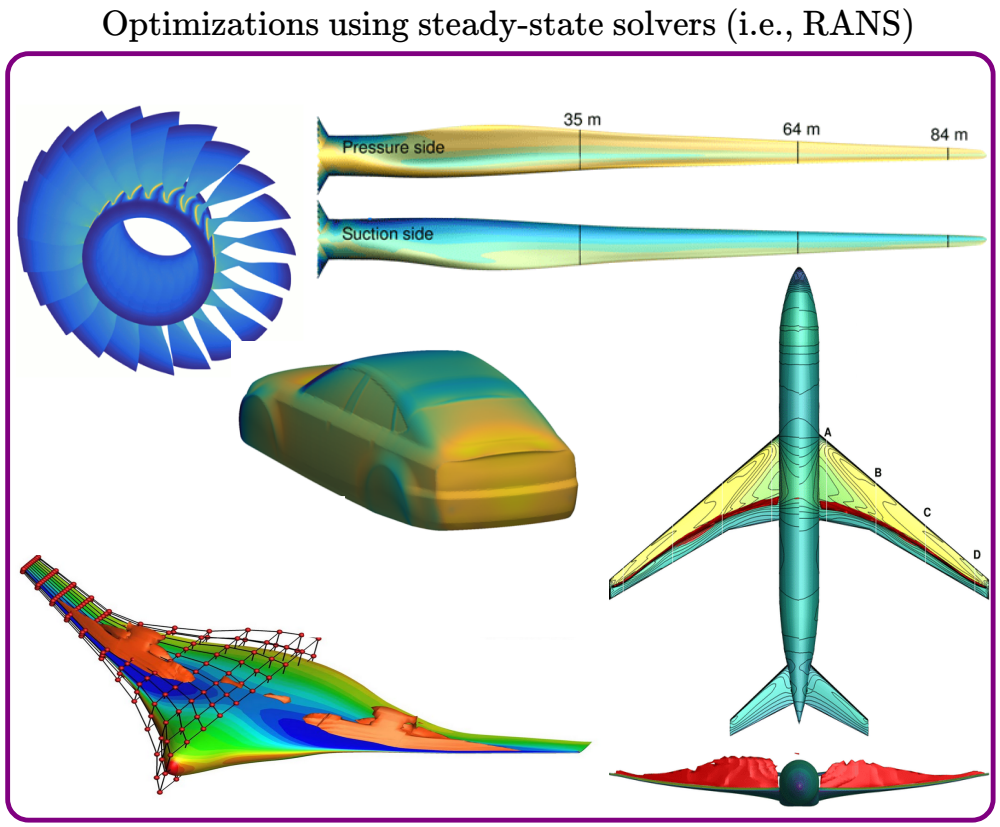
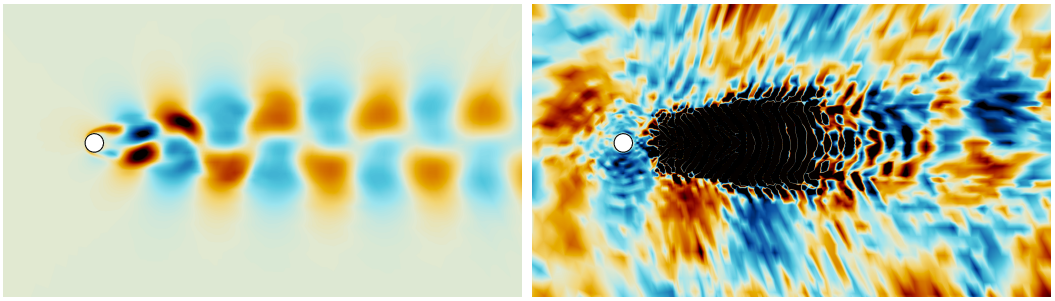


Figure 4: Different examples of PDE-constrained optimization done by RANS.

[4, 6, 86], are popular gradient-free optimization algorithms. MADS is a promising gradient-free optimization strategy. In each design cycle, it only requires the magnitude of the objective functions, not sensitivities, obtained with multiple design parameters [6]. However, unlike the adjoint method, when the number of design parameters is large, it becomes expensive since the required simulations in each design cycle increase. Therefore, applying gradient-free optimization for high-fidelity problems is limited to a relatively small number of design parameters.

In gradient-based methods, conventional optimization methods accompanied by sensitivity analysis (SA) commonly fail due to chaos. From a theoretical standpoint, the *shadowing lemma* describes the existence of a trajectory for an augmented dynamical system that stays uniformly close to the real trajectory of the unaugmented dynamical system [72]. By leveraging the *shadowing lemma*, Least Squares Shadowing (LSS) was recently proposed [18, 139], and enables computation of sensitivities for



(a) Beginning of analysis (b) End of analysis

Figure 5: Contours of sensitivity of $x-$momentun with respect to Re at the beginning and end of sensitivity analysis.

high-fidelity chaotic systems. However, LSS for large-scale PDEs is prohibitively expensive with available computing hardware [17]. The computational costs for LSS are approximately $10^4 \sim 10^5$ times more than those of the primal PDE solvers (i.e., DNS or LES) [20]. Recent formulations of LSS were developed to reduce computational costs. In Multiple Shooting Shadowing (MSS) [19], the number of solutions required in the discrete time domain is reduced, resulting in compacted linear systems. It was observed that MSS has lower memory usage and computational requirements than conventional LSS [19]. Although notable improvements were observed with MSS, it is still not applicable for large-scale PDEs. The Non-Intrusive Least Squares Shadowing (NILSS) approach [99] is another formulation for LSS, reducing computational costs by applying a minimization problem to unstable modes that exist in chaotic dynamical systems. In this method, the number of unstable modes in the dynamical system is determined via several high-fidelity simulations of primal PDEs with different initial conditions. In chaotic systems, the number of unstable modes may be large, which increases the computational cost of NILSS. For instance, sensitivity analysis using NILSS was considered for flow past a 3D circular cylinder at the Reynolds number of 525 [100]. In this example, the computational domain had 3.7×10^5 hexahedra elements. For sensitivity analysis, 40 primal CFD simulations were carried out to determine the number of unstable modes in the problem. Therefore, this sensitivity approach is also not an efficient tool for high-fidelity design.

Generally speaking, the *shadowing lemma* enables sensitivity analysis for chaotic dynamical systems, but has prohibitive computational cost. Previous research focused on reducing memory requirements and computational cost. The similarity of

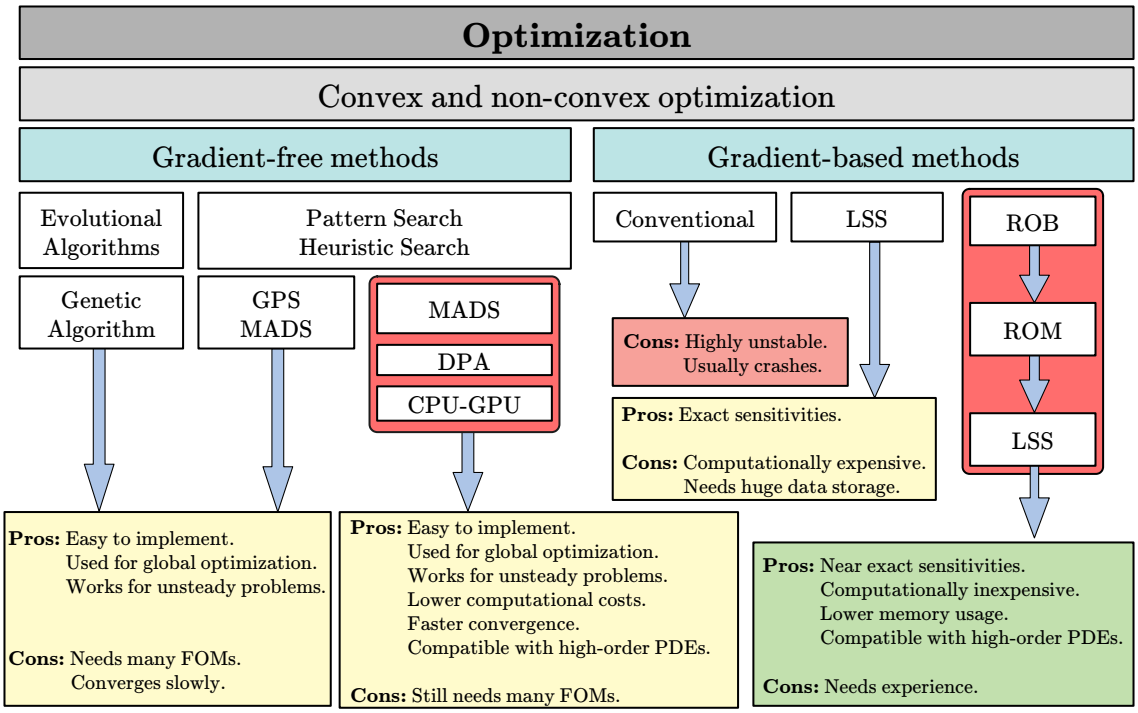


Figure 6: Different approaches for convex and non-convex optimization.

these attempts with model-reduction techniques is that both of them reduce the size of linear systems of equations. Therefore, dimensionality reduction could be an asset to solve LSS problems, which is the main concept of the second part of this study. Many classical ROMs are linear-based models developed by Galerkin projection. In these approaches, modal bases, corresponding to dominant and energetic structures in a dynamical system, are sorted in low-dimensional subspaces, and placed in a united matrix, called the trial basis function. With the help of this trial basis function, the FOM can be projected into low-dimensional subspaces [132]. While Galerkin projection has been employed successfully in many engineering problems, it often fails for non-linear systems due to the presence of unstable modes [30, 133]. This lack of stability results from the fact that *fine-scale* structures in non-linear dynamical systems play a significant role in the evolutionary behaviour of the dynamical system over long periods [133]. Closure models, such as the Least Squares Petrov-Galerkin (LSPG) [29, 30] and Adjoint Petrov-Galerkin (APG) [108], have shown particular promise for developing non-linear ROMs. In addition to dimensionality reduction, several approaches have been developed based on machine learning algorithms to model turbulent flows. Deep neural networks associated with trial basis functions

are able to develop accurate ROMs for complex flows at high Reynolds numbers [82]. Although machine learning has been successfully applied to scientific and engineering problems, it has not yet been used for sensitivity analysis and design. Conventional machine learning algorithms are used as black-box tools with large training datasets [24, 25, 93]. Their resulting ROMs cannot produce interpretable gradients (i.e., Jacobian). Furthermore, these models often yield non-physical results in off-design conditions [2, 44, 57], and are not suitable for sensitivity analysis or uncertainty quantifications.

1.4 Objectives & Scope of Thesis

In this thesis, we present several novel concepts to overcome the failure of high-fidelity sensitivity analysis and optimization. Using new mathematical frameworks, we propose several algorithms to obtain the sensitivities efficiently with high-fidelity CFD, while avoiding their divergence.

For convex and non-convex optimizations, we consider both gradient-free and gradient-based approaches. In the gradient-free approach, this study explores the utility of MADS for high-fidelity aerodynamic optimization. To the author's knowledge, there have not yet been any optimization studies done using high-fidelity flow simulations, such as LES and DNS. Since gradient-free optimization usually requires a large number of CFD simulations, we devised a set of numerical algorithms to reduce computational cost with this framework. To this end, the MADS executes optimization *tasks* on CPU cores in parallel, while each *task* launches its CFD simulations on GPU clusters. Furthermore, a method is proposed to find a proper time-domain (time-averaging window) to reduce noise in the design space within a specified threshold.

In the case of gradient-based approaches, we focus on sensitivity analysis, which is the primary challenge of this type of optimization. We propose a novel approach to compute time-averaged sensitivities of large-scale chaotic dynamical systems. We transform sensitivity functions from physical space into a lower-dimensional space (i.e., Hilbert space), and solve them using constraints that serve as optimality conditions. To this end, we develop a closure model in the form of a ROM using the weak form of the Navier-Stokes equations. This closure model is tied to the optimality

condition, and the residual of the ROM is minimized by searching optimal directions in subspaces. Furthermore, a novel physics-informed machine learning framework is developed to shape these manifolds in Hilbert space. This framework provides new algorithms for training the closure model, and computing sensitivities in the form of a boundary-value problem. This strategy can be achieved by applying LSS to the sensitivity functions in Hilbert space. The proposed approach has been developed within an in-house scientific software package, named OPtimization Toolkit for Highly NOn-linear Systems (OPTHiNOS).

1.5 Thesis Layout

This work is divided into eleven chapters, as follows

- Chapter 1 consists of the introduction, objectives, motivation of this work, and a brief literature review.
- Chapter 2 includes principal mathematical theories regarding chaotic dynamical systems. Also, different numerical schemes for spatial and temporal discretizations of the governing equations are discussed.
- Chapter 3 investigates the application of gradient-free optimization for non-convex problems, including novel strategies to reduce optimization costs for large-scale turbulent flows.
- Chapter 4 discusses sensitivity analysis in the presence of chaos. From a mathematical standpoint, the *shadowing lemma* is explained and applied to the general form of a chaotic dynamical system.
- Chapter 5 introduces methods to develop surrogate models using dimensionality reduction. It describes how to derive the weak form of high-dimensional PDEs, and convert them into lower-dimensional ODEs. It also covers the application of Galerkin and Petrov-Galerkin projection for reduced-order modelling.
- Chapter 6 describes gradient-based optimization for large-scale problems, with a new perspective for large-scale optimization. Sensitivity analysis using dimensionality reduction is also included and applied to complicated engineering problems.

- Chapter 7 includes the development of a physics-based machine learning framework to build surrogate models. Different strategies for data sampling are described, and steps are provided to devise feed-forward auto-encoders.

- Chapter 8 describes platforms, structures, and algorithms taken into account for sensitivity analysis. Each algorithmic step is described in detail, and optimization in the presence of constraints is discussed.

- Chapter 9 consists of numerical examples of sensitivity analysis. In this chapter, the proposed approaches are employed for different optimization problems, including non-chaotic and chaotic cases.

- Chapter 10 includes the application of PDE-constrained optimization to large-scale non-linear problems. This chapter tackles several case studies previously impossible due to failure of conventional sensitivity frameworks.

- Chapter 11 contains a summary, conclusions, and recommendations for future work.

Chapter 2

Chaotic Dynamical Systems

2.1 Dynamical System

A dynamical system is defined as a physical phenomenon that evolves with time. Any dynamical system consists of two things: (1) a set of *states* reflecting features or variables in a system that evolves with time, and (2) a set of *rules* for this evolution. Prominent examples of dynamical systems can be described in the form of differential equations in physical space $\mathscr{P}$, such that the dimensions of $\mathscr{P}$ are equal to the number of variables in the system (i.e., dimension of the state vector).

An arbitrary high-dimensional dynamical system, in the form of a Full-Order Model (FOM), is considered. The evolutional behaviour of this FOM can be described by a system of PDEs, with known initial and boundary conditions, such that the evolution of attractors occurs in a smooth time domain $\mathbb{T}$. Consider a FOM of dimension n_u that evolves with time $t \in \mathbb{T}$ with the initial value of $\boldsymbol{u}_0 \in \mathbb{R}^{n_u}$ at $t = 0$. This FOM can be represented as a set of primal PDEs

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f}(\boldsymbol{u}, \nabla\boldsymbol{u}, \mathcal{S}) = 0, \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \quad t \in \mathbb{T}, \tag{1}$$

where, $\boldsymbol{u} : \mathbb{T} \to \mathbb{R}^{n_u}$ is the state vector with dimension of n_u, and $\nabla.\boldsymbol{f}$ denotes divergence of the flux $\boldsymbol{f} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$ with $\boldsymbol{u} \mapsto \boldsymbol{f}(\boldsymbol{u})$. Moreover, $\mathcal{S} = [s_1, s_2, \ldots, s_{n_s}]^\top \subset \mathscr{D}$ is a set of design parameters in the design space $\mathscr{D} \in \mathbb{R}^{n_s \times n_s}$, where n_s is the number of design parameters. Here $x^\top$ denotes transpose of variable x. Additionally, the dynamical behaviour of the primal PDE evolves over $\mathbb{T}$, such that $\nabla.\boldsymbol{f} : \boldsymbol{u} \times \mathbb{T} \to \boldsymbol{u}$.

A Hamiltonian system is a mathematical formalism describing the evolution of a dynamical system in $\mathscr{P}$. When the dynamical system lies in a non-linear space,

$\mathscr{N}$, the initial-value problem for this dynamical system can not be solved analytically. The advantage of a Hamiltonian system is that it transforms the non-linear dynamical system from $\mathscr{P}$ into a non-dimensional space, referred to as Hilbert space $\mathscr{H}$. This transformation provides some practical insights into the evolutionary behaviour of underlying physical modes. From a mathematical viewpoint, dynamical modes can be described geometrically. Some of these geometric objects are shown in Figure 7, and described as follows

- **Fixed point:** If a dynamical system is steady-state, it has a unique solution, shown by a fixed point for each state in phase space (i.e., Hilbert space $\mathscr{H}$).

- **Limit cycle:** When the system is quasi-steady, and the solution converges to a repetitive pattern over an infinite time domain, $t \in \mathbb{T} \to \infty$. Regardless of the initial condition, resulting states create a cyclic pattern in phase space, indicating these states vary periodically.

- **Attractor:** The states for a non-linear system converge to a regime in phase space, where trajectories evolve around a dense orbit. This dense orbit attracts these states to itself when $t \in \mathbb{T} \to \infty$. Since none of the trajectories for any specified period in phase space are the same, the attractor can typically be shown as a surface or volume for 2D and 3D problems. Regardless of the initial condition, any trajectory in phase space is attracted by the same attractor over infinite time. When the dynamical system is chaotic, the attractor is called a *strange attractor*. An example of *strange attractor* is a butterfly-shaped set in the Lorenz system.

These geometric objects are helpful to recognize different types of dynamical systems, and when these systems have less complexity and lower-dimensionality. Additionally, in certain circumstances, a fixed point in a dynamical system turns into two different stable/unstable modes, where each mode exhibits notably different behaviours, referred to as *bifurcation*.

Dynamical systems can be evaluated more efficiently by considering a set of observables evolving with time. These observables are measurements from the state vector in physical space $\mathscr{P}$. Let us discuss an example: consider a 2D channel in which flow past a circular cylinder with constant boundary conditions (i.e., fixed free-stream velocity and pressure in farfield). The principal state of this system is the velocity

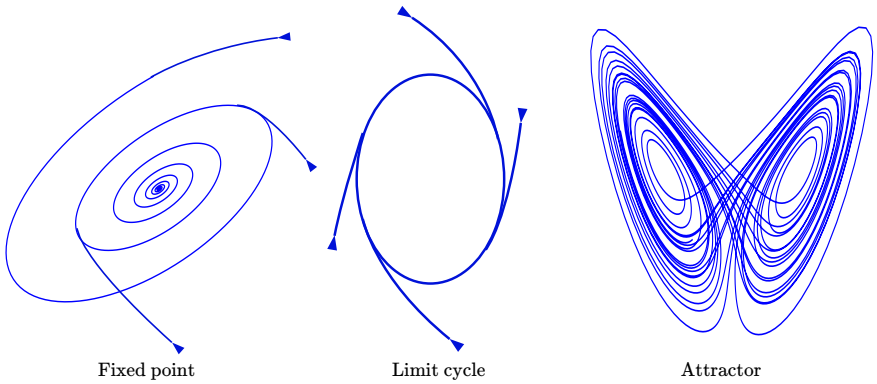


Figure 7: Schematic of geometric objects describing different dynamical systems.

components and pressure, and can be viewed as smooth surfaces in space. Moreover, the rule of evolution in this dynamical system is the Navier-Stokes equations. In experiments, we do not have access to all the information in this dynamical system, such as velocity and pressure values at every location in space. However, we can put sensors to measure them at different points. As we increase the number of sensors in this channel, our observation becomes more complete. If we put an infinite number of sensors in this channel, then these measurements capture the true behaviour of the dynamical system. On the other hand, we have a limited number of sensors, and as we increase these sensors (e.g., pressure sensors), they might influence the flow. Lack of access to this information, describing the true behaviour of the dynamical system, was challenging for centuries. We have explained an arbitrary Hamiltonian system and its different states briefly. We will now explain how to solve this system using the Koopman formalism.

In 1931, Bernard Koopman introduced a linear transformation known as the Koopman operator [74]. He discovered that this operator is unitary for Hamiltonian dynamical systems. According to Koopman's theory, a linear operator exists for a set of observables that can convert a finite-dimensional non-linear system into an infinite-dimensional linear system in Hilbert space. One way to characterize the importance of dynamical patterns embedded in a linear dynamical system is to evaluate its Eigenvalues. This idea is the principle of Koopman Mode Decomposition (KMD) [120]. It describes how we can identify a dynamical system by only considering prominent patterns called "modes". In fluid mechanics, the intuition of Koopman's theory is to view the evolution of field variables, such as velocity and pressure, governed by the

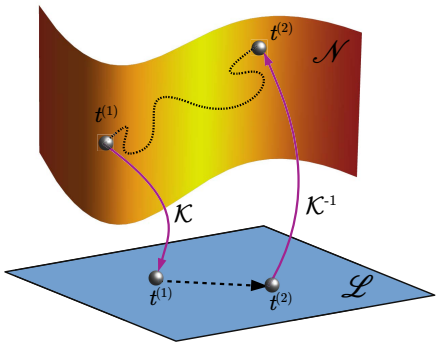


Figure 8: Schematic of Koopman's theory. Here $\mathscr{N}$ is a non-linear space with complicated structures, and $\mathscr{L}$ is a linear space, where there is an exact solution for the dynamical evolution of the generalized coordinates. $\mathcal{K}$ represents the Koopman operator, which works as a linear projection function.

Navier-Stokes equations, by only considering prominent flow features. The dynamical behaviour of these features is represented as a set of generalized coordinates in a Hilbert space.

Figure 8 shows a schematic of Koopman's theory. As solution at $t^{(1)} \in \mathbb{T}$ is embedded in the non-linear space $\mathscr{N}$, and the dashed line shows the trajectory of the solution from $t^{(1)}$ to $t^{(2)} \in \mathbb{T}$. However, because the dynamical system for this case evolves in $\mathscr{N}$, there is no exact solution at $t^{(2)}$. However, employing a Koopman operator $\mathcal{K}$, we can project this dynamical system onto a linear space $\mathscr{L}$, and convert it into a set of linear equations. Since there is an exact solution for this linear system, we can solve for it in generalized coordinates at $t^{(2)}$. Finally, we lift-back these generalized coordinates to $\mathscr{N}$ by $\mathcal{K}^{-1}$ to obtain the solution at $t^{(2)}$.

2.2 Ergodicity & Lyapunov Exponents

Most real-world engineering problems include chaotic dynamical systems. Ergodicity refers to the fact that long time-averaged solutions are ultimately independent of the initial condition. In fluid dynamics problems, the Navier-Stokes equations are identified as an ergodic system [46, 50, 55]. According to ergodic characteristics of a dynamical system, which is represented as Oseledet's Multiplicative Ergodic Theorem [106], there is a linear set of ODEs that satisfies the sensitivity solutions, such that

$$\frac{\partial}{\partial t}\mathfrak{C}_i(\boldsymbol{u}) = \frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}}\mathfrak{C}_i(\boldsymbol{u}) - \mathfrak{L}_i\mathfrak{C}_i(\boldsymbol{u}), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \tag{2}$$

where, $\mathfrak{L}_1 \geq \mathfrak{L}_2 \geq \mathfrak{L}_3 \geq \cdots \geq \mathfrak{L}_{n_u} \in \mathbb{R}$ are Lyapunov exponents (LEs). Furthermore, $\mathfrak{C}_1, \mathfrak{C}_2, ..., \mathfrak{C}_{n_u} \in \mathbb{R}^{n_u}$ represent covariant Lyapunov vectors. Consider the trajectory of a state in an arbitrary dynamical system that evolves in phase space, which is depicted in Figure 9. Here three different vectors correspond to the dimensions of the state in phase space. At instant $t^{(0)}$, the state is evenly augmented in all directions with

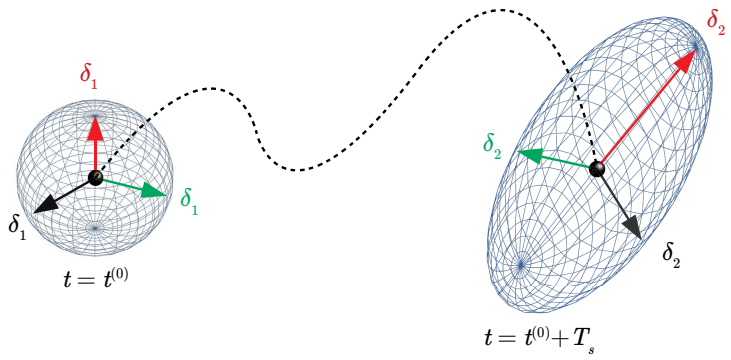


Figure 9: Explanation of covariant Lyapunov vectors and LEs.

a perturbation magnitude of δ_1. After the state advances in time until $t^{(0)} + T_s$, the augmentation of the corresponding trajectory grows, shrinks or remains unchanged in each direction. LEs are defined by the magnitudes of this augmentation in different directions. Additionally, these directions also change in phase space, which correspond to covariant Lyapunov vectors. Therefore, we can say that a small perturbation of δ_1 magnitude at $t = t^{(0)}$ in the dynamical system can be changed to $\delta_2 \approx \delta_1 \exp^{\mathfrak{L}_i(T_s - t^{(0)})}$, as $T_s \to \infty$. In general, a dynamical system can be characterized typically in four different categories as follows

- **Steady-state:** All LEs are negative, $\forall i \colon \mathfrak{L}_i < 0$.
- **Periodic:** One LE is equal to zero, and the rest are negative, $\mathfrak{L}_1 = 0, \quad \text{and} \quad \forall i \neq 1 \colon \mathfrak{L}_i < 0$.
- **Aperiodic:** Atleast two LEs are equal to zero, and other exponents are negative, $\forall j \in n_0 \colon \mathfrak{L}_j = 0, \quad \text{and} \quad \forall i \notin n_0 \colon \mathfrak{L}_i < 0$, where n_0 is list of indices at which the LEs are zero.
- **Chaotic:** Atleast one LE is positive, one is equal to zero, and other exponents are negative, $\forall k \in n_+ \colon \mathfrak{L}_k > 0, \quad \forall j \in n_0 \colon \mathfrak{L}_j = 0, \quad \text{and} \quad \forall i \notin n_0, n_+ : \mathfrak{L}_i < 0$, where n_+ is set of indices corresponding to the positive LEs.

To compute the asymptotic LEs of a large-scale dynamical system, the corresponding PDEs should be solved with DNS resolution [45], and generally speaking, obtaining the true values of these LEs is prohibitively expensive. However, there are alternative methods for estimating the LEs of a dynamical system, such as the Finite-Time Lyapunov Exponent (FTLE) [75, 97, 131]. In this study, we compute approximate values of LEs using a method introduced in [11], which is provided in Algorithm 1. Note that we only compute the first n_l leading LEs, where n_l is the total number of leading LEs for an ergodic system. In Algorithm 1, if a matrix is shown by $\mathbf{X} \in \mathbb{R}^{a\times b}$, then $\mathbf{X}[:, j] \in \mathbb{R}^a$ (where, $j \in \{0, 1, \ldots, b\}$) is the vector of j^{th} column in matrix $\mathbf{X}$. In this algorithm, covariant Lyapunov vectors are normalized using QR-factorization after each m_k iterations.

2.3 Governing Equations

2.3.1 Discrete Representation of Dynamical Systems

From the mathematical point of view, the dynamical system presented in Eq. (1) is usually non-linear for real-world engineering problems. The first step is to convert it into a set of Ordinary Differential Equations (ODEs) in a discrete space to solve this set of PDEs. These ODEs are obtained by spatial discretization techniques, such as the Finite Volume (FV), Discontinuous Galerkin (DG) or Flux Reconstruction (FR) [65] methods. In the second step, these ODEs are converted into a fully discrete linearized form. In this study, we use "linear multi-step" temporal schemes as time integrators. A linear k-step method applied to the semi-discrete form of Eq. (1) can be given by

$$\boldsymbol{r}^{(n)} := \sum_{j=0}^{k} \zeta_j \boldsymbol{u}^{(n-j)} + \Delta t^{(n)} \sum_{j=0}^{k} \beta_j \nabla . \boldsymbol{f}^{(n)} \Big(\boldsymbol{u}^{(n-j)}, t^{(n-j)}, \mathcal{S} \Big), \quad t \in \mathbb{T}, \tag{3}$$

where, $\boldsymbol{r} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$ denotes the residual of Eq. (1) in the fully-discrete form. Also, ζ and β are coefficients that define a specific temporal scheme. Moreover, $\Delta t \in \mathbb{R}_+$ is the time-step, and subscripts and superscripts in Eq. (3) denote the scheme's step and system's step, respectively. The consistency condition for the aforementioned schemes are $\zeta_0 \neq 0$ and $\sum_{j=0}^{k} \zeta_j = 0$. Also, these schemes are implicit if $\beta_0 \neq 0$. In Eq. (3), the state vector $\boldsymbol{u}^{(n)}$ is unknown, and can be solved by non-linear implicit solvers,

such as Gauss-Newton. In this method, $\boldsymbol{u}^{(n)}$ is found by minimizing the residual until we get $\boldsymbol{r}^{(n)} \to 0$. In this study, we use the Backward Diffrentiation Formula (BDF), where $\forall j \in [1, ..., k] : \beta_j = 0$. Therefore, Eq. (3) can be written as

$$\boldsymbol{r}^{(n)} := \sum_{j=0}^{k} \zeta_j \boldsymbol{u}^{(n-j)} + \Delta t^{(n)} \beta_0 \nabla . \boldsymbol{f}^{(n)}(\boldsymbol{u}^{(n)}, t^{(n)}, \mathcal{S}), \quad t \in \Delta\mathbb{T}, \tag{4}$$

where $\Delta\mathbb{T}$ is a discrete time domain, such that $t^{(n)} \in \Delta\mathbb{T}$. We use this branch of linear multi-step schemes because they are computationally cheaper than other branches in the same class of time integrator. Additionally, because of BDF's robustness and good numerical stability, it has been widely used in many scientific numerical simulations, particularly in fluid dynamics.

2.3.2 Navier-Stokes Equations

In this study, our objective is to consider the compressible Navier-Stokes equations as the dynamical system. This set of equations contain a high level of non-linearity which causes the dynamical system to be chaotic. In addition to chaoticity, the Navier-Stokes equations are usually represented as large-scale PDEs, making optimization and sensitivity analysis more challenging. Recall Eq. (1), according to the Navier-Stokes equation, one can cast the state and dynamical system in the following general form

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho e \end{bmatrix}, \tag{5}$$

and ρ is the density, ρu_i is a component of the momentum, and ρe is the total energy. The total flux $\boldsymbol{f} = \boldsymbol{f}_i - \boldsymbol{f}_v$ is the sum of the inviscid and viscous Navier-Stokes fluxes. The inviscid fluxes are

$$\boldsymbol{f}_{i,j}(\boldsymbol{u}, \mathcal{S}) = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + \delta_{ij} p \\ u_j \left(\rho e + p\right) \end{bmatrix}, \tag{6}$$

and the pressure is determined via the ideal gas law

$$p = (\gamma - 1)\rho \left(e - \frac{1}{2} u_k u_k \right), \tag{7}$$

where $\gamma = c_p/c_v$ is the ratio of specific heats, c_p is the specific heat at constant pressure, and c_v is the specific heat at constant volume. The viscous fluxes for the Navier-Stokes equations are

$$\boldsymbol{f}_{v,j}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \begin{bmatrix} 0 \\ \tau_{v,ij} \\ -q_{v,j} - u_i \tau_{v,ij} \end{bmatrix}, \tag{8}$$

where the heat flux is

$$q_{v,j} = -\frac{\mu_d}{Pr} \frac{\partial}{\partial x_j} \left(e + \frac{p}{\rho} - \frac{1}{2} u_k u_k \right), \tag{9}$$

Pr is the Prandtl number, and μ_d is the dynamic viscosity. The viscous stress tensor is then given by

$$\tau_{v,ij} = \mu_d \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{v,ij} \right), \tag{10}$$

where $\delta_{v,ij}$ is the Kronecker delta.

2.3.3 Spatial Discretization of the Navier-Stokes Equations

For spatial discretization of the Navier-Stokes equations we use Flux Reconstruction (FR) approach introduced by Hyunh [65]. FR utilizes an elementwise polynomial approximation of the solution, which allows it to obtain high-order accuracy on mixed-element unstructured grids. This also allows multiple levels of solution fidelity to be obtained on the same mesh, by simply increasing or decreasing the degree of the local polynomial approximation of the solution. The FR approach in one dimension will be described here, and its extension to multidimensional elements is available in Huynh's original paper [141].

We start by considering a computational domain, denoted by χ, decomposed into n_e elements χ_i such that

$$\chi = \bigcup_{i=1}^{n_e} \chi_i, \quad \text{and} \quad \bigcap_{i=1}^{n_e} \chi_i = 0. \tag{11}$$

Each element is mapped to a referential computational space $\chi^{'} = [-1, 1]$, within which a polynomial nodal basis representation of the solution is used. To transfer the solution from this computational domain to physical space a mapping function is used $\mathbb{P}_i : \chi \rightarrow \chi^{'}$. In each element the solution is represented at a set of nodal

basis points, referred to as solution points, such that $x_{i,p} = \{x_{i,0}, x_{i,1}, ..., x_{i,p_s}\}$ and $\xi_{i,p} = \{\xi_{i,0}, \xi_{i,1}, ..., \xi_{i,p_s}\}$, where p_s denotes the local polynomial degree that can be represented using the $p_s + 1$ solution points. Now, consider a general conservation law of the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}}{\partial x} = 0, \tag{12}$$

where $\boldsymbol{u}$ is the vector of conserved variables, $\boldsymbol{f} : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_u}$ is flux, and n_u is the number of conserved variables. The transformed solution in the reference element is defined as $\boldsymbol{u}_i^{'} = \mathbb{P}_i \boldsymbol{u}_i$. This transformed solution can be approximated using an elementwise polynomial representation of the solution

$$\boldsymbol{u}_i^{'\delta_m} = \sum_{j=0}^{p_s} \boldsymbol{u}_i^{'\delta_m}(\xi_j)\ell_j(\xi). \tag{13}$$

The flux can similarly be approximated by

$$\boldsymbol{f}_i^{'\delta_m} = \sum_{j=0}^{p_s} \boldsymbol{f}_i^{'\delta_m}(\xi_j)\ell_j(\xi), \tag{14}$$

where, $\ell(\xi)$ is one of $p_s + 1$ nodal basis functions defined as

$$\ell_j(\xi) = \prod_{k=0}^{p_s} \left(\frac{\xi - \xi_k}{\xi_j - \xi_k}\right)\left(1 - \delta_{m,jk}\right), \tag{15}$$

where, δ_m is the mesh spacing.

Since the solution is allowed to be discontinuous at the interface between elements, so too is the flux. Hence, via the FR approach, we enforce a common Riemann flux at this interface. For a given element, we denote these Riemann fluxes by $\boldsymbol{f}_l^{'}$ and $\boldsymbol{f}_r^{'}$ for the left and right-hand sides of the element, respectively. A corrected flux function is then introduced by

$$\boldsymbol{f}_c^{'}(\xi) = \left(\boldsymbol{f}_l^{'} - \boldsymbol{f}_l^{'\delta_m}\right)h_l(\xi) + \left(\boldsymbol{f}_r^{'} - \boldsymbol{f}_r^{'\delta_m}\right)h_r(\xi), \tag{16}$$

where, h_l and h_r are correction functions for the left and right-hand boundaries of the element [65]. The discrete form of the conservation equation can now be written as follows:

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\sum_{j=0}^{p_s} \boldsymbol{f}_{i,j}^{'\delta_m} \frac{d\ell_j(\xi)}{d\xi} - \left(\boldsymbol{f}_{i,l}^{'} - \boldsymbol{f}_{i,l}^{'\delta_m}\right)\frac{dh_l(\xi)}{d\xi} - \left(\boldsymbol{f}_{i,r}^{'} - \boldsymbol{f}_{i,r}^{'\delta_m}\right)\frac{dh_r(\xi)}{d\xi}. \tag{17}$$

In the current study we used the Rusanov Riemman solver for the common inviscid flux, and the Local Discontinuous Galerkin (LDG) scheme for the common viscous flux. Finally, this expression can be advanced in time using a suitable temporal integration scheme.

Chapter 3

Gradient-Free Optimization

3.1 Overview

Traditionally, aerodynamic optimization is performed using the adjoint combined with Reynolds Averaged Navier-Stokes (RANS) flow solutions [58, 83, 87, 125, 143]. However, the RANS approach has some notable limitations, specifically when turbulent transition and flow separation are observed. Therefore, the design optimization strategies applied with RANS are only suitable for limited classes of problems, those for which RANS provides accurate flow solutions. Outside of this regime high-fidelity simulations, such as Direct Numerical Simulations (DNS) or Large Eddy Simulations (LES), are required [128]. These unsteady scale-resolving methods capture some or all of the turbulent length and time scales in the flow [20, 139]. However, a reliable optimization strategy when using LES or DNS for the flow solution remains an open challenge. This is mainly due to finite time-averaging requirements, resulting in noisy objective functions that are not amenable to gradient-based optimization. Furthermore, conventional adjoint-based optimization has been shown to diverge for LES and DNS, due to the chaotic nature of the Navier-Stokes equations at high Reynolds numbers [17]. In contrast, gradient-free methods are more suitable for noisy objective functions, since they do not require any sensitivities.

Due to the deficiencies of gradient-based optimization techniques for LES and DNS, this study explores the gradient-free Mesh Adaptive Direct Search (MADS) algorithm for high-fidelity aerodynamic optimization. Recently, Bahrami et al. [7, 8] used this method for multi-objective optimization of hydraulic turbine runner blades.

They employed potential flow and RANS as a solver for the flow solver. Weiguang Yang [146] also applied MADS for a surgical design for the Fontan procedure and used RANS for solving the flow field within vessels. To the author's knowledge, there has not been any optimization done using high-fidelity flow simulations such as LES and DNS. In this study we explore the utility of MADS in this context for flow over low Reynolds number airfoils, specifically the SD7003. The open-source solver PyFR [141] is used due to its high-order accuracy and suitability for modern high-performance computing hardware architectures, meaning it can rapidly complete high-fidelity LES and DNS simulations within each design iteration. It follows that the objective of this work is to demonstrate that MADS is a suitable gradient-free shape optimization strategy when using LES as an underlying solution technique for turbulent flows.

3.2 Strategies Toward Gradient-Free Optimization

3.2.1 Solver Setup

In this chapter we use the open-source flow solver PyFR to provide numerical solutions to the governing equations [141]. In the current study we used the Rusanov Riemman solver for the common inviscid flux, and the Local Discontinuous Galerkin (LDG) scheme for the common viscous flux. Finally, this expression can be advanced in time using a suitable temporal integration scheme. In the current study we use the RK_{45} explicit Runge-Kutta method with adaptive time stepping [141].

Depending on the choice of correction functions, the FR approach can recover several existing schemes including the Discontinuous Galerkin (DG), Spectral Difference (SD), and Spectral Volume (SV) methods, depending on the choice of correction functions. Also, it is expected to recover high-order accuracy on mixed-element unstructured grids depending on the choice of solution points and polynomial basis used to represent the solution. Due to the level of data locality within each high-order element, the FR approach can exploit the performance of modern hardware accelerator hardware architectures, such as Graphical Processing Units (GPUs). Recent research has shown that this allows PyFR to obtain several orders of magnitude more accurate solutions using LES and DNS compared to current industry-standard CFD tools, at reduced computational cost. Hence, this makes PyFR an appealling framework for optimization using LES and DNS, since its high-order accuracy and computational

efficiency can significantly reduce the cost of each objective function evaluation and design iteration.

3.2.2 Mesh Adaptive Direct Search (MADS) Method

With the governing equations and flow solver defined, we now have a suitable framework for performing LES and extracting the objective functions of interest. For optimization we propose using the gradient-free MADS optimization algorithm, which is of particular interest since it can be used for non-smooth objective functions, $\mathcal{J}(\boldsymbol{u}, t, \mathcal{S})$, with respect to design parameters, $\mathcal{S}$. MADS's gradient-free nature can also be useful when several local stationary points exist in the design space or derivatives of the objective function are unavailable, either because it does not exist or cannot be accurately computed. MADS occupies a position somewhere between the Generalized Pattern Search (GPS) [4] and the Coope and Price frame-based methods [39]. The basic concept can be defined as two sequential steps in each design iteration. First, several trial points are identified in the design space, and infeasible trial points that do not satisfy the design constraints are discarded. This is referred to as the search step. The objective function is then evaluated at these trial points and compared with the incumbent value from the previous design iteration to determine if a new optimum has been found. Each of these trial points lies on a mesh constructed by a finite set of n_D directions, which is scaled by the mesh size parameter, Δ_m. Here, $\mathbf{D}$ is a matrix of directions, and it must be a positive spanning set [4, 6] and a nonnegative integer combination of the directions. Therefore, the mesh points are defined by

$$\mathcal{M}^k = \bigcup_{\mathcal{S}^k \in \mathscr{D}} \{\mathcal{S}^k + \Delta_m^k \mathbf{D}^k z : z \in \mathbb{N}^{n_D}\}. \tag{18}$$

The objective function, $\mathcal{J}$, is evaluated at the start of iteration k. The search-step is said to be empty when no trial points are considered. More discussion about search steps is given in [5]. In the next step, if an improvement in the objective function is not found, the mesh size parameter is reduced to increase the mesh resolution around the incumbent point. This is referred to as the poll step. This allows the evaluation of an objective function at trial points closer to the incumbent solution. The primary advantage of MADS over GPS is that the local exploration in $\mathscr{D}$ is not restricted to a finite number of directions. This is the primary drawback of the GPS algorithms,

and the main motivation in defining MADS was to overcome this restriction [5]. The trial points in the design space can be defined as

$$\mathcal{P}^k = \{\mathcal{S}^k + \Delta_p^k z : z \in \mathbf{D}^k\} \subset \mathcal{M}^k. \tag{19}$$

The mesh size parameter, Δ_m, may go to zero faster than the poll size parameter, Δ_p. In order to show that MADS can generate an asymptotically dense set of poll directions, the Householder matrix is used

$$\mathbf{H}^k = \mathcal{I} - 2\nu^k \nu^{k,\top} \subset \mathbb{R}^{n_s \times n_s}, \quad \text{and} \quad \mathbf{H}^k = \Big[h_1, h_2, ..., h_{n_s}\Big], \tag{20}$$

where n_s, $\mathcal{I}$ and ν are the number of design parameters, identity matrix and a normalized random vector, respectively. Additionally, $\nu^\top$ denotes the transpose of ν. To create a poll set, $\mathbf{H}^k$ is normalized to the range of the design values by

$$\begin{aligned}
&\mathbf{B}^k = \{b_1, b_2, ..., b_{n_s}\},\\
&b_j = round\left(\frac{\Delta_p}{\Delta_m}\frac{h_j}{max(\|h_j\|)}\right),\\
&\begin{cases}
\mathbf{D}^k = \mathbf{B}^k \cup \{z^{n_s+1} = -\sum_{i=1}^{n_s} z_i\},\\
\qquad \text{or}\\
\mathbf{D}^k = \mathbf{B}^k \cup (-\mathbf{B}).
\end{cases}
\end{aligned} \tag{21}$$

Additionally, the poll size parameters are defined as:

$$\Delta_m^{k+1} = \begin{cases}
\frac{\Delta_m^k}{4}, & \text{if } \mathcal{S}^k \text{ is a minimal frame center,}\\
4\Delta_m^k, & \text{if an improved mesh point is found, and if } \Delta_m^k \leq \frac{1}{4}\\
\Delta_m^k, & \text{otherwise,}
\end{cases} \tag{22}$$

$$\Delta_p^k = \begin{cases}
n_s\sqrt{\Delta_m^k} & \geq \Delta_m^k, \quad \text{if minimal positive basis,}\\
\sqrt{\Delta_m^k} & \geq \Delta_m^k, \quad \text{if maximal positive basis.}
\end{cases}$$

Figure 10 depicts the search and poll steps of MADS. At iteration k, the frame, shown by the black mesh, has the largest values of Δ_m and Δ_p. The incumbent point is $\mathcal{S}^k$, and three different random trial points ($\mathcal{P}_1^k$, $\mathcal{P}_2^k$ and $\mathcal{P}_3^k$) are selected from this mesh. In the search step, it is assumed for demonstration purposes that $\mathcal{P}_3^k$ is superior to other trial points, and hence, $\mathcal{P}_3^k$ would be $\mathcal{S}^{k+1}$ for the next iteration. In the poll

step the frame size decreases and the mesh resolution increases (here, the frame size has not been reduced for the blue mesh for clarity). When the mesh becomes finer, the number of possible trial points increases. In this case, let us assume that the blue trial points could not improve the objective function in the blue mesh frame, and for the next iteration, $k+2$, another poll step is required. As shown, the orange mesh becomes denser as the searching frame size reduces. In the end, when the poll size parameter reaches a user specified threshold for termination, the optimization stops. Note that the number of simulations required for each design cycle can be either $n_D + 1$ or $2n_D$ [4]. In this study, we use $2n_D$ simulations per design cycle.

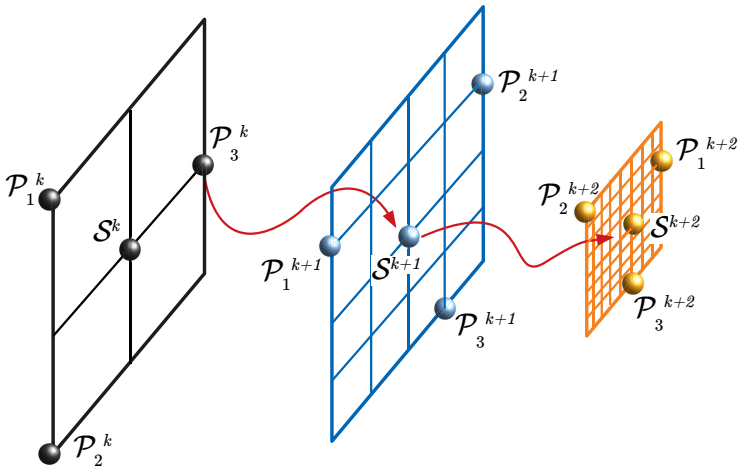


Figure 10: Schematic of the search and poll steps used by MADS.

3.2.3 Time-Averaging Sensitivity

Scale-resolving simulations are unsteady and can only be run for a finite amount of simulation time. As a result, time-averaged objective functions are inherently noisy [139]. It is essential to provide a metric to quantify this noise, to ensure that it remains within a reasonable level based on the required design precision. We assess convergence of the objective function by evaluating the time average of its derivative with respect to the time averaging period. When the time average of this derivative goes to zero, or is within a specified tolerance, it implies that the objective function

has reached a constant value. This can be approximated from

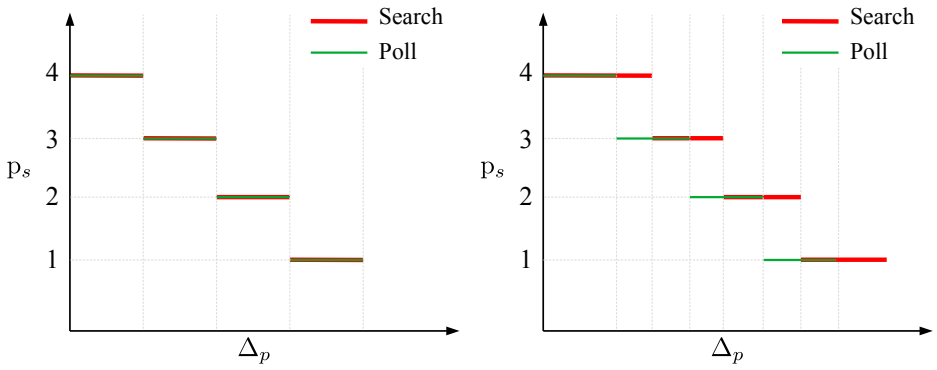
$$\begin{aligned}\frac{\overline{\Delta\overline{\mathcal{J}}}}{\Delta T} &= \frac{1}{T}\int_0^T \frac{\partial\overline{\mathcal{J}}}{\partial t}dt \\ &\approx \frac{1}{T}\int_0^T \left[\frac{\frac{1}{t+\Delta t}\int_0^{t+\Delta t}\mathcal{J}(\boldsymbol{u}(t,\mathcal{S}),\mathcal{S})dt - \frac{1}{t}\int_0^t \mathcal{J}(\boldsymbol{u}(t,\mathcal{S}),\mathcal{S})dt}{\Delta t}\right]dt\end{aligned} \tag{23}$$

where, T is the full time-averaging window and Δt is time interval. This allows us to assess the sensitivity of the time-averaged objective function with respect to the averaging period.

3.2.4 Dynamic Polynomial Approximation (DPA)

Aerodynamic shape optimization is usually initialized using a large design space to include a wide range of different possible designs. However, in practice large regions of this design space yield objective functions that are significantly worse than that of the initial design. To accelerate convergence towards the optimal design, we propose that lower-fidelity simulations can be used when MADS is performing a broad search in the design space. This allows MADS to quickly discard regions of the design space where the optimum is unlikely to be found. Then, as MADS converges towards a smaller region of the design space, we increase the fidelity of the simulations to provide more accuracy in the objective function. In the current study we achieve this by adjusting the solution polynomial degree used by PyFR, depending on the size of the search window used by MADS. We refer to this approach as Dynamic Polynomial Approximation (DPA).

In the current study we consider two different strategies for performing DPA. In the first approach, increasing and decreasing the solution polynomial degree is performed using the same threshold values, referred to here as standard DPA. In the second approach the thresholds for increasing or decreasing the solution polynomial degree are offset, referred to as binary DPA, to minimize the number of changes in degree. Figure 11 shows the difference between the standard and binary DPA approaches for a generic objective function. Figure 11a shows that the polynomial order for the standard approach will change at the same threshold level based on Δ_p, the poll size used by MADS. The red and green lines show the search and poll steps in MADS, respectively. As Δ_p decreases, the polynomial order increases, while increasing Δ_p leads to a decrease in polynomial order. Figure 11b also illustrates



(a) standard DPA. (b) binary DPA.

Figure 11: Different strategies for DPA.

binary DPA in which the thresholds for increasing or decreasing the polynomial order are offset. The utility of these two modes of DPA will be assessed in the following sections.

3.2.5 Multi-Level Parallelism

At the start of each design iteration MADS selects a number of candidates in the design space, as shown in Figure 10. Each of these requires geometry specification, mesh generation, running a simulation, and then post-processing the results to return the objective function. Importantly, each of these simulations can be performed in an embarrassingly parallel manner. This means that, provided sufficient computational resources, any number of design parameters and resulting design candidates can be evaluated in constant time. The MADS implementation starts by spawning a parent CPU *task* that is responsible for one of the design candidates. Each of these CPU *tasks* then exploits a second level of parallelism, by partitioning the mesh and launching its respective LES simulation on a group of GPUs. This allows the time for each design iteration to be reduced by strong-scaling each simulation. This strategy was found to significantly reduced the time required to complete a design cycle, and was effective at utilizing large clusters of GPUs.

3.3 Optimization Results

In this section, three different optimization cases are use to demonstrate the utility of MADS for optimizing chaotic dynamical systems. These include, the classical Lorenz system, a 2D airfoil at $Re = 10^4$, and a 3D turbulent airfoil at $Re = 6 \times 10^4$ to demonstrate gradient-free optimization using LES.

3.3.1 The Lorenz System

The Lorenz system is one of the most well-known examples of chaotic dynamical systems. It represents the relation between thermodynamic properties of a 2D fluid layer that is evenly heated from below and cooled from above. The set of equations describing its evolution are

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \sigma_L(\mathbf{y} - \mathbf{x}) \\ \frac{d\mathbf{y}}{dt} &= \mathbf{x}(\rho_L - \mathbf{z}) - \mathbf{y} \\ \frac{d\mathbf{z}}{dt} &= \mathbf{x}\mathbf{y} - \beta_L \mathbf{z} \end{aligned} \tag{24}$$

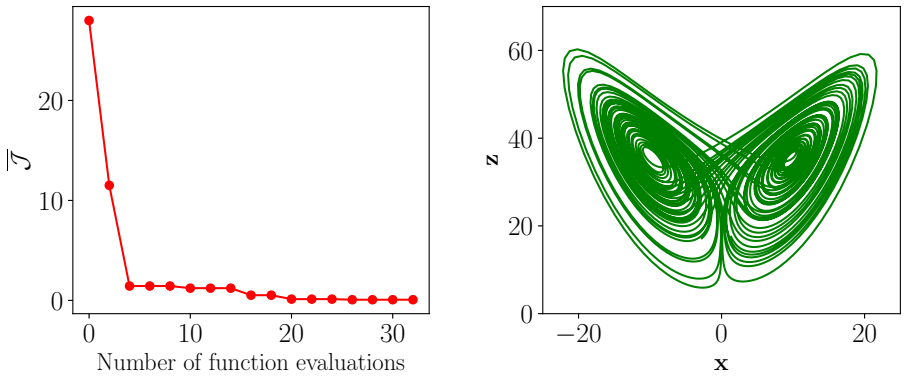
where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ describe the system state variables, corresponding to rate of convection, horizontal temperature variation, and vertical temperature variation, respectively. Furthermore, σ_L, ρ_L, and β_L are referred to as the Prandtl number, Rayleigh number, and system constant, respectively. In this study σ_L and β_L are taken to be 10 and 8/3, respectively.

3.3.2 Application of MADS to the Lorenz System

The objective function is taken to be a shifted time-average of $\mathbf{z}$

$$\overline{\mathcal{J}} = \frac{1}{T}\int_0^T (\mathbf{z} - 32)dt, \tag{25}$$

and ρ_L is taken to be the design parameter. Hence, the objective of this optimization is to drive the Lorenz system towards a solution where $\mathbf{z}$ has a mean value of 32 by changing the value of ρ_L. The initial design starts with $\rho_L = 5$, and a stopping precision of 10^{-4} is used. Each solution is run for 200 time units, and then time-averaged over 100 time units, which was found to give sufficient convergence of the



(a) Objective function progress (b) Trajectories at $\rho_L = 36.25$

Figure 12: Optimization of the Lorenz system using MADS.

objective function. MADS completed a total of 32 function evaluations (16 design iterations) and ultimately converged to the minimum value of $\overline{\mathcal{J}}$ at $\rho_L = 36.25$. Figure 12a shows the progression of the objective function versus design iterations. From this it is clear that MADS is able to rapidly converge to the region of optimal designs, and the remainder of the optimization process then reduces the objective function to within the specified tolerance. Importantly, despite the underlying dynamical system being chaotic, and the objective function containing noise, MADS is still able to successfully optimize it. Figure 12b shows the trajectory of the Lorenz attractor for the optimal design with $\rho_L = 36.25$, plotted in phase space. This shows the trajectory and chaotic behaviour of the Lorenz system, in the form of a strange attractor, at the optimum point. From this simple demonstration case it is apparent that MADS is suitable for optimization of chaotic systems with finite time-averaged objective functions.

3.3.3 Low Reynolds Number Flow: Re=10,000

Problem Description

To demonstrate the utility of MADS for the Navier-Stokes equations, we consider unsteady low Reynolds number flow over an SD7003 airfoil. The two-dimensional computational domain extends to at least $12c$ away from the airfoil surface, and is discretized using a regular C-type mesh with 2,836 quadratically-curved quadrilateral elements, as shown in Figure 13. For the ranges of angles of attack and chosen

Reynolds number the flow is expected to remain laminar over the surface of the airfoil [134]. The mesh is refined towards the surface of the airfoil to resolve the viscous boundary layer region, and a stretching ratio of 1.03 is applied outwards from the airfoil surface. Figure 13 shows the computational domain for the baseline

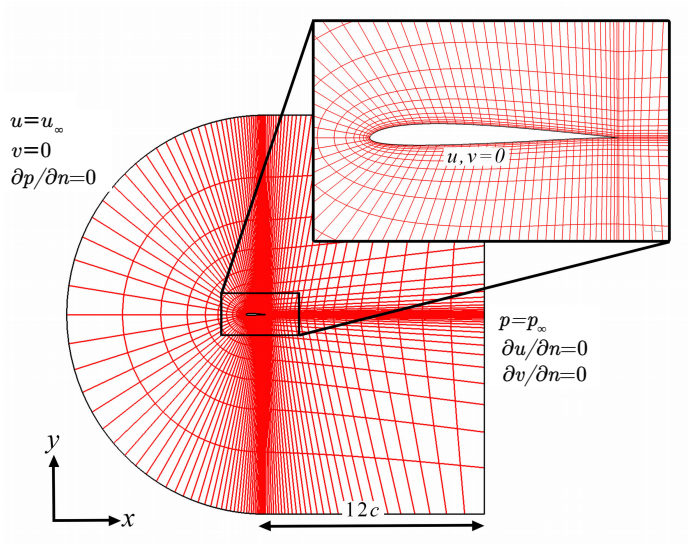


Figure 13: Computational domain and mesh for the SD7003 airfoil at $Re = 10^4$.

SD7003 with $\alpha = 0°$ and $\eta_c = 0$. The upstream and lateral edges of the domain are specified using a standard velocity inlet boundary condition with $u = u_\infty$ and $v = 0$. Additionally, the downstream boundary is specified using the ambient pressure $p = p_\infty$, and a no-slip adiabatic boundary condition is specified on the surface of the airfoil.

Resolution Requirements

Prior to optimizing the airfoil design, a sufficient level of resolution is assessed through a polynomial convergence study at an angle of attack $\alpha = 4°$. Quantitative results in Table 10 show the influence of polynomial degree on the time-average lift and drag coefficients, alongside reference results from Uranga et al. [134]. The results using $p_s = 3$ are within 1% of those using $p_s = 4$, suggesting that $p_s = 3$ provides sufficient resolution at this Reynolds number. Therefore, $p_s = 3$ is used as the highest solution polynomial degree during the optimization procedure. Figure 14 shows vorticity contours for each polynomial degree, and demonstrates qualitatively the influence of increasing the solution polynomial degree on the results. The solution using $p_s = 1$

p_s	Order of accuracy	$\overline{C_L}$	$\overline{C_D}$	Error($\overline{C_L}$)	Error($\overline{C_D}$)
1	2^{nd}	0.3172	0.04628	16.34%	6.33%
2	3^{rd}	0.3769	0.04856	0.61%	1.71%
3	4^{th}	0.3788	0.04927	0.11%	0.27%
4	5^{th}	0.3792	0.04941	-	-
Uranga et al. [134]	4^{th}	0.3743	0.04967	-	-

Table 1: Comparison of the results from different orders of accuracy with the results from Uranga et al. [134]

Figure 14: Contours of vorticity magnitude for the SD7003 verification case at $Re = 10^4$ with $p_s = 1$, 2, 3 and 4 at $\alpha = 4°$.

is overly-dissipative, leading to premature breakdown of the unsteady vortices in the wake of airfoil. However, increasing the solution polynomial degree beyond $p_s = 3$ does not result in a discernable change in the resulting flow field.

Objective Function Sensitivity

In order to obtain accurate approximations of the aerodynamic forces on the airfoil, a sufficient time-averaging period must be used. In Figure 15, the sensitivity of the time-averaged lift and drag coefficients, $\overline{C}_L$ and $\overline{C}_D$, respectively, are shown where their instantaneous values are obtained from

$$C_L = \frac{F_y}{0.5\rho_\infty c u_\infty^2}, \quad C_D = \frac{F_x}{0.5\rho_\infty c u_\infty^2}, \tag{26}$$

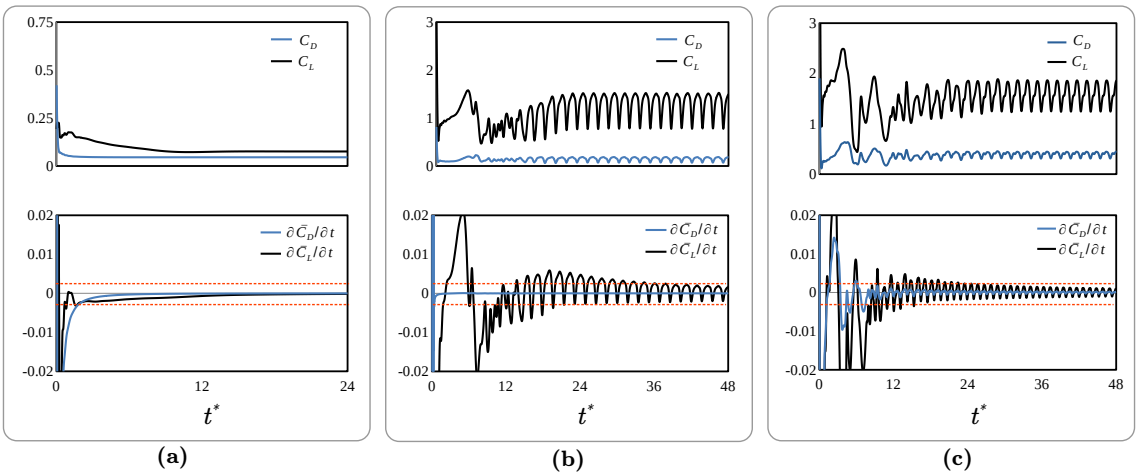


Figure 15: Instantaneous aerodynamic loads (C_L and C_D) and their sensitivity with respect to the evaluation time ($\frac{\partial \overline{C_L}}{\partial t}$ and $\frac{\partial \overline{C_D}}{\partial t}$) for the SD7003 case at $Re = 10^4$ with: (a) $\alpha = 0°$, (b) $\alpha = 7°$, and (c) $\alpha = 14°$.

where c is the airfoil chord length, and ρ_∞ is the freestream density. The plotted values $\partial \overline{C_D}/\partial t$ and $\partial \overline{C_L}/\partial t$ show the sensitivity of the time-averaged lift and drag coefficients with respect to the time-averaging period $[0, t]$. Three different angles of attack are considered for assessing this sensitivity, including $\alpha = 0°$, $\alpha = 7°$ and $\alpha = 14°$. We note that, depending on the angle of attack of the airfoil, a different time averaging period may be required to obtain sufficient convergence.

To automate the selection of the time averaging period, we start with an initial time averaging period of $24t^*$, where $t^* = tu_\infty/c$. If the sensitivity of the objective function with respect to the time-averaging window is above a specified threshold value of $\sim 2.5 \times 10^{-2}$, another $24t^*$ time-averaging window will be added. This procedure is repeated until the sensitivity of the objective function drops below the aforementioned convergence criteria. The threshold range is shown by dashed-lines. As shown in Figure 15a, the values of the instantaneous C_L and C_D for $\alpha = 0°$ become steady before $t^* = 24$. Figure 15b shows that the sensitivity of the objective function is not within the specified threshold during the first averaging period, and hence, another averaging period is added. Figure 15c also shows the same observations as Figure 15b, but at a higher angle of attack. This approach allows us to automatically select a sufficient time-averaging period, and extend the length of a simulation should additional time-averaging be required.

The Optimization Process

In this section we use MADS to optimize the shape of the SD7003 airfoil using constant solution polynomials of degree $p_s = 1$, 2, and 3. The design parameters are the angle of attack α, and the maximum camber η_c. A perturbation to the camber line is defined as

$$\begin{aligned} y_c &= \frac{\eta_c}{x_m^2}(2x_m x_c - x_x^2), \quad 0 \leq x_c < x_m, \\ y_c &= \frac{\eta_c}{(1-x_m)^2}(1 - 2x_m + 2x_m x_c - x_{c^2}), \quad x_m \leq x_c \leq 1, \end{aligned} \tag{27}$$

where x_c and y_c represents the camber line coordinates in x and y directions. Additionally, x_m shows the location of maximum camber, which is set to $x_m = 0.3$. We use MADS to optimize the shape of the SD7003 airfoil using constant solution polynomials of degree $p_s = 1$, 2, and 3. This will then be followed in the proceeding Section 3.3.3, where we demonstrate the utility of adapting the polynomial degree using DPA. Figure 16 shows the progress of the time-averaged objective function with respect to

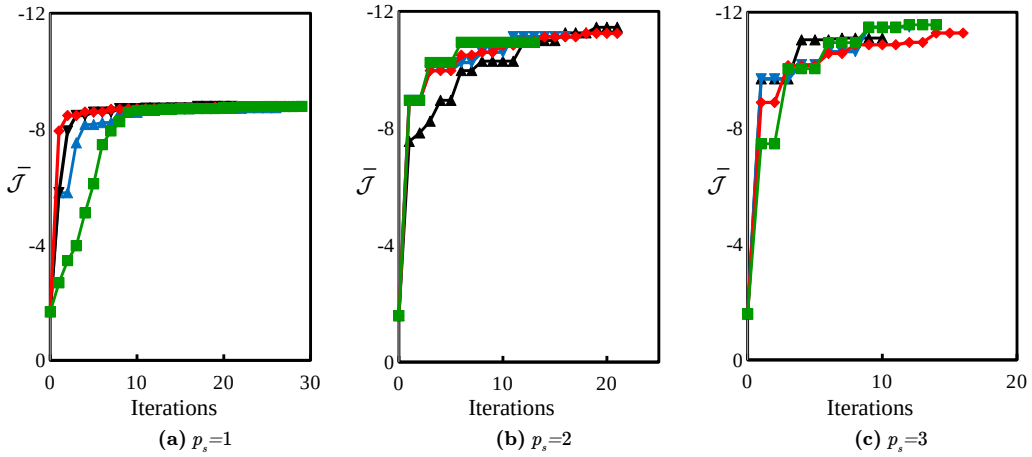
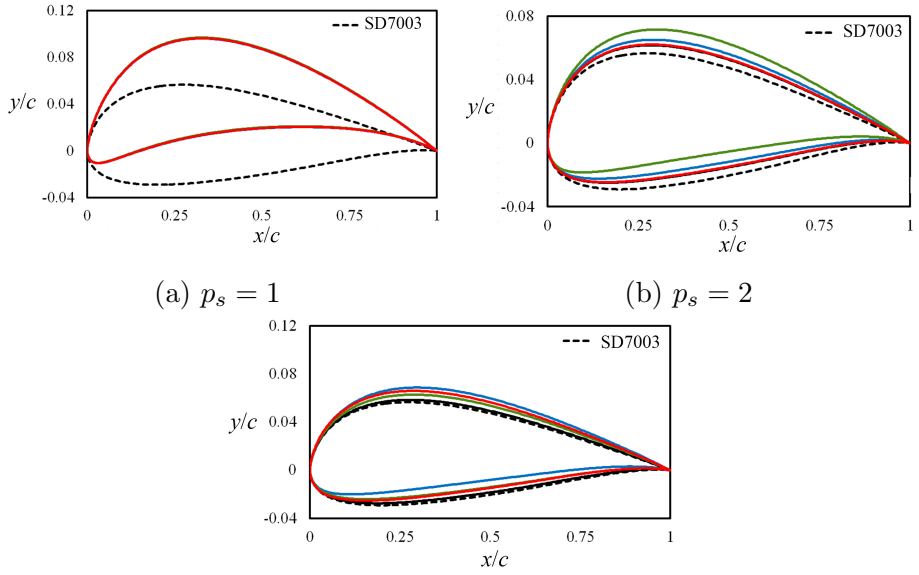


Figure 16: Progress of the objective function using MADS for optimization of the SD7003 at $Re = 10^4$.

the design iteration, which is taken to be

$$\overline{\mathcal{J}} = -\overline{C_L}/\overline{C_D}. \tag{28}$$

Here a negative sign is added since MADS is designed to solve general minimization problems.



(a) $p_s = 1$ (b) $p_s = 2$

(c) $p_s = 3$

Figure 17: Optimal airfoil shapes for the SD7003 at $Re = 10^4$. Colours correspond to each optimization represented in Figure 16.

Sice MADS uses random searches, we repeat each optimization procedure with each polynomial degree a total of four times. Figure 16a shows that each optimization iteration using $p_s = 1$ converges to approximately the same optimal value of the objective function, noting that they follow a slightly different path due to the random searches performed by MADS. Figures 16b and 16c show results from the same optimization procedure using $p_s = 2$ and $p_s = 3$. These show that as the solution resolution is increased, there is greater variability in the optimal design found by MADS. This behaviour is expected, and can be linked to the higher-resolution simulations capturing more of the chaotic non-linear structures in the flow [45], resulting in a less smooth design space when compared to the $p_s = 1$ simulations. However, despite this variability, all optimization iterations arrive at similar optimal values for the objective function. The shapes for the initial and final designes are provided in Figure 17 for each of the four optimization iterations using each polynomial degree. For $p_s = 1$, all four optimization iterations converge to nearly identical final designs. In this case, α and η_c are 3.76 ± 0.17 and 0.02154 ± 0.002, respectively. However, as shown in Figure 17b and 17c, the higher fidelity optimizations using $p_s = 2$ and $p_s = 3$ find several distinct designs that yield nearly the same value of the objective function. For example, one final design for $p_s = 2$ has $\alpha = 6.43 \pm 0.39$ and $\eta_c = 0.00525 \pm 0.0027$,

and for $p_s = 3$ an optimal design has $\alpha = 5.81 \pm 0.36$ and $\eta_c = 0.00315 \pm 0.0028$. Finally, Figure 18 shows contours of vorticity magnitude around the airfoil for the

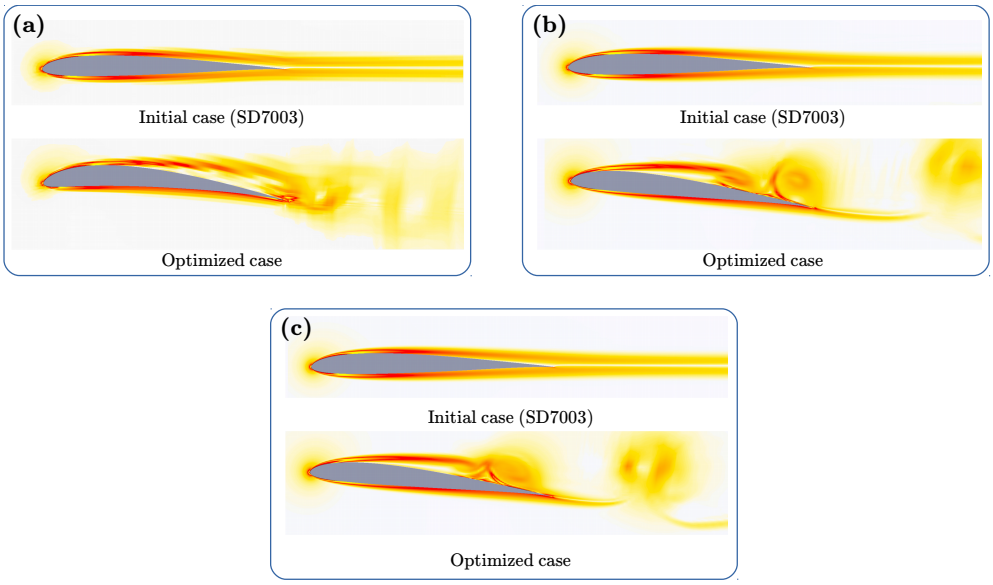


Figure 18: Contours of vorticity magnitude for the initial SD7003 at $Re = 10^4$ and $\alpha = 0°$, and the optimized airfoils with: (a) $p_s = 1$, (b) $p_s = 2$, and (c) $p_s = 3$.

initial and optimal designs for each polynomial degree. We note the under-resolution of the $p_s = 1$ simulation, whereas the final designs using $p_s = 2$ and $p_s = 3$ are similar in both shape and the behaviour of the boundary layer and wake regions of the flow.

Dynamic Polynomial Approximation (DPA)

In this section we will demonstrate the utility of DPA in reducing the cost of a complete optimization iteration. We again consider four independent optimization procedures. Table 2 is provided to show changes in the polynomial order with respect to different Δ_p for both standard and binary DPA. According to this table, in standard DPA, the polynomial order has the same threshold level based on Δ_p, while binary DPA uses a different threshold level. Figure 19 illustrates optimization results with standard DPA. In this case, the polynomial order can change from $p_s = 1$ up to $p_s = 4$, and vice-versa. Figure 19a shows that the progress of the objective function is different for all four cases. The maximum objective function achieved is 10.55 ± 0.96. Interestingly, the objective function occasionally reduces as the number of optimization iterations increases. This reduction in the objective function can

Polynomial order	Δ_p (standard DPA)	Δ_p (binary DPA)
$p_s : 1 \rightarrow 2$	0.5	0.5
$p_s : 2 \rightarrow 3$	0.25	0.25
$p_s : 3 \rightarrow 4$	0.0625	0.0625
$p_s : 2 \rightarrow 1$	0.5	2.0
$p_s : 3 \rightarrow 2$	0.25	1.0
$p_s : 4 \rightarrow 3$	0.0625	0.25

Table 2: Poll size parameters for standard and binary DPA.

occur when there is a change in p_s, which means using DPA with MADS may not converge monotonically. Figure 19b also shows four different paths taken through the design space with optimal points denoted by stars. These results demonstrate that the standard DPA approach can lead to a continuous expansion/contraction of Δ_p in MADS, which can lead to a large number of design iterations. Figure 20

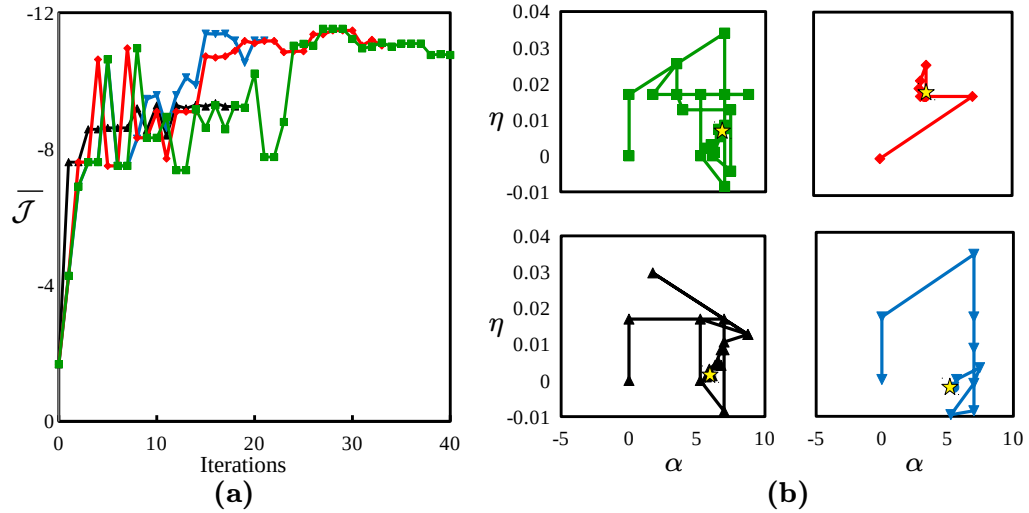


Figure 19: Optimization of the SD7003 at $Re = 10^4$ with standard DPA: (a) progress of the objective function and (b) trajectories in the design space.

shows the optimization results for four similar optimization cases but using the binary DPA approach. Figure 20a shows different trajectories of the optimization through the design space. This demonstrates that the binary DPA approach converges in fewer iterations than the standard DPA approach, reaching a maximum objective function of 10.98 ± 0.08. Additionally, fluctuations in the objective function are significantly reduced when compared to standard DPA. Figure 20b shows that binary

DPA rapidly converges to the optimal design space. In conclusion, standard DPA can result in unwanted oscillations between solution polynomial degrees, and fluctuations in the objective function. In contrast, binary DPA, which offsets the thresholds for increasing or decreasing the solution polynomial degree, is an appealling approach for reducing computational cost and rapidly converging towards the optimal design space.

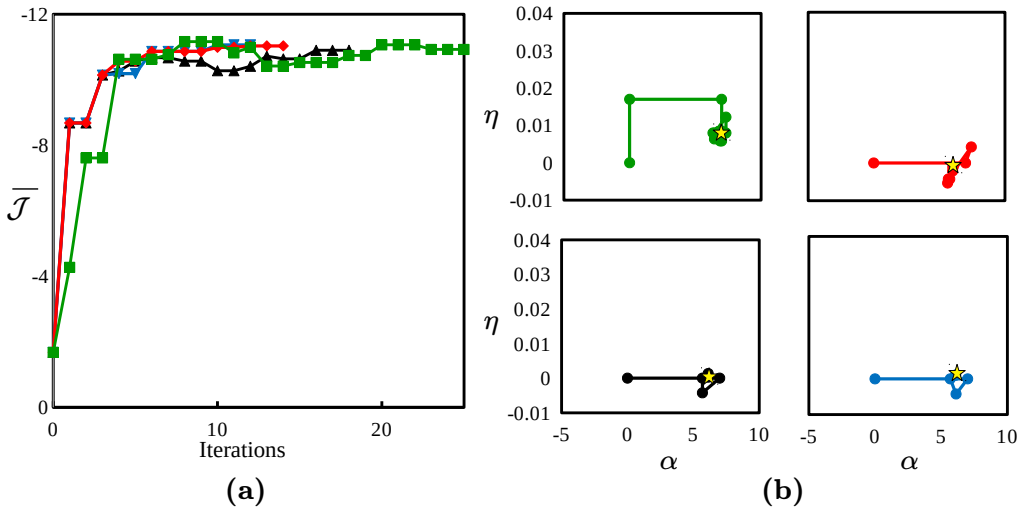


Figure 20: Optimization of the SD7003 at $Re = 10^4$ with binary DPA: (a) progress of the objective function and (b) trajectories in the design space.

Figure 21 represents the contours of the vorticity magnitude around the airfoil for one of the optimal designs obtained using binary DPA. Figure 21a shows the initial design with $p_s = 1$. After several MADS iterations, the polynomial order is automatically increased to $p_s = 2$, as shown in Figure 21b, at which point the design has $\eta_c = 0$ and $\alpha = 7°$. Finally, the polynomial degree is automatically increased to $p_s = 3$, when the design has $\alpha = 5.68°$ and $\eta_c = -0.0043$, as shown in Figure 21c. Results for the final optimal design are then shown in Figure 21d, which has $\alpha = 6.04°$ and $\eta_c = -0.0011$. Table 3 represents the average computational cost for a range of optimizations with different polynomial orders. As seen, standard DPA interferes with the performance of MADS and therefore, it is not recommended. Table 4 compares the computational cost of optimization with standard DPA and binary DPA. The average computational cost per iteration grows as the polynomial order increases. Interestingly, binary DPA using up to $p_s = 4$ has a lower computational cost than using constant $p_s = 3$. Hence, binary DPA is a suitable approach for

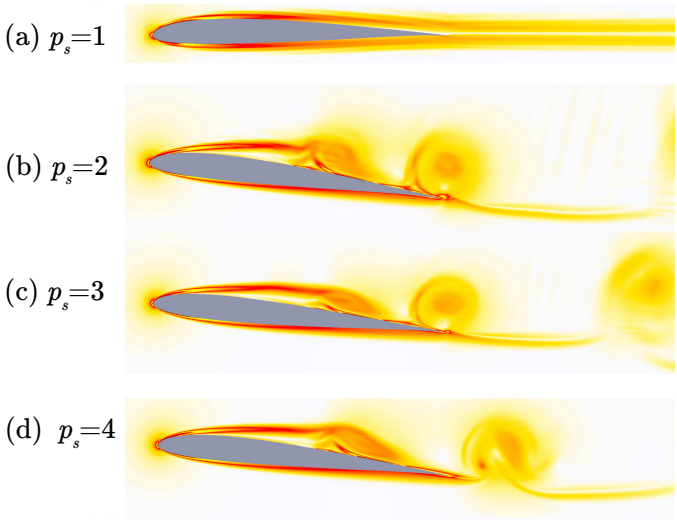


Figure 21: Contours of vorticity magnitude for the SD7003 at $Re = 10^4$ using binary DPA.

decreasing the total computational cost of optimization using MADS.

Polynomial order	Optimization iterations	cost per iteration $(hr/iter)$
$p_s = 1$	25	0.33
$p_s = 2$	17	0.68
$p_s = 3$	13	0.82
standard DPA, $p_s = [1, 4]$	28	0.93
binary DPA, $p_s = [1, 4]$	17	0.76

Table 3: Average computational cost of optimization using different polynomial degrees for the SD7003 airfoil at $Re = 10^4$.

3.3.4 Turbulent Flow: $Re = 6 \times 10^4$

Mesh Generation

As a final test case, we consider shape optimization of a turbulent SD7003 airfoil using LES. An unstructured 2D mesh is developed and then extruded $0.2c$ in the spanwise direction using 12 layers. A structured boundary layer region is used, with quadratically curved elements on the surface of the airfoil. The outer boundaries of the domain extend to $12c$ away from the airfoil surface, and the boundary conditions are the same as those used in the low Reynolds airfoil case. Figure 22 shows the

Polynomial order	Simulation cost (hr)	Number of simulations (standard DPA)	Number of simulations (binary DPA)
$p_s = 1$	0.33	12 ± 4	4 ± 1
$p_s = 2$	0.51	8 ± 2	2 ± 1
$p_s = 3$	1.02	11 ± 4	8 ± 4
$p_s = 4$	1.67	4 ± 1	5 ± 2
	Total cost (hr)	$17.85 \sim 34.03$	$10.59 \sim 27.11$

Table 4: Comparison of computational cost for optimization using standard and binary DPA for the SD7003 at $Re = 10^4$.

topology of the mesh. A refined region is also used on the upper surface of the airfoil and in the wake near the trailing edge to accurately capture the turbulent flow in these regions.

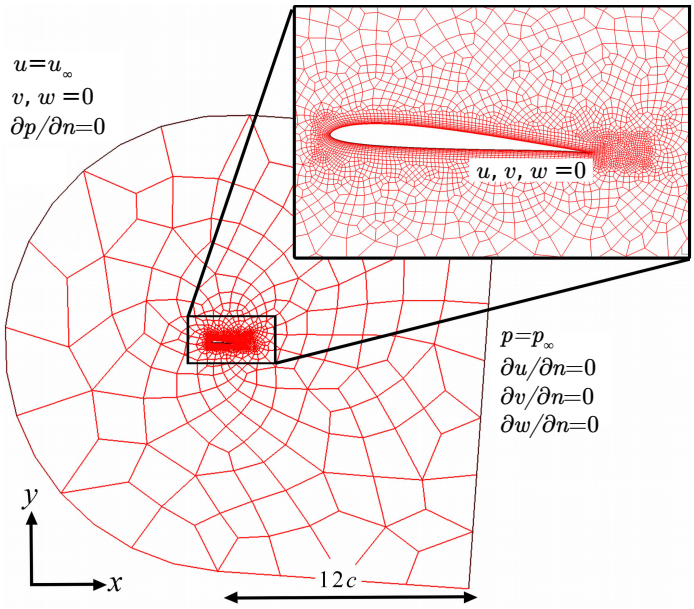


Figure 22: Computational domain and mesh for the SD7003 airfoil at $Re = 6 \times 10^4$.

Verification

To verify the mesh resolution, we consider the initial SD7003 design at an angle of attack $\alpha = 8°$. Mean aerodynamic loads are compared with the numerical results of Vermeire et al. [136] and Beck et al. [10] in Table 5. The values of $\overline{C_L}$ and $\overline{C_D}$ remain relatively unchanged beyond $p_s = 4$, and the errors corresponding to $\overline{C_L}$ and $\overline{C_D}$ are 0.1% and 1.59%, respectively, when compared to the values obtained using $p_s = 5$.

Additionally, the lift and drag coefficients at $p_s = 4$ are in good agreement with the reference results of Vermeire et al. [136] and Beck et al. [10]. Hence, $p_s = 4$ is chosen as the highest polynomial degree to use during the optimization procedure.

p_s	Order of accuracy	$\overline{C_L}$	$\overline{C_D}$	error($\overline{C_L}$)	Error($\overline{C_D}$)
1	2^{nd}	0.984	0.028	4.23%	44.22%
2	3^{rd}	0.976	0.039	3.38%	22.31%
3	4^{th}	0.954	0.043	1.05%	14.34%
4	5^{th}	0.945	0.051	0.10%	1.59%
5	6^{th}	0.944	0.0502	-	-
Vermeire et al. [136]	5^{th}	0.941	0.049	-	-
Beck et al. [10]	8^{th}	0.932	0.050	-	-

Table 5: Verification data for the SD7003 at $Re = 6 \times 10^4$ and $\alpha = 8°$.

Optimization Process

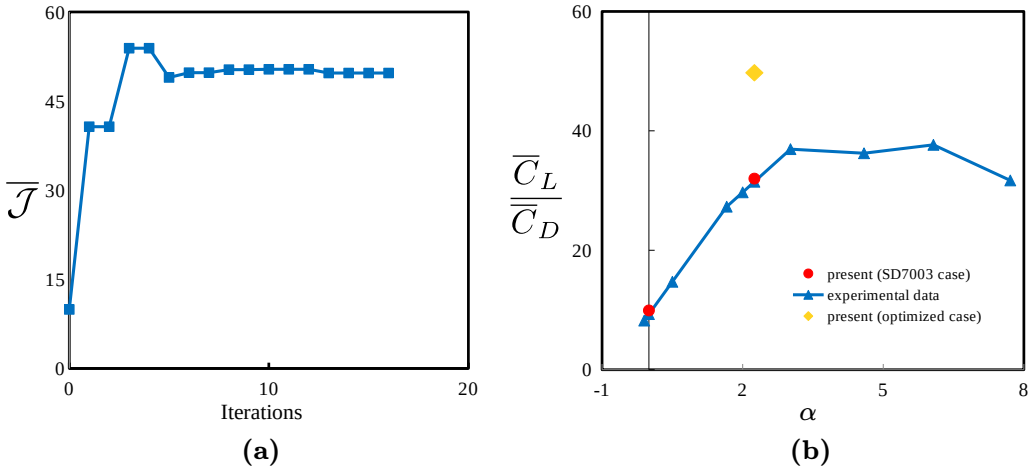
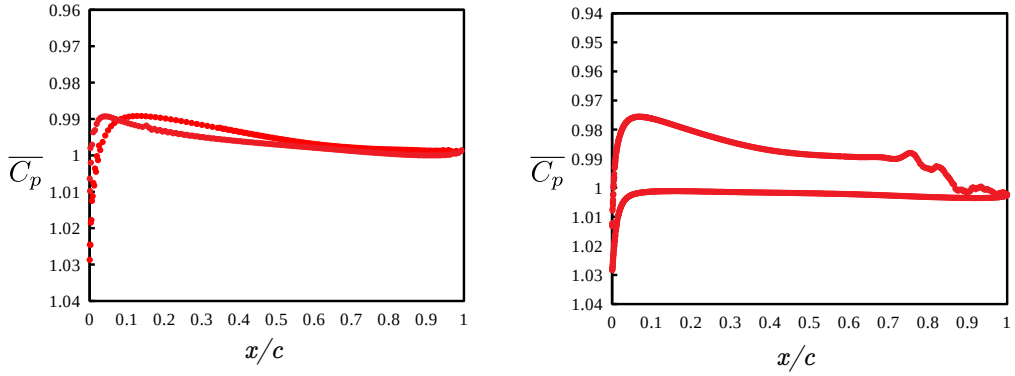


Figure 23: Optimization results for the SD7003 at $Re = 6 \times 10^4$ with binary DPA, including the objective function versus optimization iterations (a), and trial points in the design space (b).

Based on the verification results, we now consider a single optimization procedure using binary DPA for the turbulent SD7003 airfoil. Figure 23a shows the variation of the objective function versus the number of design iterations. The angle of attack and the maximum camber of the initial design is set to $\alpha = 0°$ and $\eta_c = 0$, respectively. Following the DPA approach, the initial design is launched with $p_s = 1$. The objective function quickly increases to 53 with $p_s = 2$, and slightly decreases when the solution

polynomial degree is increased to $p_s = 3$. Finally, MADS converges to the optimal design using $p_s = 4$ with an objective function of 49.7, which has $\alpha = 2.25°$ and $\eta_c = 0.00348$. Figure 23b compares the aerodynamic characteristics of the optimized airfoil with those of the initial SD7003. Experimental data [124] for the SD7003 shows that the maximum value of $\overline{C_L}/\overline{C_D}$ is around 37.6 at $\alpha = 6.07°$, as shown in Figure 23b. The optimum design obtained using MADS and binary DPA achieves a 32% increase in the objective function to $\overline{C_L}/\overline{C_D} = 49.7$. An additional simulation of the initial SD7003 is performed at the same $\alpha = 6.07°$, showing excellent agreement with the reference experimental data and that the optimal airfoil obtains a significant increase in aerodynamic performance. Figure 24 shows the time-averaged pressure coefficient,



(a) SD7003 **(b)** Optimized case

Figure 24: Pressure coefficient distribution for the initial and final SD7003 designs at $Re = 6 \times 10^4$ and $\alpha = 6.07°$.

$\overline{C_p}$, for both the initial and optimal designs. This shows that the optimal design has a significantly large pressure difference between the upper and lower surfaces, resulting in greater lift. Figure 25 shows isosurfaces of Q-criterion for the optimized airfoil. This shows that the flow transitions to turbulence in the aft portion of the airfoil. In conclusion, these results demonstrate that MADS combined with binary DPA is a suitable strategy for aerodynamic shape optimization using LES.

3.4 Remarks

In this study, we have demonstrated the utility of MADS for shape optimization using time-averaged objective functions obtained via LES for turbulent flows. The general approach was initially validated using optimization of the chaotic Lorenz system. We

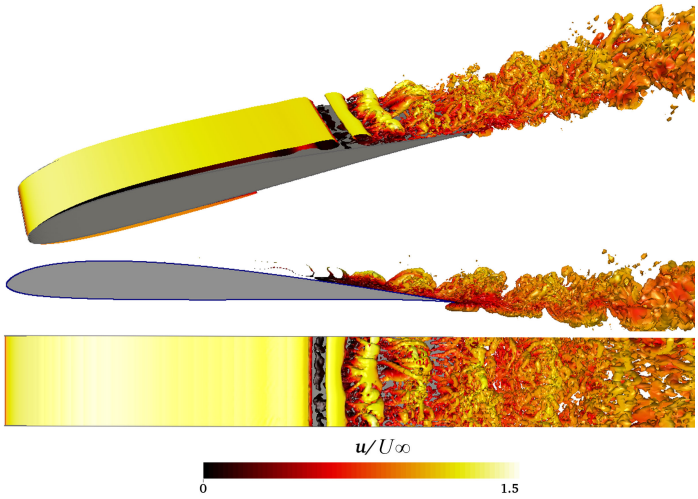


Figure 25: Isosurfaces of Q-criterion for the optimized SD7003 airfoil at $Re = 6 \times 10^4$.

then applied the approach to two aerodynamics applications using the SD7003 airfoil as an initial design. The first low Reynolds case was used to investigate DPA, where the solution polynomial degree was dynamically adapted based on the current poll size used by MADS. When the poll size is large, corresponding to a broad search in the design space, a lower solution polynomial degree is used to reduce computation cost. Conversely, when the poll size is small, corresponding to a narrow search near the optimal design, a high solution polynomial degree was used to increase solution accuracy. Of the two DPA modes investigated, the binary mode was found to be more robust as it did not introduce unwanted oscillations in the objective function.

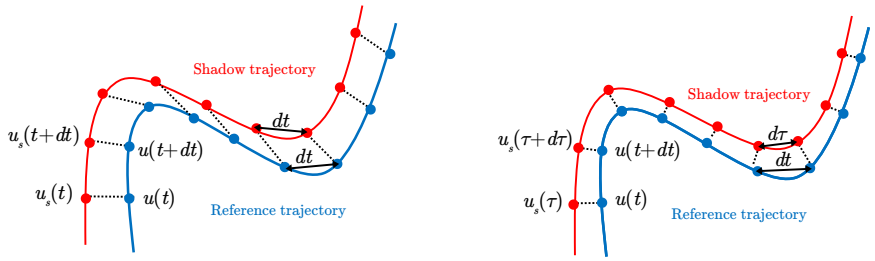
Finally, a demonstration case of shape optimization of a turbulent SD7003 airfoil was considered using LES and the binary DPA approach with MADS. Following verification of the baseline configuration, MADS was able to find a new airfoil geometry that was 32% more efficient than the baseline design. This represents a significant increase in performance and, importantly, this was achieved using high-fidelity LES as opposed to a lower-fidelity RANS-based approach. Hence, we have demonstrated that optimization using LES is feasible, and that it can be achieved using MADS. Future work will focus on exploring a larger number of design parameters and a greater range of design spaces for which conventional RANS-based flow solvers are known to be inaccurate.

Chapter 4

Sensitivity Analysis of Chaotic Systems

4.1 Overview

Computational methods in a wide range of engineering fields are valuable tools for design, optimization, control, and uncertainty quantification [54, 67, 79, 96]. Sensitivity analysis, i.e., considering the derivatives of an output of a dynamical system to entries, is the main part of optimization, or control approach, for real-world engineering problems. For engineering design, physical phenomena can be mathematically modelled in the form of Partial Derivative Equations (PDEs), which are referred to as governing equations of a dynamical system. These physical phenomena often convey complicated non-linear behaviours, making engineering design more challenging. In fluid mechanics, conventional approaches in sensitivity analysis often fail to compute practical sensitivities for scale-resolving turbulent flow simulations, such as Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) [17]. With the advent of powerful computational hardware, these high-fidelity tools have been widely used for turbulent flow analysis in science and engineering fields. Unlike low-fidelity Reynolds-Averaged Navier-Stokes (RANS) solver, LES and DNS resolve most fine structures, containing chaotic behaviour of turbulent flows [18]. However, these approaches often compute impractical sensitivity solutions due to chaotic divergence of sensitivity functions [139]. This is because there is at least one unstable mode with a positive Lyapunov exponent in the dynamical system.



(a) Without time dilation (b) With time dilation

Figure 26: Schematic of shadow and reference trajectories.

Least Squares Shadowing (LSS), proposed by Wang et al.[139], is an efficient method to evaluate sensitivities of chaotic dynamical systems [139]. LSS leverages *shadowing lemma*, and ergodicity of dynamical system to solve these sensitivities [18]. According to the *shadowing lemma*, there is one perturbed solution for any ergotic dynamical system, such that this perturbed solution sticks as close as possible to the reference one over time domain $\mathbb{T}$. Figure 26 shows a schematic of a shadow solution that tracks a reference trajectory for two different conditions. In this figure, $\boldsymbol{u} : \mathbb{T} \to \mathbb{R}^{n_u}$ is a reference solution, evolving with physical time $t \in \mathbb{T}$, and $dt \in \mathbb{R}_+$ denotes the time-step. Additionally, $\tau \in \mathbb{T}$ is a time transformation for a perturbed dynamical system, and $d\tau \in \mathbb{R}_+$ is the time-step to advance shadow solution with time, $\boldsymbol{u}_s : \mathbb{T} \to \mathbb{R}^{n_u}$. In Figure 26a, shadow and reference trajectories have the same discrete time-steps, $d\tau = dt$, yielding low accuracy results for LSS approach. This is because the distances between these solutions are not minimum over $\mathbb{T}$. On the other hand, as shown in Figure 26b, shadow solution evolves with time transformation, yielding a flexible solution in physical time that remains close to the reference one. In other words, this time transformation applies an orthogonal constraint to the perturbed dynamical system, such that the distance lines, connecting shadow and reference solutions at each time interval, become perpendicular to both trajectories over $\mathbb{T}$. LSS particularly seeks for a new solution with different initial condition that has the lowest loss function, $\frac{1}{T}\int_0^T \|\boldsymbol{u}_s - \boldsymbol{u}\|_2^2 dt$, where $[0, T] \in \mathbb{T}$ is a time window with final time $T \in \mathbb{R}_+$. Therefore, this approach paves the way to consider sensitivity analysis, uncertainty quantification, and large-scale PDE-constrained optimization of chaotic dynamical systems.

4.2 Partial Derivatives

Let us consider Eq. (1) with $\boldsymbol{u}(t,S) = [u_1(t,\mathcal{S}), u_2(t,\mathcal{S}), ..., u_{n_u}(t,\mathcal{S})]^\top$, where $u_i(t,\mathcal{S})$ with $i = 1, 2, \ldots, n_u$ is the state of the dynamical system. Suppose that the design parameters are slightly perturbed by $\delta\mathcal{S} \to \epsilon$, where, ϵ is small. The augmented state is $u_{s,i}(\tau, \mathcal{S} + \delta\mathcal{S})$, and the augmented state vector can be defined as $\boldsymbol{u}_s(\tau, \mathcal{S} + \delta\mathcal{S}) = [u_{s,1}(\tau, \mathcal{S} + \delta\mathcal{S}), u_{s,2}(\tau, \mathcal{S} + \delta\mathcal{S}), ..., u_{s,n_u}(\tau, \mathcal{S} + \delta\mathcal{S})]^\top$, where τ is a time transformation for the augmented dynamical system.

According to the *shadowing lemma*, any state vector with a significant perturbation can not be considered as the "shadow" of the reference state vector. Generally, three essential criteria should be taken into account for this purpose: (1) when $\delta\mathcal{S} \to \epsilon$, there is only one unique shadow solution, $\boldsymbol{u}_s(\tau, \mathcal{S} + \delta\mathcal{S}) : \mathbb{T} \to \mathbb{R}^{n_u}$; (2) the behaviour of $\boldsymbol{u}_s(\tau, \mathcal{S} + \delta\mathcal{S})$ remains in the flux of the augmented dynamical system, $\boldsymbol{f}_s : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$, such that these evolutional behaviours should be in $\tau \in \mathbb{T}$ and $\tau \neq t$, leading to $\nabla.\boldsymbol{f}_s : \boldsymbol{u}_s \times \tau \to \boldsymbol{u}_s$, which is often not equal to $\nabla.\boldsymbol{f} : \boldsymbol{u} \times t \to \boldsymbol{u}$; and (3) with respect to the *shadowing lemma*, the shadow solution satisfies $\parallel \boldsymbol{u}_s(\tau, \mathcal{S} + \delta\mathcal{S}) - \boldsymbol{u}(t, \mathcal{S}) \parallel^2 \leq \varepsilon$, where $\varepsilon \in \mathbb{R}_+$ is the maximum distance between the reference and augmented state vectors. These three criteria ensure the existence of a shadow for the reference state vector. Therefore, the sensitivity of the reference state with respect to a set of design parameters can be defined as follows

$$\boldsymbol{v}(t,\mathcal{S}) = \frac{\partial \boldsymbol{u}}{\partial \mathcal{S}} \approx \lim_{\delta\mathcal{S}\to 0} \frac{1}{\delta\mathcal{S}} \bigg(\boldsymbol{u}_s\big(\tau, \mathcal{S} + \delta\mathcal{S}\big) - \boldsymbol{u}(t,\mathcal{S}) \bigg). \tag{29}$$

Employing a first-order forward Finite Difference (FD) scheme, Eq. (29) indicates that there is a direct relation between the shadow and forward sensitivity solutions. Rewriting Eq. (29) in the form of a Taylor series by neglecting higher-order terms, and assuming that $\delta\mathcal{S} \to 0$, the shadow solution of the reference state vector can be obtained by

$$\boldsymbol{u}_s\big(\tau, \mathcal{S} + \delta\mathcal{S}\big) = \boldsymbol{u}(t,\mathcal{S}) + \delta\mathcal{S}\boldsymbol{v}(t,\mathcal{S}). \tag{30}$$

Finding the derivative of Eq. (30) with respect to $t \in \mathbb{T}$, the velocity of the state vector for the augmented dynamical system is given by

$$\lim_{\delta\mathcal{S}\to 0} \frac{\partial \boldsymbol{u}_s}{\partial t} = -\nabla.\boldsymbol{f}\big(\boldsymbol{u}, t, \mathcal{S}\big) + \frac{\partial \boldsymbol{v}}{\partial t}\delta\mathcal{S}. \tag{31}$$

It is worth mentioning that Eq. (31) shows that the augmented dynamical system is

proportional to the velocity of the sensitivity solution, $\frac{\partial \boldsymbol{v}}{\partial t}$. This means that the augmented dynamical system has the potential to be formulated with respect to $\boldsymbol{v}(t, \mathcal{S})$. Hence, applying the *shadowing lemma* to sensitivity analysis of chaotic dynamical systems paves the way to relate the sensitivity solution to the shadow solution of the reference state vector. Similarly, by definition of the shadow solution, the partial derivative of Eq. (1) can be written as

$$\frac{d\nabla.\boldsymbol{f}}{d\mathcal{S}} = \lim_{\delta\mathcal{S}\to 0} \frac{1}{\delta\mathcal{S}}\Big(\nabla.\boldsymbol{f}_s\big(\boldsymbol{u}_s, \tau, \mathcal{S}+\delta\mathcal{S}\big) - \nabla.\boldsymbol{f}\big(\boldsymbol{u}, t, \mathcal{S}\big)\Big) = \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}}. \quad (32)$$

Similar to Eq. (30), the shadow of the dynamical system is

$$\nabla.\boldsymbol{f}_s\Big(\boldsymbol{u}_s, \tau, \mathcal{S}+\delta\mathcal{S}\Big) = \nabla.\boldsymbol{f}\big(\boldsymbol{u}, t, \mathcal{S}\big) + \delta\mathcal{S}\Big(\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}}\Big), \quad t \in \mathbb{T}, \quad (33)$$

where Eq. (33) notes that, regardless of $\tau \in \mathbb{T}$, $\nabla.\boldsymbol{f}_s$ can be computed in the physical time, $t \in \mathbb{T}$, such that computing $\nabla.\boldsymbol{f}$ and $d\nabla.\boldsymbol{f}/d\mathcal{S}$ yields the approximation of $\nabla.\boldsymbol{f}_s$.

Based on the *shadowing lemma*, the time transformation of the shadow solution is a function of the physical time, $\tau \equiv \eta(t)$, where $\eta \in \mathbb{R}$ is the time dilation defined by

$$\eta(t) = \lim_{\delta\mathcal{S}\to 0} \frac{\delta\big(d\tau/dt - 1\big)}{\delta\mathcal{S}}. \quad (34)$$

It is worth noting that when $\delta\mathcal{S} \to 0$, the time dilation becomes negligible (i.e. $\eta \to 0$), and both the shadow and reference solutions embed in the same time domain (i.e. $\tau \cong t \in \mathbb{T}$). Integrating Eq. (34) yields

$$\int_0^{d\tau/dt-1} d\big(\frac{d\tau}{dt} - 1\big) = \int_{\mathcal{S}}^{\mathcal{S}+\delta\mathcal{S}} \eta(t) d\mathcal{S}, \quad (35)$$

and therefore, the solution of Eq. (35) is obtained by

$$\frac{d\tau}{dt} = 1 + \eta(t) d\mathcal{S}. \quad (36)$$

The same idea can also be applied to find the shadow of the objective function. Therefore, the shadow solution will help compute the sensitivity of the objective function in the design space $\mathscr{D}$. Let us assume that the shadow of the objective function is represented as $\mathcal{J}_s(\boldsymbol{u}_s, \tau, \mathcal{S}+\Delta\mathcal{S}) : \mathbb{R}^{n_u} \to \mathbb{R}$. Hence, the full derivative of the objective function is

$$\frac{d\mathcal{J}}{d\mathcal{S}} = \lim_{\delta\mathcal{S}\to 0} \frac{1}{\delta\mathcal{S}}\Big(\mathcal{J}_s\big(\boldsymbol{u}_s, \tau, \mathcal{S}+\delta\mathcal{S}\big) - \mathcal{J}\big(\boldsymbol{u}, t, \mathcal{S}\big)\Big) = \Big[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\Big]^\top \boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}, \quad (37)$$

and, similar to Eq. (30), we can represent Eq. (37) as

$$\mathcal{J}_s\Big(\boldsymbol{u}_s, \tau, \mathcal{S}+\delta\mathcal{S}\Big) = \mathcal{J}\big(\boldsymbol{u}, t, \mathcal{S}\Big) + \delta\mathcal{S}\bigg(\Big[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\Big]^\top \boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}\bigg). \tag{38}$$

Note that Eq. (29) to Eq. (38) are basic definitions in sensitivity analysis, and using them we can stabilize "ill-posed" forward sensitivity functions of chaotic dynamical systems [21].

4.3 Least Squares Shadowing (LSS)

Least-Squares Shadowing (LSS) tries to find $\boldsymbol{u}_s(\tau, \mathcal{S}+\delta\mathcal{S})$ by minimizing the distance between the trajectories of the shadow solution and the reference state vector over $[0, T] \in \mathbb{T}$. Therefore, the Least-Squares minimization formulated for LSS is given by

$$\begin{aligned} &\underset{\boldsymbol{u}_s\in\mathbb{R}^{n_u},\tau\in\mathbb{R}}{\text{minimize}} \quad \lim_{\delta\mathcal{S}\to 0}\frac{1}{T}\int_0^T \|\boldsymbol{u}_s(\tau,\mathcal{S}+\delta\mathcal{S}) - \boldsymbol{u}(t,\mathcal{S})\|^2 + \alpha_{lss}^2\|\frac{d\tau}{dt} - 1\|^2 dt, \\ &\text{subject to} \quad \frac{\partial \boldsymbol{u}_s}{\partial \tau} + \nabla.\boldsymbol{f}_s = 0, \quad \tau \in \mathbb{T}, \end{aligned} \tag{39}$$

where α_{lss} is the weighting factor of the time dilation. When $\delta\mathcal{S} \to 0$, Eq. (29) and Eq. (34) can be used to reformulate Eq. (39) as

$$\begin{aligned} &\underset{\boldsymbol{u}_s\in\mathbb{R}^{n_u},\tau\in\mathbb{R}}{\text{minimize}} \quad \lim_{\delta\mathcal{S}\to 0}\frac{1}{T}\int_0^T \|\boldsymbol{v}(t,\mathcal{S})\|^2 + \alpha_{lss}^2\|\eta(t)\|^2 dt, \\ &\text{subject to} \quad \frac{\partial \boldsymbol{u}_s}{\partial \tau} + \nabla.\boldsymbol{f}_s = 0, \quad \tau \in \mathbb{T}. \end{aligned} \tag{40}$$

The main challenge in Eq. (40) is that the augmented state vector evolves with $\tau \in \mathbb{T}$, and the shadow of the dynamical system is unknown. According to Section 4.2, the augmented state vector should be solved in the physical time, $t \in \mathbb{T}$. Using chain rule, the velocity of the state vector for augmented dynamical system can be defined as

$$\frac{\partial \boldsymbol{u}_s(\tau,\mathcal{S})}{\partial t} = \frac{\partial \tau}{\partial t}\frac{\partial \boldsymbol{u}_s(\tau,\mathcal{S})}{\partial \tau}. \tag{41}$$

According to Eq. (41), Eq. (33) and Eq. (36), the constraint function in Eq. (40) are

$$\begin{aligned} \frac{\partial \boldsymbol{u}_s}{\partial t} + \nabla.\boldsymbol{f} + \eta\delta\mathcal{S}\nabla.\boldsymbol{f} + \delta\mathcal{S}\bigg(\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}}\bigg) \\ + \eta\delta\mathcal{S}^2\bigg(\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}}\bigg) = 0. \end{aligned} \tag{42}$$

When $\delta\mathcal{S} \to 0$, the last term of Eq. (42) will be negligible, and then Eq. (42) can be simplified to

$$\frac{\partial \boldsymbol{u}_s}{\partial t} + \nabla.\boldsymbol{f} + \delta\mathcal{S}\left(\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f}\right) + \mathcal{O}(\delta\mathcal{S}^2) = 0. \tag{43}$$

Therefore, by subtituting Eq. (43) into Eq. (31), the constraint function becomes

$$\frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f} = 0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \quad t \in \mathbb{T}, \tag{44}$$

where, $\eta\nabla.\boldsymbol{f}$ shows the effect of time dilation on the augmented dynamical system. Note that if we set $\delta\mathcal{S} \to 0$, or neglect the effect of time dilation on the results (i.e., $\tau \cong t$), then $\eta\nabla.\boldsymbol{f}$ can be removed from Eq. (44). In the end, the final formulation of Eq. (40) is given by

$$\begin{aligned} \underset{\boldsymbol{v}\in\mathbb{R}^{n_u},\eta\in\mathbb{R}}{\text{minimize}} \quad & \lim_{\delta\mathcal{S}\to 0} \frac{1}{T}\int_0^T \|\boldsymbol{v}(t,\mathcal{S})\|_2^2 + \alpha_{lss}^2\|\eta(t)\|_2^2 dt \\ \text{subject to} \quad & \frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f} = 0, \quad t \in \mathbb{T}. \end{aligned} \tag{45}$$

According to Eq. (45), for a given perturbation, $\delta\mathcal{S} \to \epsilon$, LSS seeks to find the smallest values of $\boldsymbol{v}$ and η, guaranteeing the results of the augmented dynamical system are as close as possible to the shadow solution.

4.4 Lagrange Multiplier

The Lagrange function for Eq. (45) is

$$\mathcal{L} = \frac{1}{T}\int_0^T \left(\boldsymbol{v}^\top\mathbf{K}\boldsymbol{v} + \alpha_{lss}^2\eta^2\right)dt + \int_0^T \Lambda^\top\left(\frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f}\right)dt, \quad t \in \mathbb{T}, \tag{46}$$

where, $\Lambda(t) \in \mathbb{R}^{n_n}$ is the Lagrange multiplier, and $\mathbf{K} \in \mathbb{R}^{n_u \times n_u}$ is a weighting matrix. By applying the Karush-Kuhn Tucker (KKT) condition, the solution of the Lagrange function is

$$\begin{aligned} &\frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f} = 0, \\ &\frac{\partial \Lambda}{\partial t} - \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}^\top\Lambda - \frac{1}{T}\mathbf{K}^\top\boldsymbol{v} = 0, \quad \Lambda(0) = \Lambda(T) = 0, \\ &\frac{2}{T}\alpha_{lss}^2\eta + \Lambda^\top\nabla.\boldsymbol{f} = 0. \end{aligned} \tag{47}$$

Note that there are some conditions for Lambda(t) to exist. Since the physical phenomena we are looking at already satisfy these conditions, we do not investigate them. Solving Eq. (47) returns the sensitivity solution and the time dilation over $[0, T] \in \mathbb{T}$. However, Eq. (47) represents a set of non-linear ODEs that must be solved numerically in the discrete time domain, $\Delta\mathbb{T}$, such that $t^{(n)} = \{t^{(1)}, t^{(2)}, \ldots, t^{(m_u)}\} \in \Delta\mathbb{T}$, where n denotes time index, and m_u is the total number of time steps. To solve Eq. (47) numerically, we discretize it using the BDF scheme in Eq. (4). Therefore, the discrete form of Eq. (47) can be written as

$$\begin{aligned}
&\lim_{\delta\mathcal{S}\to 0} \frac{d\boldsymbol{r}^{(n)}}{d\mathcal{S}} := \sum_{j=0}^{k} \zeta_j \boldsymbol{v}^{(n-j)} + \beta_0 \Delta t \frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \boldsymbol{v}^{(n)} + \Delta t \frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \mathcal{S}} + \eta^{(n)} \Delta t \nabla.\boldsymbol{f}^{(n)} = 0, \\
&\sum_{j=0}^{k} \zeta_j \Lambda^{(n+j)} + \beta_0 \Delta t \left[\frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top} \Lambda^{(n)} + \frac{1}{m_u} \mathbf{K}^{\top} \boldsymbol{v}^{(n)} = 0, \quad \Lambda^{(0)} = \Lambda^{(m_u)} = 0, \\
&2\alpha_{lss}^2 \eta^{(n)} - m_u \Delta t \Lambda^{\top(n)} \nabla.\boldsymbol{f}^{(n)} = 0.
\end{aligned} \tag{48}$$

Rearranging Eq. (48), the following formulation is obtained

$$\begin{aligned}
&\left(\zeta_0 \mathcal{I} + \beta_0 \Delta t \frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right) \boldsymbol{v}^{(n)} + \sum_{j=1}^{k} \zeta_j \boldsymbol{v}^{(n-j)} + \beta_0 \Delta t \frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \mathcal{S}} + \eta^{(n)} \Delta t \nabla.\boldsymbol{f}^{(n)} = 0, \\
&\left(\zeta_0 \mathcal{I} + \beta_0 \Delta t \left[\frac{\partial \nabla.\boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\right) \Lambda^{(n)} + \sum_{j=1}^{k} \zeta_j \Lambda^{(n+j)} + \frac{1}{m_u} \mathbf{K}^{\top} \boldsymbol{v}^{(n)} = 0, \quad \Lambda^{(0)} = \Lambda^{(m_u)} = 0, \\
&2\alpha_{lss}^2 \eta^{(n)} - m_u \Delta t \Lambda^{\top(n)} \nabla.\boldsymbol{f}^{(n)} = 0,
\end{aligned} \tag{49}$$

where, $\mathcal{I} \in \mathbb{R}^{n_u \times n_u}$ is identity matrix. Finally, the residual form of Eq. (49) with $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$, $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(p)}}$ (where, $p \neq n$), and $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}}$ is

$$\begin{aligned}
&\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \boldsymbol{v}^{(n)} + \sum_{j=1}^{k} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n-j)}} \boldsymbol{v}^{(n-j)} + \frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}} + \eta^{(n)} \Delta t \nabla.\boldsymbol{f}^{(n)} = 0, \\
&\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top} \Lambda^{(n)} + \sum_{j=1}^{k} \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n+j)}}\right]^{\top} \Lambda^{(n+j)} + \mathbf{K}^{\top} \boldsymbol{v}^{(n)} = 0, \quad \Lambda^{(0)} = \Lambda^{(m_u)} = 0, \\
&\alpha_{lss}^2 \eta^{(n)} - m_u \Delta t \Lambda^{\top(n)} \nabla.\boldsymbol{f}^{(n)} = 0.
\end{aligned} \tag{50}$$

Note that because α_{lss} and $\mathbf{K}$ are weighting parameters, their factors (i.e., 2 and $\frac{1}{m_u}$) are embedded. For simplicity, we solve Eq. (50) separately for each $s_i \in \mathcal{S} =$

$\{s_1, s_2, \ldots, s_{n_s}\}$. Consequently, Eq. (50) can be represented as a multi-block matrix

$$\left[\begin{array}{c|c|c} \mathcal{B} & \mathcal{C} & \mathbf{0} \\ \hline \mathbf{0} & \alpha_{lss}^2\mathcal{I} & m_u\mathcal{C}^\top \\ \hline \mathbf{K}^\top & \mathbf{0} & \mathcal{B}^\top \end{array}\right] \left[\begin{array}{c} \boldsymbol{v} \\ \hline \eta \\ \hline \Lambda \end{array}\right] = - \left[\begin{array}{c} \mathcal{D} \\ \hline \mathbf{0} \\ \hline \mathbf{0} \end{array}\right], \tag{51}$$

where,

$$\begin{aligned} \mathcal{B} &= \begin{bmatrix} \frac{\partial \boldsymbol{r}^{(k)}}{\partial \boldsymbol{u}^{(1)}} & \frac{\partial \boldsymbol{r}^{(k)}}{\partial \boldsymbol{u}^{(2)}} & \cdots & \frac{\partial \boldsymbol{r}^{(k)}}{\partial \boldsymbol{u}^{(k)}} & & \\ & \frac{\partial \boldsymbol{r}^{(k+1)}}{\partial \boldsymbol{u}^{(2)}} & \frac{\partial \boldsymbol{r}^{(k+1)}}{\partial \boldsymbol{u}^{(3)}} & \cdots & \frac{\partial \boldsymbol{r}^{(k+1)}}{\partial \boldsymbol{u}^{(k+1)}} & \\ & & \ddots & & \ddots & \ddots \\ & & & \frac{\partial \boldsymbol{r}^{(m_u)}}{\partial \boldsymbol{u}^{(m_u-k)}} & \frac{\partial \boldsymbol{r}^{(m_u)}}{\partial \boldsymbol{u}^{(m_u-1)}} & \cdots & \frac{\partial \boldsymbol{r}^{(m_u)}}{\partial \boldsymbol{u}^{(m_u)}} \end{bmatrix}, \\ \mathcal{C} &= \Delta t \begin{bmatrix} \nabla.\boldsymbol{f}^{(k)} & & & \\ & \nabla.\boldsymbol{f}^{(k+1)} & & \\ & & \ddots & \\ & & & \nabla.\boldsymbol{f}^{(m_u)} \end{bmatrix}, \quad \mathcal{D} = \begin{bmatrix} \frac{\partial \boldsymbol{r}^{(k)}}{\partial s_i} \\ \frac{\partial \boldsymbol{r}^{(k+1)}}{\partial s_i} \\ \vdots \\ \frac{\partial \boldsymbol{r}^{(m_u)}}{\partial s_i} \end{bmatrix}. \end{aligned} \tag{52}$$

In Eq. (52), $\mathcal{B} \in \mathbb{R}^{n_u(m_u+1-k)\times n_u(m_u+1)}$, $\mathcal{C} \in \mathbb{R}^{n_u(m_u+1-k)\times m_u}$ are sparse multi-block matrices, and $\mathcal{D} \in \mathbb{R}^{n_u m_u}$ is a vector. Eq. (51) can be represented in lower dimensions using Gaussian elimination

$$\left[\mathcal{B}\mathcal{B}^\top + \frac{1}{\alpha_{lss}^2}\mathcal{C}\mathcal{C}^\top\right]\Lambda = \mathcal{D}, \tag{53}$$

where, $\mathcal{B}\mathcal{B}^\top \in \mathbb{R}^{n_u(m_u+1-k)\times n_u(m_u+1-k)}$ and $\mathcal{C}\mathcal{C}^\top \in \mathbb{R}^{n_u(m_u+1-k)\times n_u(m_u+1-k)}$. In general, $\mathcal{B}$ and $\mathcal{C}$ have a large number of non-zeros, which is usually the case for high-fidelity chaotic dynamical systems. Moreover, according to Eq. (53), the derivative components, $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}$ and $\frac{\partial \boldsymbol{r}}{\partial s_i}$, should be stored for all time intervals to a hard disk (e.g., Hard Disk Drive (HDD) or Solid-State Drive (SSD)), and Random Access Memory (RAM), which makes Eq. (53) prohibitively expensive to solve. For instance, the number of Jacobian matrices that should be stored in the computing device is m_u, and we require all Jacobian matrices at the same time to build $\mathcal{B}$.

4.5 Long-Term Sensitivity of the Objective Function

In order to find the long-term sensitivity of the objective function with respect to a set of design parameters, we define the vector sensitivities

$$\frac{\overline{\Delta\mathcal{J}}}{\delta\mathcal{S}} = \frac{\overline{\mathcal{J}_s} - \overline{\mathcal{J}}}{\delta\mathcal{S}}, \tag{54}$$

where, $\overline{x}$ denotes the time-averaged of $x(t)$ over $\mathbb{T}$. To obtain the sensitivities in Eq.(54), the gradient of the objective function is

$$\overline{\Delta\mathcal{J}} = \overline{\mathcal{J}_s} - \overline{\mathcal{J}} = \frac{1}{\tau(T)-\tau(0)} \int_{\tau(0)}^{\tau(T)} \mathcal{J}_s\Big(\boldsymbol{u}_s, t, \mathcal{S}+\delta\mathcal{S}\Big)dt - \frac{1}{T}\int_0^T \mathcal{J}\Big(\boldsymbol{u}, t, \mathcal{S}\Big)dt, \tag{55}$$

where, $\mathcal{J}_s$ is a function of $\tau \in \mathbb{T}$, and $\mathcal{J}$ evolves with $t \in \mathbb{T}$. Afterward, we use Eq.(36) to eliminate τ from Eq. (55). Therefore, we integrate both side of Eq.(36)

$$\int_0^T 1 + \eta(t)d\mathcal{S} = \int_{\tau(0)}^{\tau(T)} \frac{d\tau}{dt} = \tau(T) - \tau(0), \tag{56}$$

where, the linear relation of the time transformation, $\tau(T) - \tau(0)$, is expressed as an integral function. Therefore, by substituting Eq.(38) and Eq.(56) into Eq.(55), we obtain

$$\begin{aligned}\overline{\Delta\mathcal{J}} = &\frac{1}{\int_0^T 1+\eta(t)d\mathcal{S}} \int_0^T \Big(1+\eta(t)d\mathcal{S}\Big)\left[\mathcal{J}\big(\boldsymbol{u},t,\mathcal{S}\big) + \delta\mathcal{S}\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v} + \frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)\right]dt \\ &- \frac{1}{T}\int_0^T \mathcal{J}\Big(\boldsymbol{u},t,\mathcal{S}\Big)dt.\end{aligned} \tag{57}$$

Rearranging Eq.(57) as follows

$$\begin{aligned}\overline{\Delta\mathcal{J}} = &\frac{1}{\int_0^T [1+\eta(t)]d\mathcal{S}} \int_0^T \left[\eta(t)\delta\mathcal{S}\mathcal{J}(\boldsymbol{u},t,\mathcal{S}) + (\delta\mathcal{S})^2\eta(t)\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v} + \frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)\right. \\ &\left. + \delta\mathcal{S}\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v} + \frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)\right]dt - \frac{\int_0^T \eta(t)\delta\mathcal{S}dt \int_0^T \mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt}{T\int_0^T [1+\eta(t)]d\mathcal{S}},\end{aligned} \tag{58}$$

where the second term in the first integral becomes negligible due to small value of $(\delta\mathcal{S})^2$, if $\delta\mathcal{S} \to 0$. Therefore, Eq.(58) can be summarized as follows

$$\begin{aligned}\overline{\Delta\mathcal{J}} = &\frac{1}{\int_0^T [1+\eta(t)]d\mathcal{S}} \int_0^T \delta\mathcal{S}\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v} + \frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)dt \\ &+ \frac{\int_0^T \eta(t)\delta\mathcal{S}\mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt}{\int_0^T [1+\eta(t)]d\mathcal{S}} - \frac{\int_0^T \eta(t)\delta\mathcal{S}dt \int_0^T \mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt}{T\int_0^T [1+\eta(t)]d\mathcal{S}} + \mathcal{O}(\delta\mathcal{S}^2).\end{aligned} \tag{59}$$

Since $\delta\mathcal{S}$ is a known value, we can factor it out

$$\begin{aligned}\overline{\Delta\mathcal{J}} = \Bigg[&\frac{1}{\int_0^T[1+\eta(t)]d\mathcal{S}}\int_0^T\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v}+\frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)dt \\ &+\frac{\int_0^T\eta(t)\mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt}{\int_0^T[1+\eta(t)]d\mathcal{S}}-\frac{\int_0^T\eta(t)dt\int_0^T\mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt}{T\int_0^T[1+\eta(t)]d\mathcal{S}}\Bigg]\delta\mathcal{S}+\mathcal{O}(\delta\mathcal{S}^2),\end{aligned} \tag{60}$$

and we can assume that the amount of $\eta(t)$ would be negligible over time, since $\delta\mathcal{S}\to 0$. Therefore, Eq.(60) can be represented as

$$\begin{aligned}\frac{\overline{\Delta\mathcal{J}}}{\delta\mathcal{S}} = &\frac{1}{T}\int_0^T\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v}+\frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)dt+\frac{1}{T}\int_0^T\eta(t)\mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt \\ &-\frac{1}{T^2}\int_0^T\eta(t)dt\int_0^T\mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt.\end{aligned} \tag{61}$$

Finally, by rearranging those terms in Eq.(61), we obtain

$$\frac{\overline{\Delta\mathcal{J}}}{\delta\mathcal{S}} = \frac{1}{T}\int_0^T\left[\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v}+\frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)+\eta(t)\Big(\mathcal{J}(\boldsymbol{u},t,\mathcal{S})-\overline{\mathcal{J}}\Big)\right]dt. \tag{62}$$

We apply Eq. (62) to find the sensitivities of the objective function with respect to $\mathcal{S}$. The second term in Eq. (62) is responsible for the effect of the time transformation on the shadow solution. This term is often small if $\delta\mathcal{S}\to 0$, but it might become important when highly sensitive objective functions are considered. Another simplified form of Eq.(62) can be cast as

$$\frac{\overline{\Delta\mathcal{J}}}{\delta\mathcal{S}} = \overline{\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{u}}\boldsymbol{v}+\frac{\partial\mathcal{J}}{\partial\mathcal{S}}\right)}+\overline{\eta\mathcal{J}}-\overline{\mathcal{J}}\,\overline{\eta}. \tag{63}$$

In this Chapter, implementing the *shadowing lemma* in a form of the mathematical formulation will let us consider sensitivity analysis and optimization in the presence of chaotic dynamical systems. Since conventional LSS is often prohibitively expensive for scale-resolving problems, we will introduce a novel framework to implement this idea with a feasible amount of computational resources for large-scale problems.

Chapter 5

Dimensionality Reduction & Closure Models

5.1 Overview

Expensive computational costs of high-fidelity Full-Order Models (FOMs) led to the emergence of surrogate models, referred to as Reduced-Order Models (ROMs), to approximate the solution of FOMs. Proper Orthogonal Decomposition(POD) [38, 73], Dynamic Mode Decomposition(DMD) [114, 122], Balanced truncation [90, 92], Eigen Realization Analysis (ERA) [69, 70, 110] are common techniques that have been developed, and applied successfully to a wide range of engineering problems. However, these techniques are data-driven, with a lack of information from the governing equations. In other words, the ROM identifies the evolutionary behaviour of the dynamical system using linear models and raw data to approximate the solution of the FOM. However, most engineering problems, such as fluid physics and turbulence, are highly non-linear or chaotic. Therefore, in the form of the ROM, those linear models often fail to predict the dynamics of non-linear systems over long time domains.

Physics-constrained techniques apply the governing equations (i.e. Navier-Stokes equations) directly to build the ROM. These techniques are referred to as *projection-based* approaches. In these approaches, the solution from lower dimensional space is projected into a higher dimensional one, and vice versa. One of the famous *projection-based* approaches is Galerkin projection, which encompasses the governing equations in its infrastructures. While the Galerkin projection is the simplest one, it has been

successfully applied to various engineering and scientific problems [37, 47, 115]. However, it loses its stability, accuracy, and convergence to predict non-linear and chaotic systems [118]. This failure results from the lack of the *priori* property [61]. This means that adding more subspaces (i.e., modes) to the model does not necessarily improve or guarantee the accuracy of the solution.

Recently, the Least-Squares Petrov Galerkin (LSPG) approach was developed [30], which is a promising *projection-based* approach to overcome stability issues in the ROM of non-linear dynamical systems. Specifically, LSPG relies on the least-squares residual minimization for model reduction of these non-linear systems [30]. The LSPG is defined as a fully discrete approach, and it solves for the solution in lower dimensional space by minimizing the residual of the ROM, where this residual is projected into higher dimensional space at each discrete time step. Furthermore, this approach inherently includes physical constraints, such as conservation, since it is formulated as a minimization problem [31]. Although LSPG has the potential to be applied to the ROM for highly non-linear and chaotic systems, it has some limitations. Firstly, it does not guarantee *priori* stability, which means that it is still unclear how many modes, or subspaces, are required to obtain the best computational performance [29, 34, 62]. Secondly, solutions predicted via LSPG are significantly influenced by time integrator, and time-step [29]. Thirdly, when explicit time integrators are employed, this LSPG mimics the Galerkin projection and, hence, it becomes unstable. Therefore, the application of the LSPG is only limited to implicit time integrators [29], resulting in higher computational costs compared to those with explicit schemes.

5.1.1 The Weak Form of the Full-Order Model (FOM)

In order to build a closure model in the form of the projection-based ROM, we need to develop the weak form of the FOM. The FOM is in high-dimensional physical space $\mathscr{P}$, but can be decomposed into multiple lower-dimensional subspaces by the Galerkin method. First, a trial basis function is defined as a mapping function between $\mathscr{P}$ and a non-dimensional space $\mathscr{H}$ (i.e., Hilbert space). This trial basis function includes a set of orthogonal bases

$$\mathbf{\Phi} = \begin{bmatrix} | & | & & | \\ \phi_1 & \phi_2 & \dots & \phi_{n_u} \\ | & | & & | \end{bmatrix}, \tag{64}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n_u \times n_u}$ and $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \mathcal{I}$. These orthogonal vectors can be generated by Proper Orthogonal Decomposition (POD). Since $\boldsymbol{\Phi}$ is a full rank matrix, the state vector can be defined as a set of linear combinations of these orthogonal vectors

$$\boldsymbol{u}(t) = \overline{\boldsymbol{u}} + \sum_{i=1}^{n_u} \phi_i q_i(t) = \overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}, \tag{65}$$

where, $\overline{\boldsymbol{u}} \in \mathbb{R}^{n_u}$ is the reference vector of the state, and $\boldsymbol{q} : \mathbb{T} \to \mathbb{R}^{n_u}$ is the generalized coordinates of the trial basis function with $\boldsymbol{q} = [q_1(t), q_2(t), \ldots, q_{n_u}(t)]^\top$. In a similar way, we define a test basis function, as a full rank matrix

$$\boldsymbol{\Psi} = \begin{bmatrix} | & | & & | \\ \psi_1 & \psi_2 & \ldots & \psi_{n_u} \\ | & | & & | \end{bmatrix}, \tag{66}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{n_u \times n_u}$, and its columns are linearly independent vectors. Eq. (1) can be represented in the form of the generalized coordinates by substituting Eq. (65) into Eq. (1), which yields

$$\boldsymbol{\Phi}\frac{\partial \boldsymbol{q}}{\partial t} + \nabla.\boldsymbol{f}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}, t, \mathcal{S}) = 0, \quad \boldsymbol{q}(0) = \boldsymbol{q}_0, \quad t \in \mathbb{T}, \tag{67}$$

where, $\boldsymbol{q}_0 \in \mathbb{R}^{n_u}$ with $\boldsymbol{\Phi}\boldsymbol{q}_0 \mapsto \boldsymbol{u}_0$ is the initial condition. The weak form of Eq. (67) can also be represented as

$$\boldsymbol{\Psi}^\top\boldsymbol{\Phi}\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{\Psi}^\top\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}, t, \mathcal{S}) = 0, \quad \boldsymbol{q}(0) = \boldsymbol{q}_0, \quad t \in \mathbb{T}. \tag{68}$$

From a mathematical perspective, first we define the exact representation of the FOM in $\mathscr{H}$ as $\mathbb{F} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$. In other words, $\mathbb{F}$ is the weak form of $\nabla.\boldsymbol{f}$ that evolves in $\mathscr{H}$. then we employ Eq. (68) as a closure model to project the state vector of the FOM from $\mathscr{P}$ into the generalized coordinates in $\mathscr{H}$, such that $\mathbb{F} : \boldsymbol{q} \times \mathbb{T} \to \boldsymbol{q}$ with $\boldsymbol{q} \mapsto \mathbb{F}(\boldsymbol{q})$.

5.1.2 Dimensionality Reduction

Significant evolutional behaviours of a dynamical system are usually governed by dominant features, structures, or patterns. If we decompose the trial and test basis functions into multiple lower-dimensional subspaces, these dominant structures are common in the *coarse-scale* subspaces. Therefore, based on this concept, we can

rearrange vectors (i.e., columns) in the test and trial basis functions based on their priority (i.e., according to the eigenvalue of each column). Next, we select the first r rank of vectors (where, $r << n_u$) from the test and trial basis functions as the *coarse-scale* subspaces, and consider the remaining as the *fine-scale* subspaces. These basis functions are

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix} | & | & & | \\ \phi_1 & \phi_2 & \cdots & \phi_r \\ | & | & & | \end{bmatrix}, \quad \boldsymbol{\Phi}' = \begin{bmatrix} | & | & & | \\ \phi_{r+1} & \phi_{r+2} & \cdots & \phi_{n_u} \\ | & | & & | \end{bmatrix}, \tag{69}$$

$$\tilde{\boldsymbol{\Psi}} = \begin{bmatrix} | & | & & | \\ \psi_1 & \psi_2 & \cdots & \psi_r \\ | & | & & | \end{bmatrix}, \quad \boldsymbol{\Psi}' = \begin{bmatrix} | & | & & | \\ \psi_{r+1} & \psi_{r+2} & \cdots & \psi_{n_u} \\ | & | & & | \end{bmatrix}, \tag{70}$$

where $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{n_u \times r}$ and $\boldsymbol{\Phi}' \in \mathbb{R}^{n_u \times (n_u - r)}$ are the *coarse-scale* and *fine-scale* trial subspaces, respectively. Additionally, $\tilde{\boldsymbol{\Psi}} \in \mathbb{R}^{n_u \times r}$ and $\boldsymbol{\Psi}' \in \mathbb{R}^{n_u \times (n_u - r)}$ are defined in the same manner. The test and trial basis functions can also be written as

$$\boldsymbol{\Phi} = \tilde{\boldsymbol{\Phi}} \oplus \boldsymbol{\Phi}', \quad \text{and} \quad \boldsymbol{\Psi} = \tilde{\boldsymbol{\Psi}} \oplus \boldsymbol{\Psi}'. \tag{71}$$

It is worth pointing out that decomposition of space into two distinct subspaces should have two important features: firstly, the subspaces do not overlap, i.e. $\tilde{\boldsymbol{\Phi}} \cap \boldsymbol{\Phi}' = 0$, and secondly, they are orthogonal, i.e. $\tilde{\boldsymbol{\Phi}} \perp \boldsymbol{\Phi}'$. This is because the vectors in the basis function are orthogonal. However, the test basis function may not have the second feature. In a similar way, the state vector can also be decomposed into the *coarse-scale* and *fine-scale* subspaces

$$\tilde{\boldsymbol{u}}(t) = \sum_{i=1}^{r} \tilde{\Phi}_i \tilde{q}_i(t) = \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{q}}, \tag{72}$$

$$\boldsymbol{u}'(t) = \sum_{i=r+1}^{N} \Phi'_i q'_i(t) = \boldsymbol{\Phi}' \boldsymbol{q}', \tag{73}$$

where, $\tilde{\boldsymbol{u}} : \mathbb{T} \to \mathbb{R}^{n_u}$, $\boldsymbol{u}' : \mathbb{T} \to \mathbb{R}^{n_u}$, $\tilde{\boldsymbol{q}} : \mathbb{T} \to \mathbb{R}^r$ and $\boldsymbol{q}' : \mathbb{T} \to \mathbb{R}^{n_u - r}$. Also, $\boldsymbol{u} = \overline{\boldsymbol{u}} + \tilde{\boldsymbol{u}} + \boldsymbol{u}'$ represents the linear relation between the reference vector, *coarse-scale* and *fine-scale* solutions. Hence, representing the state vector of the FOM in multi-scale subspaces allows us to write Eq. (68) as two linearly independent systems

$$\tilde{\boldsymbol{\Psi}}^{(n)\top} \tilde{\boldsymbol{\Phi}} \frac{\partial \tilde{\boldsymbol{q}}}{\partial t} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \boldsymbol{\Phi}' \frac{\partial \boldsymbol{q}'}{\partial t} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \nabla . \boldsymbol{f}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{q}} + \boldsymbol{\Phi}' \boldsymbol{q}', t, \mathcal{S}) = 0, \quad t \in \mathbb{T}, \tag{74}$$

$$\boldsymbol{\Psi}'^{\top}\tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t}+\boldsymbol{\Psi}'^{\top}\boldsymbol{\Phi}'\frac{\partial\boldsymbol{q}'}{\partial t}+\boldsymbol{\Psi}'^{\top}\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}+\boldsymbol{\Phi}'\boldsymbol{q}',t,\mathcal{S})=0,\quad t\in\mathbb{T},\tag{75}$$

where a combination of Eq. (74) and Eq. (75) builds the wake form of the FOM. In other words, these equations are the *coarse-scale* and *fine-scale* functions of the FOM in $\mathscr{H}$.

In order to build a closure model in the form of the ROM, the first assumption is that the dynamical system contains only the *coarse-scale* solutions (i.e., $\boldsymbol{u}'\approx 0$). This assumption turns the weak form of the FOM into a new equation for only the *coarse-scale* dynamical features that are embedded in $\mathscr{H}$. On the other hand, the evolution of these *coarse-scale* features depends on the *fine-scale* dynamical features [108]. This issue is addressed as a closure problem that should be taken into account. Therefore, the closure model is obtained by

$$\tilde{\boldsymbol{\Psi}}^{\top}\tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t}+\tilde{\boldsymbol{\Psi}}^{\top}\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}},t,\mathcal{S})=0,\quad \tilde{\boldsymbol{q}}(0)=\tilde{\boldsymbol{q}}_0,\quad t\in\mathbb{T}.\tag{76}$$

5.2 Galerkin Projection

Galerkin projection is a technique to build the ROM, where lower dimensional subspaces are created using the POD technique, which is performed as the *offline* part of the ROM [12]. Subsequently, the *online* part of the ROM is to solve for the solution in constructed subspaces [12]. This approach is not practical if the number of physical modes in these subspaces is small. Additionally, for long term approximation, this method might become unstable [13]. In Galerkin projection, the test basis and trial basis functions are equivalent (i.e. $\tilde{\boldsymbol{\Psi}}=\tilde{\boldsymbol{\Phi}}$). Therefore, Eq. (76) will be simplified to

$$\frac{\partial\tilde{\boldsymbol{q}}}{\partial t}+\tilde{\boldsymbol{\Phi}}^{\top}\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}},t,\mathcal{S})=0,\quad \tilde{\boldsymbol{q}}(t=0)=\tilde{\boldsymbol{q}}_0.\tag{77}$$

If temporal discretization is based on the residual minimization at each discrete time-step, $t^{(n)}\in\Delta\mathbb{T}$, Eq. (77) can be written in the form of the residual, $\boldsymbol{r}\colon\mathbb{R}^{n_u}\to\mathbb{R}^{n_u}$, as follows

$$\tilde{\boldsymbol{\Phi}}^{\top}\boldsymbol{r}^{(n)}:=\tilde{\boldsymbol{\Phi}}^{\top}\left[\tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t}+\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}},t,\mathcal{S})\right]=0,\quad \tilde{\boldsymbol{q}}^{(0)}=\tilde{\boldsymbol{q}}_0.\tag{78}$$

We assume that those vectors in $\tilde{\boldsymbol{\Phi}}$ are orthogonal, $\tilde{\boldsymbol{\Phi}}^{\top}\tilde{\boldsymbol{\Phi}}=\mathcal{I}$, otherwise the full projection should be taken into account via $\left(\tilde{\boldsymbol{\Phi}}^{\top}\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Phi}}$. Additionally, the initial condition for the ROM is approximated by $\boldsymbol{q}_0=\tilde{\boldsymbol{\Phi}}^{\top}\boldsymbol{u}^{(0)}$.

It is shown that this Galerkin projection is a continuous optimal technique, such that it minimizes ℓ^2-norm of the high dimensional solution over the trial basis subspaces [30]. Due to this optimal property, adding more trial basis vectors to the basis function, $\tilde{\boldsymbol{\Phi}} \rightarrow \boldsymbol{\Phi}$, yields a monotonic reduction in the ℓ^2-norm of the residual [29]. Recall Eq. (3), the ROM using this Galerkin projection is represented in the discrete form

$$\tilde{\boldsymbol{\Phi}}^\top \boldsymbol{r}^{(n)} := \sum_{j=0}^{k} \zeta_j \tilde{\boldsymbol{q}}^{(n-j)} + \Delta t \sum_{j=0}^{k} \beta_j \tilde{\boldsymbol{\Phi}}^\top \nabla.\boldsymbol{f}\left(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n-j)}, t^{(n-j)}, \mathcal{S}\right) = 0, \tag{79}$$

where, Eq. (79) indicates that the residual increases due to the accumulation of numerical errors from the previous time intervals. As a matter of fact, the *coarse-scale* subspaces produce excessive errors, influencing the dynamics in $\nabla.\boldsymbol{f}$. However, this residual for the linear dynamical system is negligible, and the ROM is often stable. On the other hand, the ROM becomes unstable if the dynamical system is highly non-linear or chaotic. Therefore, having poor optimality conditions restricts the application of Galerkin projection to linear dynamical systems.

5.3 Least Squares Petrov-Galerkin Projection

In the weak form of a FOM, if we apply a discrete time integrator, it will be possible to check and control optimality conditions at all $t^{(n)} \in \Delta\mathbb{T}$. This discrete optimality concept raises up as a key step in the closure modelling, such that ℓ^2-norm of the residual at each time interval is minimized. According to Eq. (3), the residual of the dynamical system at each time interval is

$$\boldsymbol{r}^{(n)} := \frac{\Delta \boldsymbol{u}^{(n)}}{\Delta t^{(n)}} + \nabla.\boldsymbol{f}(\boldsymbol{u}^{(n)}, t^{(n)}, \mathcal{S}) \approx 0, \quad \boldsymbol{u}^{(0)} = \boldsymbol{u}_0, \tag{80}$$

where $\Delta t^{(n)} \in \Delta\mathbb{T}$ is the time-step at $t^{(n)}$. Here, we use a constant time-step for all time intervals, $\Delta t^{(n)} = \Delta t$. To solve for this dynamical system in time, we need to minimize the residual in Eq. (80) at each time interval. To this end, we employ the Petrov-Galerkin approach to convert Eq. (80) into a minimization problem

$$\mathcal{W}^\top \boldsymbol{r}^{(n)} = \mathcal{W}^\top \left(\frac{\Delta \boldsymbol{u}^{(n)}}{\Delta t} + \nabla.\boldsymbol{f}(\boldsymbol{u}^{(n)}, t^{(n)}, \mathcal{S}) \right) = 0, \quad \boldsymbol{u}^{(0)} = \boldsymbol{u}_0, \tag{81}$$

where $\mathcal{W} \in \mathbb{R}^{n_u \times n_u}$ is a weighting function that enforces $\boldsymbol{r}^{(n)}$ to be minimum over $\Delta\mathbb{T}$. Solving this minimization problem using the Petrov-Galerkin approach with

optimality condition, where ℓ^2-norm of the residual is considered, is called Least-Squares Petrov-Galerkin (LSPG) approach. Therefore, LSPG formulation is written as follows

$$\boldsymbol{u}^{(n)} \in \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \mathbf{A}(\boldsymbol{z}) \boldsymbol{r}^{(n)}(\boldsymbol{z}, t^{(n)}, \mathcal{S}) \right\|_2^2, \quad \text{and} \quad n = 1, \ldots, m_u, \tag{82}$$

where m_u is the number of time intervals, and $\mathbf{A} \in \mathbb{R}^{n_u \times n_u}$ is a weighting matrix that enables definition of weighted norm [33]. The weighting matrix can also be used for hyper-reduction at which we can only select a finite number of the states, and compute the corresponding residual at those states [34]. In standard LSPG, the residual for all states are considered, $\mathbf{A} = \mathcal{I}$.

We can extend the application of the LSPG to solve for the generalized coordinates in lower subspaces as follows

$$\tilde{\boldsymbol{q}}^{(n)} \in \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \mathbf{A}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}} \boldsymbol{z}) \boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}} \boldsymbol{z}, t^{(n)}, \mathcal{S}) \right\|_2^2, \quad \text{and} \quad n = 1, \ldots, m_u, \tag{83}$$

where the residual is a full-rank vector, $\boldsymbol{r}^{(n)} \in \mathbb{R}^{n_u}$, while the vector of the generalized coordinates is r-rank, $\boldsymbol{q}^{(n)} \in \mathbb{R}^r$, which means that this minimization is under-determined problem. Recall Eq. (3), we discretize Eq. (81), and cast it into

$$\mathcal{W}^\top \left(\sum_{j=0}^{k} \zeta_j \boldsymbol{u}^{(n-j)} + \Delta t^{(n)} \sum_{j=0}^{k} \beta_j \nabla . \boldsymbol{f}^{(n)} \Big(\boldsymbol{u}^{(n-j)}, t^{(n-j)}, \mathcal{S} \Big) \right) = 0, \tag{84}$$

and we use Eq. (65) to rewrite Eq. (84) as

$$\mathcal{W}^\top \sum_{j=0}^{k} \zeta_j \tilde{\boldsymbol{\Phi}} \boldsymbol{q}^{(n-j)} + \mathcal{W}^\top \left(\Delta t^{(n)} \sum_{j=0}^{k} \beta_j \nabla . \boldsymbol{f}^{(n)} \Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi} \boldsymbol{q}^{(n-j)}, t^{(n-j)}, \mathcal{S} \Big) \right) = 0. \tag{85}$$

Since the trial basis function, $\tilde{\boldsymbol{\Phi}}$, remains constant over $\Delta \mathbb{T}$, then Eq. (85) is presented as follows

$$\mathcal{W}^\top \tilde{\boldsymbol{\Phi}} \sum_{j=0}^{k} \zeta_j \boldsymbol{q}^{(n-j)} + \mathcal{W}^\top \left(\Delta t^{(n)} \sum_{j=0}^{k} \beta_j \nabla . \boldsymbol{f}^{(n)} \Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi} \boldsymbol{q}^{(n-j)}, t^{(n-j)}, \mathcal{S} \Big) \right) = 0. \tag{86}$$

Comparing Eq. (86) and Eq. (68), we realize that the ROM developed via LSPG is the same as the weak form of the FOM. Accordingly, we conclude that $\mathcal{W} = \tilde{\boldsymbol{\Psi}}$. Therefore, Eq. (86) for a ROM with r-rank is

$$\tilde{\boldsymbol{\Psi}}^{(n)\top} \tilde{\boldsymbol{\Phi}} \sum_{j=0}^{k} \zeta_j \boldsymbol{q}^{(n-j)} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \left(\Delta t^{(n)} \sum_{j=0}^{k} \beta_j \nabla . \boldsymbol{f}^{(n)} \Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi} \boldsymbol{q}^{(n-j)}, t^{(n-j)}, \mathcal{S} \Big) \right) = 0. \tag{87}$$

In implicit temporal schemes, such as Backward Differentiation Formula (BDF), the second term on the righ-hand side of Eq. (87) is not considered, $\forall j > 0 \colon \beta_j = 0$. Therefore, Eq. (87) can be simplified to

$$\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\sum_{j=0}^{k}\zeta_j \boldsymbol{q}^{(n-j)} + \Delta t^{(n)}\beta_0 \tilde{\boldsymbol{\Psi}}^{(n)\top}\nabla.\boldsymbol{f}^{(n)}\Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}\Big) = 0, \quad \tilde{\boldsymbol{q}}^{(0)} = \tilde{\boldsymbol{q}}_0. \tag{88}$$

Finally, we realize that Eq. (88) is a specific form of Eq. (76), when the BDF shceme is used.

5.4 LSPG vs. Galerkin Projection

To obtain the continuous form of the LSPG, we muliply both sides of Eq. (76) by $\left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}$

$$\frac{\partial \tilde{\boldsymbol{q}}}{\partial t} + \Big(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\Big)^{-1}\tilde{\boldsymbol{\Psi}}^\top\nabla.\boldsymbol{f}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}) = 0, \quad \tilde{\boldsymbol{q}}^{(0)} = \tilde{\boldsymbol{q}}_0. \tag{89}$$

Similar to Eq. (89), we use the BDF scheme to discretize Eq. (88)

$$\sum_{j=0}^{k}\zeta_j \boldsymbol{q}^{(n-j)} + \Delta t^{(n)}\beta_0\Big(\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\Big)^{-1}\tilde{\boldsymbol{\Psi}}^{(n)\top}\nabla.\boldsymbol{f}^{(n)}\Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}\Big) = 0, \quad \tilde{\boldsymbol{q}}^{(0)} = \tilde{\boldsymbol{q}}_0. \tag{90}$$

It is shown that the test basis function for either continuous or discrete form of the LSPG is dependent on time [29]

$$\tilde{\boldsymbol{\Psi}} = \mathbf{A}^\top\mathbf{A}\Big(\zeta_0\mathcal{I} + \Delta t\beta_0\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\Big)\tilde{\boldsymbol{\Phi}}, \tag{91}$$

where $\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}$ is the Jacobian matrix. The optimality condition applied to the ROM at all time intervals is given by

$$\tilde{\boldsymbol{\Psi}}^{(n)\top}\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}) = 0. \tag{92}$$

By assuming $\mathbf{A} = \mathcal{I}$, and substituting Eq. (91) into Eq. (92), we obtain

$$\tilde{\boldsymbol{\Phi}}^\top\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}) + \Delta t\beta_0\tilde{\boldsymbol{\Phi}}^\top\left[\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\right]^\top\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}) = 0. \tag{93}$$

Here, we can distinguish Eq. (93) into two distinct terms. The first term is Galerkin projection, and the second term is for stabilization. This means that LSPG adds an additional term to the Galerkin projection to stabilize it over $\Delta\mathbb{T}$. Moreover, this

stabilization term is proportional to residual, which means the LSPG is a residual-based method. It is worth pointing out that the stabilization term is used to control disorders in the ROM due to eliminating the *fine-scale* modes in the dynamical system. Therefore, LSPG converts the Galerkin projection into a closure problem. Some features of this closure model, which is developed via LSPG, are enlisted as follows

1. When $\Delta t \to 0$, LSPG turns into a Galerkin projection. Therefore, intermediate Δt leads to optimal accuracy [29].

2. For explicit schemes, $\beta_0 \to 0$, both LSPG and Galerkin projection return equivalent results.

3. LSPG method is strongly dependent on temporal discretization. Therefore, a different temporal scheme results in a different test basis function. Hence, the stabilization term will perform differently for each scheme [108].

The relation between the residual of the ROM and that of the FOM in $\mathscr{H}$ and $\mathscr{P}$ is

$$\mathcal{R}^{(n)}(\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) = \tilde{\boldsymbol{\Psi}}^{\top} \boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}), \tag{94}$$

where $\mathcal{R} : \mathbb{R}^r \to \mathbb{R}^r$, and $r \ll n_u$. Eq. (94) indicates that those terms in the ROM do not necessarily need to be known. In spite of finding unknow terms in this ROM, we can indirectly take them into account using $\mathcal{R}^{(n)}(\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S})$. We will use Eq. (94) later to obtain a closure model, in the form of a ROM, for sensitivity analysis.

5.5 Gauss-Newton Method

The Gauss-Newton method is an iterative mathematical approach to solve non-linear least-squares problems. The residual of a FOM at each time interval is defined as a cost function, $\boldsymbol{r}_{gn}^{(n)} \in \mathbb{R}^{n_u}$. This cost function in the form of minimization is defined as

$$\begin{aligned} &\min_{\tilde{\boldsymbol{q}}_i^{(n)}} \quad \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)}) \\ &\text{subject to } \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)}) = \frac{1}{2}\|\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\|_2^2 \\ &\qquad\qquad = \boldsymbol{r}^{(n)\top}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S}), \end{aligned} \tag{95}$$

where index i denotes the number of inner iterations in Gauss-Newton's minimization. The Gauss-Newton uses the first derivative of the cost function with respect to the solution

$$\nabla \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)}) = \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \tilde{\boldsymbol{q}}_i^{(n)}}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\right]^{\top} \boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S}), \tag{96}$$

and also its second derivative

$$\begin{aligned}\nabla^2 \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)}) = &\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \tilde{\boldsymbol{q}}_i^{(n)}}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\right]^{\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \tilde{\boldsymbol{q}}_i^{(n)}}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S}) \\ &+ \sum_{i=1}^{(n)} \frac{\partial^2 \boldsymbol{r}^{(n)}}{\partial^2 \tilde{\boldsymbol{q}}_i^{(n)}}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S}),\end{aligned} \tag{97}$$

where, the second derivative of this cost function is approximated due to its complication. Therefore, the second term in Eq. (97) is neglected, and the inner iteration in the Gauss-Newton method is obtained by

$$\tilde{\boldsymbol{q}}_{i+1}^{(n)} = \tilde{\boldsymbol{q}}_i^{(n)} + \Delta\tilde{\boldsymbol{q}}_{i+1}^{(n)}, \quad \nabla^2 \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)})\Delta\tilde{\boldsymbol{q}}_{i+1}^{(n)} = \nabla \boldsymbol{r}_{gn}^{(n)}(\tilde{\boldsymbol{q}}_i^{(n)}). \tag{98}$$

Alternatively, the normal form of Eq. (98) can be expressed as

$$\Delta\tilde{\boldsymbol{q}}_{i+1}^{(n)} \in \underset{z \in \mathbb{R}^r}{\operatorname{argmin}} \left\| \frac{\partial \boldsymbol{r}^{(n)}}{\partial \tilde{\boldsymbol{q}}_i^{(n)}}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S})\mathbf{z} + \boldsymbol{r}^{(n)}(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}_i^{(n)}, t^{(n)}, \mathcal{S}) \right\|_2^2. \tag{99}$$

Chapter 6

Gradient-Based Optimization

6.1 Overview

Inspired by Koopman theory, we will apply a similar approach to develop a framework for optimization of non-linear systems. Figure 27 shows a general approach for optimization problems. Conventionally optimization problems are solved in physical space, $\mathscr{P}$. Consider the left-hand side object as the initial design in Figure 27. To optimize the shape of this object, we solve the optimization problem subject to a Hamiltonian dynamical system defined as $\nabla.\mathcal{H}(\boldsymbol{u}, \mathcal{S})$. In this example, $\boldsymbol{u}$ and $\mathcal{S}$ represent state and shape parameters, respectively. The right-hand side object in $\mathscr{P}$ is the optimized shape. According to the properties of a Hamiltonian system, the dynamical system for each object has a unique representation in Hilbert space $\mathscr{H}$. Usually, this representation is as a set of generalized coordinates, $\boldsymbol{q}$, that define the dynamical behaviour of the corresponding object in $\mathscr{H}$. Therefore, in order to find the representation of the reference object, we project its corresponding state vector from $\mathscr{P}$ to $\mathscr{H}$ using a projection function $\mathbb{P}$. We assume that $\mathbb{F}$ is a surrogate model that represents the exact evolution of the state vector in $\mathscr{H}$, such that that $\mathbb{F} : \boldsymbol{q} \times \mathbb{T} \rightarrow \boldsymbol{q}$ with $(\boldsymbol{q}, t) \mapsto \mathbb{F}(\boldsymbol{q}, t)$ in $t \in \mathbb{T}$. Now if we apply this optimization problem subject to $\mathbb{F}$, we obtain the optimized set of generalized coordinates. In the end, we lift-back these optimized generalized coordinates from $\mathscr{H}$ to $\mathscr{P}$ via $\mathbb{P}'$, obtaining the same optimized object in $\mathscr{P}$. It is worth mentioning that $\mathbb{F}$ is unknown, and we should build a closure model with rank $r \in \mathbb{R}_+$ given by $\mathcal{F} : \mathbb{R}^r \rightarrow \mathbb{R}^r$, such that $\mathcal{F} : \boldsymbol{q} \times \mathbb{T} \rightarrow \boldsymbol{q}$, in the form of a ROM to approximate $\mathbb{F}$. The closer $\mathcal{F}$ to $\mathbb{F}$, the

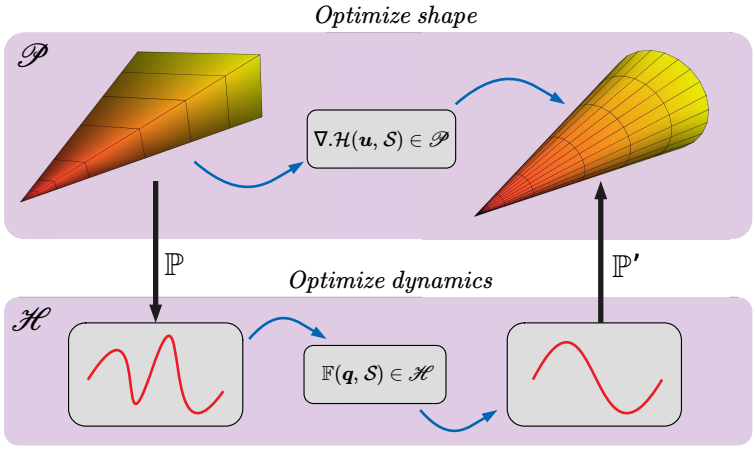


Figure 27: Optimization of the shape of an arbitrary object with subject to a Hamiltonian system in physical space $\mathscr{P}$ versus optimization of dynamics with subject to a surrogate model in Hilbert space $\mathscr{H}$.

closer the optimization results will be to the true optimal design in $\mathscr{P}$.

In gradient-based optimization, sensitivity analysis plays a significant role in obtaining the gradients of the objective function with respect to the design parameters. These gradients are used in the optimizer (i.e., Newton's methods). The general purpose of the aforementioned concept is that gradient-based optimization often fails in the presence of large-scale chaotic dynamical systems. On the other hand, conventional LSS is extremely expensive for real-world engineering problems. Hence, we want to develop a framework that allows us to apply this LSS to sensitivity analysis of large-scale chaotic systems with less computational costs and memory requirements. As a FOM, the state vector of a high-dimensional dynamical system can be decomposed into steady and unsteady terms. The steady term is the reference state vector, and computed via $\overline{\boldsymbol{u}} = \frac{1}{T}\int_0^\top \boldsymbol{u}(t,\mathcal{S})dt$. The unsteady terms, the *coarse-scale* and *fine-scale*, can also be defined as a set of linear combinations of those orthogonal vectors in the trial basis function. The reference state vector and the trial basis function are only dependent on the design parameters, $\mathcal{S}$. Although these terms are computed from the high-dimensional unsteady dataset in the *offline* step (building the ROM), they remain constant in the *online* step (employing the ROM for analysis). On the contrary, the generalized coordinates in the *coarse-scale* and *fine-scale* are dependent on both the design parameters and time. Therefore, the general form of the state

vector can be defined as

$$\boldsymbol{u}(t,\mathcal{S}) = \overline{\boldsymbol{u}} + \underbrace{\tilde{\boldsymbol{\Phi}}(\mathcal{S})\tilde{\boldsymbol{q}}(t,\mathcal{S})}_{coarse-scale} + \underbrace{\boldsymbol{\Phi}'(\mathcal{S})\boldsymbol{q}'(t,\mathcal{S})}_{fine-scale}, \tag{100}$$

where these separated terms in Eq. (100) will help us distinguish between different types of variables, dominant patterns, coherent structures, and the evolutionary behaviours of system in different subspaces. For instance, in turbulent flow, velocity magnitude can be decomposed into different flow structures (i.e., modes), eddies (i.e. turbulent length), vortex shedding pattern (i.e., dynamics). With this classification, we can recognize the influence of each aforementioned term on the turbulent flow.

6.2 Gradient-Based Optimization for PDE

6.2.1 Optimization Using the Forward Sensitivity Function

In PDE-constrained optimization, the objective function, $\mathcal{J}(\boldsymbol{u},\mathcal{S}) : \mathbb{R}^{n_u} \rightarrow \mathbb{R}$, is constrained by the primal PDEs. Since the time-averaged value of the objective function, $\overline{\mathcal{J}}$, is of interest in engineering problems, primal PDEs are considered in the range of $[0,T] \in \mathbb{T}$, where $T \in \mathbb{R}_+$ denotes the final solution time. Here $\overline{x}$ denotes the time-averaged of x. Additionally, in order to reduce the magnitude of $\overline{\mathcal{J}}$, the optimizer seeks better design parameters in the design space, $\mathcal{S} = [s_1, s_2, \dots s_{n_s}]^\top \subset \mathscr{D}$, at each design cycle. This optimization problem can be defined as

$$\begin{aligned} \underset{\substack{\boldsymbol{u}\in\mathbb{R}^{n_u}\\ \mathcal{S}\in\mathscr{D}}}{\text{minimize}} \quad & \overline{\mathcal{J}} = \frac{1}{T}\int_0^T \mathcal{J}(\boldsymbol{u},t,\mathcal{S})dt, \\ \text{subject to} \quad & \frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f}(\boldsymbol{u},\nabla\boldsymbol{u},\mathcal{S}) = 0, \quad t \in \mathbb{T}, \\ & \mathbf{C}(\boldsymbol{u},\mathcal{S}) \leq 0, \end{aligned} \tag{101}$$

where $\mathbf{C}$ represents constraint functions that can be dependent on either the state vector or design parameters. For optimization, we need to find the sensitivity of the state vector with respect to the set of design parameters, $\frac{\partial \boldsymbol{u}}{\partial \mathcal{S}} : \mathbb{R}^{n_u} \times \mathbb{T} \rightarrow \mathbb{R}^{n_u \times n_s}$, which can be obtained by dual representation of the primal PDEs

$$\frac{d}{d\mathcal{S}}\left(\frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f}\right) = 0, \quad t \in \mathbb{T}. \tag{102}$$

Let us suppose that the primal PDEs are the Navier-Stokes equations. In this case, a set of the dual PDEs is the "forward sensitivity" function that describes the evolution of the sensitivity solution with respect to the design parameters over $\mathbb{T}$. Therefore, the forward sensitivity function, in the form of dual PDEs, is given by

$$\frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} = 0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \quad t \in \mathbb{T}, \tag{103}$$

where, $\boldsymbol{v} : \mathbb{R}^{n_u} \times \mathbb{T} \to \mathbb{R}^{n_u \times n_s}$ is the sensitivity solution, and $\boldsymbol{v}_0 \in \mathbb{R}^{n_u \times n_s}$ is the initial condition. Moreover, Eq. (103) illustrates that the sensitivity solution has its own specific dynamical features, which are connected to those of the state vector in Eq. (1). Additionally, Eq. (103) is advanced in time with $\boldsymbol{v} \times 0 \to \boldsymbol{v}_0$ as the initial condition at time $0 \in \mathbb{T}$. Therefore, $\boldsymbol{v}$ is often called the forward sensitivity solution.

According to Eq. (101), in order to find the time-averaged sensitivity of the objective function with respect to $\mathcal{S}$, the gradient of the objective function is computed by

$$\frac{\delta \mathcal{J}}{\delta \mathcal{S}} = \Big[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\Big]^\top \boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}, \tag{104}$$

where $\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u \times n_s}$. Substituting Eq. (104) into Eq. (101), we can obtain the time-averaged sensitivity of the objective function by

$$\frac{\overline{\Delta \mathcal{J}}}{\Delta \mathcal{S}} = \frac{1}{T}\int_0^T \left(\Big[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\Big]^\top \boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}\right) dt \quad \text{s.t.} \quad \frac{d}{d\mathcal{S}}\left(\frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f}\right) = 0. \tag{105}$$

Therefore, the final formulation of a PDE-constrained optimization using the forward sensitivity function is

$$\begin{aligned} \underset{\substack{\boldsymbol{u},\,\boldsymbol{v} \in \mathbb{R}^{n_u} \\ \mathcal{S} \in \mathscr{D}}}{\text{minimize}} \quad & \overline{\mathcal{J}} = \frac{1}{T}\int_0^T \mathcal{J}(\boldsymbol{u}, t, \mathcal{S}) dt, \\ \text{subject to} \quad & \frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f} = 0, \quad t \in \mathbb{T}, \\ & \frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} = 0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0, \quad t \in \mathbb{T}, \\ & \mathbf{C}(\boldsymbol{u}, \mathcal{S}) \leq 0. \end{aligned} \tag{106}$$

The objective of this study is to show how to compute $\frac{\overline{\Delta \mathcal{J}}}{\Delta \mathcal{S}}$, which is the primary challenge in sensitivity analysis and uncertainty quantifications of chaotic problems. Note that the minimization set in Eq. (106) requires a Hessian matrix, which can be approximated via a Quasi-Newton method series, such as Broyden-Fletcher-Goldfarb-Shanno (BFGS), and Conjugate Gradient (CG), developed for gradient-based optimizations.

6.2.2 Optimization Using the Backward Sensitivity Function

According to Section 6.2.1, the unsteady sensitivity solution must be computed for all design parameters, $\mathcal{S} = [s_1, s_2, \dots s_{n_s}]^\top \subset \mathscr{D}$, such that a total number of n_s sensitivity functions must be solved. With increasing n_s, optimization process becomes computationally expensive, and sometimes for a large number of design parameters (e.g., $n_s > 100$) it becomes impractical to launch this optimization. Therefore, we need to solve for the sensitivities in such a way that it becomes independent of the number of design parameters. This approach leads to develop the backward sensitivity function, referred to as *adjoint method*. To obtain the backward sensitivity function, we combine Eq. (104) and Eq. (103), and rewrite it

$$\overline{\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}}} = \frac{1}{T}\int_0^T \left(\left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^\top \boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}} \right) dt + \int_0^T \boldsymbol{w}^\top \left(\frac{\partial \boldsymbol{v}}{\partial t} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} \right) dt, \quad (107)$$

where $\boldsymbol{w} \in \mathbb{R}^{n_u}$ is a set of weighting factors, which is called *adjoint solutions*. By applying integration-by-part and some rearrangements, we obtain

$$\overline{\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}}} = \int_0^T \left(\frac{1}{T}\left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^\top - \frac{\partial \boldsymbol{w}^\top}{\partial t} + \boldsymbol{w}^\top \frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}} \right) \boldsymbol{v}\, dt + \int_0^T \left(\frac{1}{T}\frac{\partial \mathcal{J}}{\partial \mathcal{S}} + \boldsymbol{w}^\top \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} \right) dt + \boldsymbol{w}^\top \boldsymbol{v}|_0^T. \quad (108)$$

In *adjoint method*, we enforce $\boldsymbol{w}(T) = 0$ to eliminate the last term in Eq. (108). Furthermore, to get rid of $\boldsymbol{v}$ from Eq. (108), the *adjoint solutions* are solved in such a way that the first expression in the first parenthesis remains zero over $t \in \mathbb{T}$. Therefore, the backward sensitivity function is given by

$$-\frac{\partial \boldsymbol{w}}{\partial t} + \left[\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\right]^\top \boldsymbol{w} + \frac{1}{T}\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}} = 0, \quad \boldsymbol{w}(T) = 0, \quad (109)$$

where the initial condition is $\boldsymbol{w}(T) = 0$, which means that we should solve Eq. (109) for $\boldsymbol{w}$ backward in time. As seen from Eq. (109), the backward sensitivity function is independent of $\mathcal{S}$. This means that we need to solve Eq. (109) only once, regardless of the number of design parameters. Moreover, the time-averaged sensitivity of the objective function is obtained by

$$\overline{\frac{\delta \mathcal{J}}{\delta \mathcal{S}}} = \int_0^T \left(\frac{1}{T}\frac{\partial \mathcal{J}}{\partial \mathcal{S}} + \boldsymbol{w}^\top \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} \right) dt. \quad (110)$$

Therefore, the gradient-based optimization using the backward sensitivity function is defined as

$$\begin{aligned}
&\underset{\substack{\boldsymbol{u},\,\boldsymbol{w}\,\in\,\mathbb{R}^{n_u}\\ \mathcal{S}\,\in\,\mathscr{D}}}{\text{minimize}} \quad \overline{\mathcal{J}} = \frac{1}{T}\int_0^T \mathcal{J}(\boldsymbol{u}, t, \mathcal{S})dt \\
&\text{subject to} \quad \frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f} = 0, \quad t \in \mathbb{T}, \\
&\qquad\qquad -\frac{\partial \boldsymbol{w}}{\partial t} + \left[\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{\top}\boldsymbol{w} + \frac{1}{T}\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}} = 0, \quad \boldsymbol{w}(T) = 0, \quad t \in \mathbb{T}, \\
&\qquad\qquad \mathbf{C}(\boldsymbol{u}, \mathcal{S}) \leq 0,
\end{aligned} \tag{111}$$

6.2.3 Steady-State Optimization

In Sections 6.2.1 and 6.2.2, the closed-form of the forward and the backward unsteady sensitivity functions were discussed. To obtain the steady-state sensitivity function, we represent the non-linear dynamical system in the form of a steady-state problem

$$\nabla.\boldsymbol{f}(\boldsymbol{u}, \nabla.\boldsymbol{u}, \mathcal{S}) = 0. \tag{112}$$

In this formulation, solutions are only dependent on $\mathcal{S}$. In fluid mechanics, this dynamical system could be a simplified version of the Navier-Stokes equations, such as the Reynolds-Averaged Navier-Stokes (RANS) or steady-state Euler equations. Therefore, the forward sensitivity solution for Eq. (103) is

$$\boldsymbol{v} = -\left[\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{-1}\frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}}, \tag{113}$$

and the sensitivity of the objective function with respect to $\mathcal{S}$ is given by

$$\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}} = \left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^{\top}\boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}. \tag{114}$$

The forward sensitivity solution from Eq. (113) can be substituted into Eq. (114)

$$\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}} = -\left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^{\top}\left[\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{-1}\frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}. \tag{115}$$

Additionally, with reference to Eq. (109), the steady-state backward sensitivity solution is obtained by

$$\boldsymbol{w} = -\left[\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{-\top}\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}, \tag{116}$$

and accordingly, the sensitivity of the objective function with respect to $\mathcal{S}$, but for the steady-state backward sensitivity function, is given by

$$\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}} = \frac{\partial \mathcal{J}}{\partial \mathcal{S}} + \boldsymbol{w}^{\top} \frac{\partial \nabla . \boldsymbol{f}}{\partial \mathcal{S}}. \tag{117}$$

Finally, substituting Eq. (116) into Eq. (117), we obtain

$$\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}} = -\left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^{\top} \left[\frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{-1} \frac{\partial \nabla . \boldsymbol{f}}{\partial \mathcal{S}} + \frac{\partial \mathcal{J}}{\partial \mathcal{S}}. \tag{118}$$

According to Eq. (115) and Eq. (118), sensitivity analysis using both the forward and the backward functions yields the same formulation if the dynamical system is independant of the metric directions (i.e., x, y, and z in spatial domain), but approaches will be different. However, this *adjoint method* is usually preferred for sensitivity analysis, because of its economical computational costs. The most computationally expensive term in sensitivity analysis is the inverse of Jacobian matrix (i.e. $\frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}}$). To handle this issue, numerical methods, such as Jacobi or Gauss-Seidel, are used to solve these sensitivity functions. Finally, optimization problem using the steady-state forward sensitivity function is defined as

$$\begin{aligned} \underset{\substack{\boldsymbol{u},\, \boldsymbol{v} \,\in\, \mathbb{R}^{n_u} \\ \mathcal{S} \,\in\, \mathscr{D}}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{u}, \mathcal{S}) \\ \text{subject to} \quad & \nabla . \boldsymbol{f} = 0, \quad t \in \mathbb{T}, \\ & \boldsymbol{v} \in \underset{\boldsymbol{z} \,\in\, \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}} \boldsymbol{z} + \frac{\partial \nabla . \boldsymbol{f}}{\partial \mathcal{S}} \right\|_2^2, \quad t \in \mathbb{T}, \\ & \mathbf{C}(\boldsymbol{u}, \mathcal{S}) \leq 0, \end{aligned} \tag{119}$$

where the forward sensitivity function is represented in the form of least-squares minimization problem. Similarly, optimizaiton problem using the steady-state backward sensitivity function is given by

$$\begin{aligned} \underset{\substack{\boldsymbol{u},\, \boldsymbol{w} \,\in\, \mathbb{R}^{n_u} \\ \mathcal{S} \,\in\, \mathscr{D}}}{\text{minimize}} \quad & \mathcal{J}(\boldsymbol{u}, \mathcal{S}) \\ \text{subject to} \quad & \nabla . \boldsymbol{f} = 0, \quad t \in \mathbb{T}, \\ & \boldsymbol{w} \in \underset{\boldsymbol{z} \,\in\, \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \left[\frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}}\right]^{\top} \boldsymbol{z} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{u}} \right\|_2^2, \quad t \in \mathbb{T}, \\ & \mathbf{C}(\boldsymbol{u}, \mathcal{S}) \leq 0. \end{aligned} \tag{120}$$

Although we will not use these steady-state optimizations, their formulations will be useful to obtain manifolds in Hilbert space, which will be discussed in Chapter 7.

6.3 Sensitivity Analysis Using Closure Models

6.3.1 Sensitivity of Invariants

In Section 5.1.1, the high-dimensional state vector was decomposed into three different parts. This section describes the sensitivity of a high-dimensional state vector with respect to a set of design parameters. Finding derivatives of Eq. (65) with respect to $\mathcal{S}$, and writting it in a discrete form yields

$$\boldsymbol{v}^{(n)} = \frac{\partial \boldsymbol{u}^{(n)}}{\partial \mathcal{S}} = \overline{\boldsymbol{v}} + \frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}} \tilde{\boldsymbol{q}}^{(n)} + \frac{\partial \boldsymbol{\Phi}'}{\partial \mathcal{S}} \boldsymbol{q}'^{(n)} + \tilde{\boldsymbol{\Phi}} \frac{\partial \tilde{\boldsymbol{q}}^{(n)}}{\partial \mathcal{S}} + \boldsymbol{\Phi}' \frac{\partial \boldsymbol{q}'^{(n)}}{\partial \mathcal{S}}, \tag{121}$$

where, the first term in the right-hand side of Eq. (121), $\overline{\boldsymbol{v}} \in \mathbb{R}^{n_u \times n_s}$, shows the reference sensitivity solution. Additionally, $\frac{\partial \tilde{\boldsymbol{q}}^{(n)}}{\partial \mathcal{S}} : \mathbb{R}^r \times \mathbb{T} \rightarrow \mathbb{R}^{r \times n_s}$ and $\frac{\partial \boldsymbol{q}'^{(n)}}{\partial \mathcal{S}} : \mathbb{R}^{(n_u - r)} \times \mathbb{T} \rightarrow \mathbb{R}^{(n_u - r) \times n_s}$ denote the sensitivity of the *coarse-scale* and *fine-scale* generalized coordinates with respect to $\mathcal{S}$. Also, $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}} \in \mathbb{R}^{n_u \times r \times n_s}$ and $\frac{\partial \boldsymbol{\Phi}'}{\partial \mathcal{S}} \in \mathbb{R}^{n_u \times (n_u - r) \times n_s}$ indicate tensors containing the sensitivity of the *coarse-scale* and *fine-scale* trial basis functions with respect to $\mathcal{S}$. The sensitivities of the trial basis function describe how a design parameter can influence the dominant structures/modes of the dynamical system. By defining $\tilde{\boldsymbol{h}}^{(n)} = \frac{\partial \boldsymbol{q}^{(n)}}{\partial \mathcal{S}}$, we can write Eq. (121) as

$$\boldsymbol{v}^{(n)} = \overline{\boldsymbol{v}} + \frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}} \tilde{\boldsymbol{q}}^{(n)} + \frac{\partial \boldsymbol{\Phi}'}{\partial \mathcal{S}} \boldsymbol{q}'^{(n)} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{h}}^{(n)} + \boldsymbol{\Phi}' \boldsymbol{h}'^{(n)}, \tag{122}$$

where we can define $\tilde{\boldsymbol{h}}$ as the sensitivity of the relevant coherent stuctures that exist in the dynamical system with respect to $\mathcal{S}$. In other words, $\tilde{\boldsymbol{h}}$ indicates how the evolutional behaviours of a dynamical system changes in $\mathscr{H}$ by any perturbation, if the manifold (i.e., spatio-stuctures) remains unchanged in $\mathscr{P}$.

In this study, we want to relate the sensitivity solution of the FOM to that of the ROM by neglecting the *fine-scale* trial and test basis functions. Therefore, Eq. (122) can be summarized as follows

$$\boldsymbol{v}^{(n)} \approx \overline{\boldsymbol{v}} + \frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}} \tilde{\boldsymbol{q}}^{(n)} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{h}}^{(n)}, \tag{123}$$

where the total sensitivity of a high-dimensional state vector is decomposed into three different terms. The first and second terms in Eq. (123) are almost stable, since their derivatives remain unchanged over time. The first term, $\overline{\boldsymbol{v}}$, is stable because it is computed from the ensemble value of high-dimensional state datasets. Furthermore,

the second term $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}}$ is only dependent on the design parameters and does not change over time. However, the third term could be highly unstable due to the relevant chaotic behaviour of the FOM, which is projected into $\tilde{\boldsymbol{h}}^{(n)}$. Note that if all dominant modes are not in the *coarse-scale* trial basis function, the error in Eq. (123) grows over time. Hence, having a poor trial basis function results in a spurious dynamical system that does not produce proper dynamics compared to the actual system. On the other hand, developing a poor trial basis function increases the growth rate of errors in the ROM, since transformation of the solutions between $\mathscr{P}$ and $\mathscr{H}$ frequently occurs at each time interval. In the end, since chaotic dynamical systems have unstable modes with positive LEs, we will observe even more severe growth of these errors in Eq. (123).

It is worth mentioning that $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}}$ is a challenging term, since there is no direct solution for it. The only possible way to approximate it is to collect data from several FOM simulations that have different perturbations to the design parameters, i.e., $\delta\mathcal{S}$. Then the trial basis function for each dataset is computed. Finally, a FD method can be applied to these trial basis functions to approximate $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}}$. In this case, we need several high-dimensional simulations, which increases computational cost. Therefore, it is not practical to find $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}}$ directly. To address this issue, first we suppose that the trial basis function can be used for many perturbed systems in the design space. In that case, the model will approximate sensitivity solutions without computing $\frac{\partial \tilde{\boldsymbol{\Phi}}}{\partial \mathcal{S}}$. With this assumption, the sensitivity solution is

$$\boldsymbol{v}^{(n)} \approx \overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{h}}^{(n)}, \tag{124}$$

and subsequently, we will train the trial basis function in such a way that it becomes flexible to perturbations in the design space.

6.3.2 Descrete Form of Sensitivity Equation

In this section, the residual formulation of the ROM is developed as a set of ODEs, and then its Largrange function is found. Afterward, these ODEs are discretized, which is referred to as OΔEs. To this end, we assume that $\mathbb{F}$ is the exact representation of the FOM in $\mathscr{H}$, then we have $\mathbb{F} : \tilde{\boldsymbol{q}} \times \mathbb{T} \to \tilde{\boldsymbol{q}}$ with $(\tilde{\boldsymbol{q}}, t) \mapsto \mathbb{F}(\tilde{\boldsymbol{q}}, t)$ over $t \in \mathbb{T}$. As a matter of fact, $\mathbb{F}$ is not available, and we can only approximate it using $\mathcal{F} : \mathbb{R}^r \to \mathbb{R}^r$ as a ROM. In this case, we suppose that $\mathcal{F} \equiv \mathbb{F}$, and can then formulate the discrete

form of the ROM according to Eq. (4). Hence a set of OΔEs is obtained by

$$\mathcal{R}^{(n)} := \sum_{j=0}^{k} \zeta_j \tilde{\boldsymbol{q}}^{(n-j)} + \Delta t^{(n)} \beta_0 \mathcal{F}^{(n)}(\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) + \Omega^{(n)}, \quad \tilde{\boldsymbol{q}}(t^{(0,\dots,k)}) = \tilde{\boldsymbol{q}}_{0,\dots,k}, \quad t^{(n)} \in \Delta\mathbb{T}, \tag{125}$$

where $\mathcal{R} : \mathbb{R}^r \to \mathbb{R}^r$ is the residual of the ROM, Ω represents a source term, and $\tilde{\boldsymbol{q}}_{0,\dots,k}$ are the initial conditions, such that $\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}_{0,\dots,k} \mapsto \boldsymbol{u}_{0,\dots,k}$. Note that $\|\mathcal{R}^{(n)}\|_2$ converges to $\|\boldsymbol{r}^{(n)}\|_2$ when the number of subspaces in the trial and test basis functions is sufficiently large, i.e., $r \to n_n$. Therefore, the residual of the ROM is given by

$$\lim_{r \to n_u} \mathcal{R}^{(n)}(\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) \approx \boldsymbol{r}^{(n)}(\boldsymbol{u}^{(n)}, t^{(n)}, \mathcal{S}). \tag{126}$$

According to the *shadowing lemma*, if a ROM with the reference generalized coordinates, $\tilde{\boldsymbol{q}}^{(n)}$, is perturbed by $\mathcal{S} + \delta\mathcal{S}$, then there is a vector of the shadow solution, $\tilde{\boldsymbol{q}}_s^{(n)} : \mathbb{T} \to \mathbb{R}^r$, that satisfies Eq. (125) with another time transformation, $\tau^{(n)} \in \Delta\mathbb{T}$. This definition for the shadow of the generalized coordinates is

$$\lim_{r \to n_u} \lim_{\delta\mathcal{S} \to 0} \lim_{\delta\tau \to 0} \mathcal{R}^{(n)}(\tilde{\boldsymbol{q}}_s^{(n)}, \tau^{(n)}, \mathcal{S} + \delta\mathcal{S}) \approx 0, \quad \tau^{(n)} \in \Delta\mathbb{T}. \tag{127}$$

Recall Eq. (29), and we can find the limit of the generalized coordinates with respect to the perturbation as

$$\tilde{\boldsymbol{h}}^{(n)} \approx \lim_{\delta\mathcal{S} \to 0} \frac{\tilde{\boldsymbol{q}}_s^{(n)} - \tilde{\boldsymbol{q}}^{(n)}}{\delta\mathcal{S}}, \tag{128}$$

and, similar to Eq. (44), we can apply the *shadowing lemma* to Eq. (125). In this case, the low-dimensional sensitivity function in the form of OΔEs can be obtained by

$$\lim_{\delta\mathcal{S} \to 0} \frac{d\mathcal{R}^{(n)}}{d\mathcal{S}} := \sum_{j=0}^{k} \frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n-j)}} \tilde{\boldsymbol{h}}^{(n-j)} + \frac{\partial \mathcal{R}^{(n)}}{\partial \mathcal{S}} + \frac{\partial \mathcal{R}^{(n)}}{\partial \Omega^{(n)}} + \mathcal{E}^{(n)} \Delta t \mathcal{F}^{(n)} = 0, \quad t^{(n)} \in \Delta\mathbb{T}, \tag{129}$$

where, $\mathcal{E} \in \mathbb{R}$ is the time dilation of the generalized coordinates. Finding $\mathcal{F}$ in Eq. (125) could be prohibitively expensive. However, the advantage of formulating these OΔEs into the residual form is that we do not need to find $\mathcal{F}$ directly. Interestingly, we can apply the Petrov-Galerkin approach to minimize $\|\mathcal{R}^{(n)}\|_2$, such that errors, which are produced by dimensionality reduction, remain minimal over $\Delta\mathbb{T}$.

The Lagrange function for LSS minimization is given by

$$\begin{aligned}\mathcal{L} = &\frac{1}{m_u}\sum_{n=0}^{m_u}\left[\tilde{\boldsymbol{h}}^{(n)\top}\mathbf{K}\tilde{\boldsymbol{h}}^{(n)} + \left(\alpha_{lss}\mathcal{E}^{(n)}\right)^2\right] \\ &+ \sum_{n=0}^{m_u}\lambda^{(n)\top}\left[\sum_{j=0}^{k}\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n-j)}}\tilde{\boldsymbol{h}}^{(n-j)} + \frac{\partial\mathcal{R}^{(n)}}{\partial\mathcal{S}} + \frac{\partial\mathcal{R}^{(n)}}{\partial\Omega^{(n)}} + \mathcal{E}^{(n)}\Delta t\mathcal{F}^{(n)}\right],\end{aligned} \tag{130}$$

where $\lambda^{(n)}$ is the discrete Lagrange multiplier. Also, the KKT condition is applied to Eq. (130), which yields

$$\begin{aligned}&\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n)}}\tilde{\boldsymbol{h}}^{(n)} + \sum_{j=1}^{k}\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n-j)}}\tilde{\boldsymbol{h}}^{(n-j)} + \frac{\partial\mathcal{R}^{(n)}}{\partial\mathcal{S}} + \frac{\partial\mathcal{R}^{(n)}}{\partial\Omega^{(n)}} + \mathcal{E}^{(n)}\Big(\frac{\Delta t}{\beta_0}\mathcal{R}^{(n)} - \sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n-j)}\Big) = 0, \\ &\left[\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n)}}\right]^\top\lambda^{(n)} + \sum_{j=1}^{k}\left[\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n+j)}}\right]^\top\lambda^{(n+j)} + \mathbf{K}^\top\tilde{\boldsymbol{h}}^{(n)} = 0, \quad \lambda^{(0)} = \lambda^{(m_u)} = 0, \\ &\alpha_{lss}^2\mathcal{E}^{(n)} - m_u\lambda^{\top(n)}\Big(\frac{\Delta t}{\beta_0}\mathcal{R}^{(n)} - \sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n-j)}\Big) = 0.\end{aligned} \tag{131}$$

Since Eq. (131) is developed in the form of a low-dimensional closure model, it needs less computational resources, which is a promising factor for applying Eq. (131) to sensitivity analysis of large-scale chaotic systems. We can solve Eq. (131) using Eq. (52) at which the size of multi-block matrices are reduced to $\mathcal{B} \in \mathbb{R}^{r(m_u-k)\times rm_u}$, $\mathcal{C} \in \mathbb{R}^{rm_u\times m_u}$, and $\mathcal{D} \in \mathbb{R}^{rm_u}$, where $r \ll n_u$.

Although we eliminated $\mathcal{F}$ from Eq. (131) by rewriting it as $\mathcal{F} = \mathcal{R} - \frac{\partial\boldsymbol{q}}{\partial t}$, there is an alternative method to find $\mathcal{F}$. Additionally, there is an unknown term in Eq. (131), which is the derivative of the residual with respect to the source term, $\frac{\partial\mathcal{R}}{\partial\Omega}$. Therefore, in order to find $\mathcal{F}$ and $\frac{\partial\mathcal{R}}{\partial\Omega}$ directly from the FOM, we project the first equation in Eq. (48) into $\mathscr{H}$. then the projected results should be equal to the first OΔE in Eq. (131). Therefore, we can write

$$\begin{aligned}\lim_{\delta\mathcal{S}\to 0}\tilde{\boldsymbol{\Psi}}^{(n)\top}\frac{d\boldsymbol{r}^{(n)}}{d\mathcal{S}} &= \frac{d\mathcal{R}^{(n)}}{d\mathcal{S}} \\ &= \sum_{j=0}^{k}\frac{\partial\mathcal{R}^{(n)}}{\partial\tilde{\boldsymbol{q}}^{(n-j)}}\tilde{\boldsymbol{h}}^{(n-j)} + \frac{\partial\mathcal{R}^{(n)}}{\partial\mathcal{S}} + \tilde{\boldsymbol{\Psi}}^{(n)\top}\sum_{j=0}^{k}\frac{\partial\boldsymbol{r}^{(n)}}{\partial\boldsymbol{u}^{(n-j)}}\overline{\boldsymbol{v}} + \eta^{(n)}\Delta t\tilde{\boldsymbol{\Psi}}^{(n)\top}\nabla.\boldsymbol{f}^{(n)},\end{aligned} \tag{132}$$

and comparing Eq. (131) and Eq. (132), $\mathcal{F}$ and $\frac{\partial \mathcal{R}}{\partial \Omega}$ can be obtained by

$$\begin{aligned} \frac{\partial \mathcal{R}^{(n)}}{\partial \Omega^{(n)}} &= \tilde{\boldsymbol{\Psi}}^{(n)\top} \sum_{j=0}^{k} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n-j)}} \overline{\boldsymbol{v}}, \\ \mathcal{E}^{(n)} \Delta t \mathcal{F}^{(n)} &= \eta^{(n)} \Delta t \tilde{\boldsymbol{\Psi}}^{(n)\top} \nabla . \boldsymbol{f}^{(n)}. \end{aligned} \tag{133}$$

From Eq. (133), the unknown terms in $\mathscr{H}$ are directly derived from the known terms in $\mathscr{P}$. Therefore, it suggests that the same procedure can be applied to remaining unknown terms in $\mathscr{H}$. In Petrov-Galerkin projection, the relation between the FOM and its corresponding ROM is given by $\mathcal{R}^{(n)} = \tilde{\boldsymbol{\Psi}}^{(n)\top} \boldsymbol{r}^{(n)}$ at all time intervals over $\Delta\mathbb{T}$. Consequently, derivatives of the residuals are

$$\begin{aligned} \frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n)}} &= \Big(\Big[\frac{\partial \tilde{\boldsymbol{\Psi}}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \Big]^{\top} \boldsymbol{r}^{(n)} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \Big) \frac{\partial \boldsymbol{u}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n)}}, \\ \frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(p)}} &= \Big(\Big[\frac{\partial \tilde{\boldsymbol{\Psi}}^{(n)}}{\partial \boldsymbol{u}^{(p)}} \Big]^{\top} \boldsymbol{r}^{(n)} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(p)}} \Big) \frac{\partial \boldsymbol{u}^{(p)}}{\partial \tilde{\boldsymbol{q}}^{(p)}}, \quad n \neq p, \\ \frac{\partial \mathcal{R}^{(n)}}{\partial \mathcal{S}} &= \Big(\Big[\frac{\partial \tilde{\boldsymbol{\Psi}}^{(n)}}{\partial \mathcal{S}} \Big]^{\top} \boldsymbol{r}^{(n)} + \tilde{\boldsymbol{\Psi}}^{(n)\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}} \Big). \end{aligned} \tag{134}$$

Solving the low-dimensional closure model using the LSPG with enough subspaces causes the residual of the FOM to become approximately zero, such that for all $t^{(n)} \in \Delta\mathbb{T}$ we have $\lim_{r \to n_u} \boldsymbol{r}^{(n)}(\tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{q}}^{(n)}) \approx 0$. This definition causes the first term in the derivatives to be negligible. On the other hand, we can unanimously assume that $\frac{\partial \boldsymbol{u}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n)}} = \tilde{\boldsymbol{\Phi}}$. Hence, with regard to the definition of the test basis function, $\tilde{\boldsymbol{\Psi}}^{(n)} = \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \tilde{\boldsymbol{\Phi}}$, we can simplify Eq. (134) as follows

$$\begin{aligned} \frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n)}} &= \tilde{\boldsymbol{\Psi}}^{(n)\top} \tilde{\boldsymbol{\Psi}}^{(n)}, \\ \frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(p)}} &= \zeta_p \tilde{\boldsymbol{\Psi}}^{(n)\top} \tilde{\boldsymbol{\Phi}}, \quad n \neq p, \\ \frac{\partial \mathcal{R}^{(n)}}{\partial \mathcal{S}} &= \tilde{\boldsymbol{\Psi}}^{(n)\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}}. \end{aligned} \tag{135}$$

In the end, substituting Eq. (135) into Eq. (131), we can find the the final form of

the OΔEs

$$
\begin{aligned}
&\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Psi}}^{(n)}\tilde{\boldsymbol{h}}^{(n)} + \sum_{j=1}^{k}\zeta_j\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}}^{(n-j)} + \tilde{\boldsymbol{\Psi}}^{(n)\top}\frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}} \\
&\qquad + \tilde{\boldsymbol{\Psi}}^{(n)\top}\Big(\sum_{j=1}^{k}\zeta_j\mathcal{I} + \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\Big)\overline{\boldsymbol{v}} - \mathcal{E}^{(n)}\sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n-j)} = 0, \\
&\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Psi}}^{(n)}\lambda^{(n)} + \sum_{j=1}^{k}\zeta_j\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\lambda^{(n+j)} + \mathbf{K}^{\top}\tilde{\boldsymbol{h}}^{(n)} = 0, \quad \lambda^{(0)} = \lambda^{(m_u)} = 0, \\
&\alpha_{lss}^2\mathcal{E}^{(n)} - m_u\lambda^{(n)\top}\sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n-j)} = 0,
\end{aligned}
\tag{136}
$$

which results in the final form of the sensitivity function in $\mathscr{H}$. The amount of error in Eq. (136) depends on the error bounds in LSPG [29], and the error bounds for dimensionality reduction. The total sensitivity of the objective function with respect to a set of design parameters can be represented as

$$
\frac{\overline{\Delta\mathcal{J}}}{\Delta\mathcal{S}} \approx \frac{1}{T}\int_0^T \left[\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\right]^{\top}(\overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}})dt + \frac{1}{T}\int_0^T \mathcal{E}\Big(\mathcal{J}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}) - \overline{\mathcal{J}}\Big)dt + \frac{1}{T}\int_0^T \frac{\partial \mathcal{J}}{\partial \mathcal{S}}dt,
\tag{137}
$$

and the discrete form of Eq. (137) is given by

$$
\frac{\overline{\Delta\mathcal{J}}}{\Delta\mathcal{S}} \approx \frac{1}{m_u}\sum_{n=1}^{m_u}\left[\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}(\overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}}^{(n)}) + \frac{1}{m_u}\sum_{n=1}^{m_u}\mathcal{E}^{(n)}\Big(\mathcal{J}^{(n)} - \overline{\mathcal{J}}\Big) + \frac{1}{m_u}\sum_{n=1}^{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \mathcal{S}}.
\tag{138}
$$

As shown in Eq. (138), the first term approximates the high-dimension sensitivity solution. The second term is the sensitivity of the time transformation. Finally, the last term represents the sensitivity of the objective function with respect to the design parameters, regardless of any perturbation in the state vector.

Chapter 7

Physics-Constrained Reduced-Order Models

7.1 Reduced-Order Models

Conventional ROMs are usually linear, and approximate the behaviour of the dynamical system using a linear combination of independent flow patterns obtained by spatio-temporal techniques (e.g., POD modes). On the other hand, it is observed that many unsteady flows evolve on lower-dimensional manifolds [102, 103], which indicates that the non-linear behaviour of these dynamical systems is in lower dimensions. This non-linearity increases for fluids with increasing Reynolds number. This requires non-linear closure models, since linear models often fail to predict these dynamics.

Conventional methods, which work based on data-driven approaches, usually fail to build the exact shape of the manifolds in Hilbert space. This is because datasets are usually limited, and may not provide all information required to build an accurate ROM. In this study, to build a closure model in lower dimensions, we explicitly derive the overall structures of manifolds in Hilbert space from the physical governing equations, and then shape them via a training dataset. This strategy is called the physics-constrained data-driven approach, and can result in an accurate prediction of the closure model with less training data. Hence, we will leverage this approach to address downsides of conventional approaches, making optimization more robust with reduced requirements for the training dataset. To clarify what we seek in building

a closure model, some essential points in this approach are explained. If a closure model is represented as $\mathcal{F}$ with a solution $\tilde{\boldsymbol{q}}$, then an ideal model should guarantee four criteria, represented as follows

1. Accurate prediction of the solution, $\tilde{\boldsymbol{q}} \mapsto \boldsymbol{u}$.
2. Accurate prediction of the dynamical response, $\mathcal{F} \mapsto \boldsymbol{f}$.
3. Strong numerical stability, $\|\mathcal{F}\| \leq \epsilon$, where ϵ is a bounded value.
4. Representation of the derivatives in lower dimensions, $\nabla.\mathcal{F} \cong \nabla.\boldsymbol{f}$.

The first and second items denote that the dimensionality reduction should not influence the prediction accuracy in the closure model over a finite time interval $\mathbb{T}$. This accuracy can be improved by adding an adequate number of subspaces (i.e., POD modes) to the ROB. In the third item, having a rich training dataset can improve prediction accuracy in the closure model, but does not guarantee stability. The Galerkin projection is represented as a continuous form of minimization in the closure model [29]. This projection becomes highly unstable for non-linear dynamical systems (i.e. in high Reynolds number flows). The reason is that small flow structures play an essential role in the subsequent flow evolution [95]. To overcome this issue, other implicit projection methods, such as Least-Squares Petrov-Galerkin (LSPG), can be employed to build a stable closure model [29, 30, 109]. In the fourth item, derivatives of the closure model should closely approximate those of the FOM in lower dimensions. Most ROMs (linear, auto-encoder or data-driven closure models) are only designed to approximate the state vector over $t \in \mathbb{T}$. In other words, there is no general model that represents the Jacobian of the ROM. Therefore, conventional ROMs may not be suitable for sensitivity analysis. Besides accurate prediction of the state vector, ROMs should project the most prominent features of the Jacobian and related derivates of the FOM (i.e., $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$ and $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}}$) into lower dimensions.

7.2 Training Strategies

The training procedure is significantly dependent on the data collected from the results of the FOM simulations. A snapshot is a vector of the state, or other related

variables, that is obtained at a specific time interval. This snapshot could be multi-dimensional, but it is often converted into a vector (i.e., stack all data in a column). If the dataset is built from entire snapshots at all time intervals, it will often need a large memory allocation. So these types of data are usually built by collecting snapshots, as the samples, from a limited set of time intervals. Therefore, selecting a proper and efficient sampling strategy is an essential matter in building an accurate ROM. The main objective of this study is to employ the ROM for optimization purposes. Hence, this ROM we build should approximate the sensitivity solutions over a finite time for a different set of design parameters.

Figure 28 shows three different data sampling strategies, and their impact on the sensitivity solution approximated via the ROM. In this case, data sampling is provided from a 2D simulation of steady-state flow past a NACA 0012 airfoil at $Re = 1000$, $M_\infty = 0.2$, and $\alpha_{eff} = 0^\circ$. Let us assume that $\mathcal{S}_1$ is in design space, $\mathscr{D}$. We want to build a ROM that approximates the state and sensitivity solutions in a subspace $\mathscr{D}_1 \subset \mathscr{D}$. We start sampling the data when the FOM is not yet steady-state, and we continue it until this FOM becomes completely steady-state. Therefore, the corresponding dataset contains both quasi-steady and steady solutions, and it can be applied to build the ROB. In Strategy 1, the ROM is built based on the state dataset collected at $\mathcal{S}_1$, which is denoted by green colour in design space. In this case, the sensitivity solution reflects a wrong answer compared to the sensitivity solution computed by FOM, which is computed by $\boldsymbol{v} = -(\frac{\partial \nabla . \boldsymbol{f}}{\partial \boldsymbol{u}})^{-1} \frac{\partial \nabla . \boldsymbol{f}}{\partial \mathcal{S}_1}$. This situation usually occurs when the dynamical system is susceptible to design parameters. In Strategy 2, several FOMs for different design parameters, $\mathcal{S} \in \{\mathcal{S}_1, \ldots, \mathcal{S}_{n_{dp}}\} \subset \mathscr{D}_1$, are performed, and afterward, the ROM is built by the state datasets collected at all $\mathcal{S}_1, \ldots, \mathcal{S}_{n_{dp}}$. In this case, the sensitivity solution approximated by the ROM is notably improved, but there is still a large discrepancy between the result of the ROM and that of the FOM. The accuracy in Strategy 2 can be increased by adding more data into the dataset, collected from further FOM simulations at different design parameters in $\mathscr{D}_1$. This strategy can be helpful to approximate those solutions in $\mathscr{D}_1$ using thrust region approaches. However, it may not be economical when the number of FOM simulations increases. As shown in Strategy 3, the ROM is built using both the state and the sensitivity datasets. In this case, the sensitivity function is only solved for the sensitivity solution at $\mathcal{S}_1$, denoted by blue colour. The sensitivity

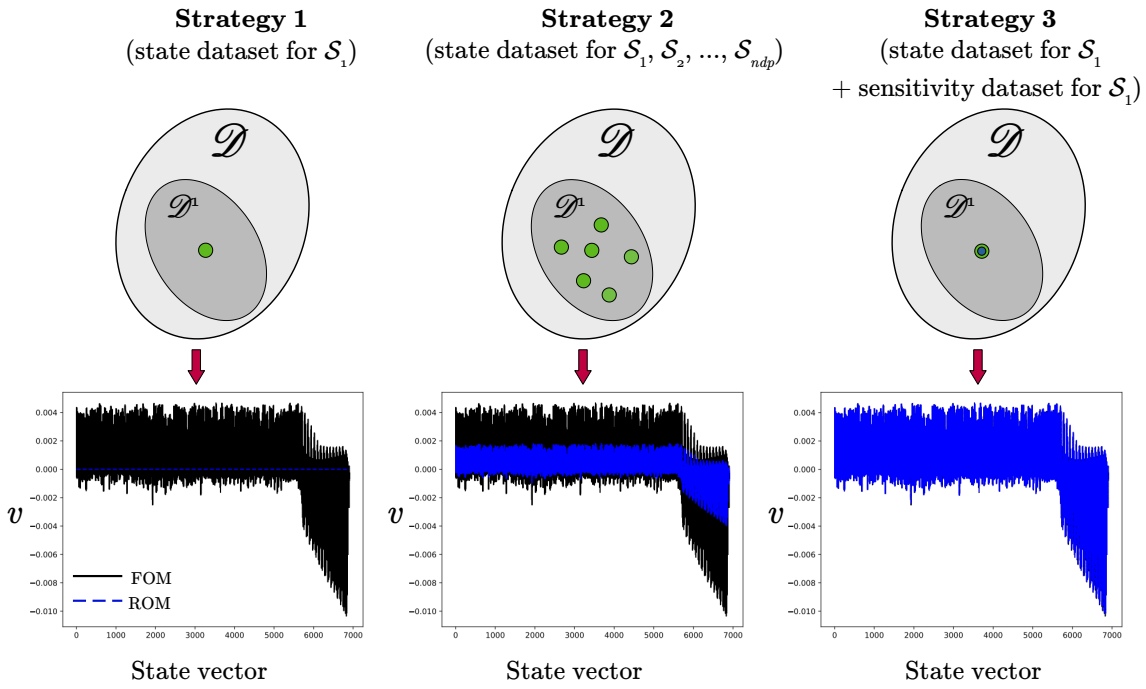


Figure 28: Evaluating different strategies for data sampling in design space.

solution computed using the ROM and the FOM are in good agreement, and they are almost on top of each other. This is because the ROM is trained via extra information, containing the prominent structure of the sensitivity solution. Therefore, the ROB delivers accurate information to the ROM to approximate the sensitivity solution in lower dimensions. Interestingly, Strategy 3 only needs one simulation of the FOM in subspace $\mathscr{D}_1$. This approach indicates that we can not find dominant structures of the sensitivity solution from those in the flow field. This finding becomes an essential matter when the dynamical system becomes highly non-linear.

Figure 29 is also provided to compare Strategies 2 and 3 during the optimization procedure. Let us assume that $\mathcal{S}_1$, as a baseline design, is optimized in design space $\mathscr{D}$. Each design cycle has its subspace, such as $\mathscr{D}_1$ and $\mathscr{D}_2$ for the first and second iterations. In Strategy 2, we select n_{dp} different set of design parameters, $\mathcal{S} \in \{\mathcal{S}_1, \ldots, \mathcal{S}_{n_{dp}}\} \subset \mathscr{D}_i$, where index i indicates the design iteration. Therefore, n_{dp} different FOMs are simulated, followed by building the ROB, and performing sensitivity analysis (SA). According to Figure 28, this approach may need a large number of FOMs if the number of design parameters in $\mathcal{S} = [s_1, s_2, \ldots, s_{n_s}]$ is very large. Additionally, the updated design may not be very exact in each subspace, since

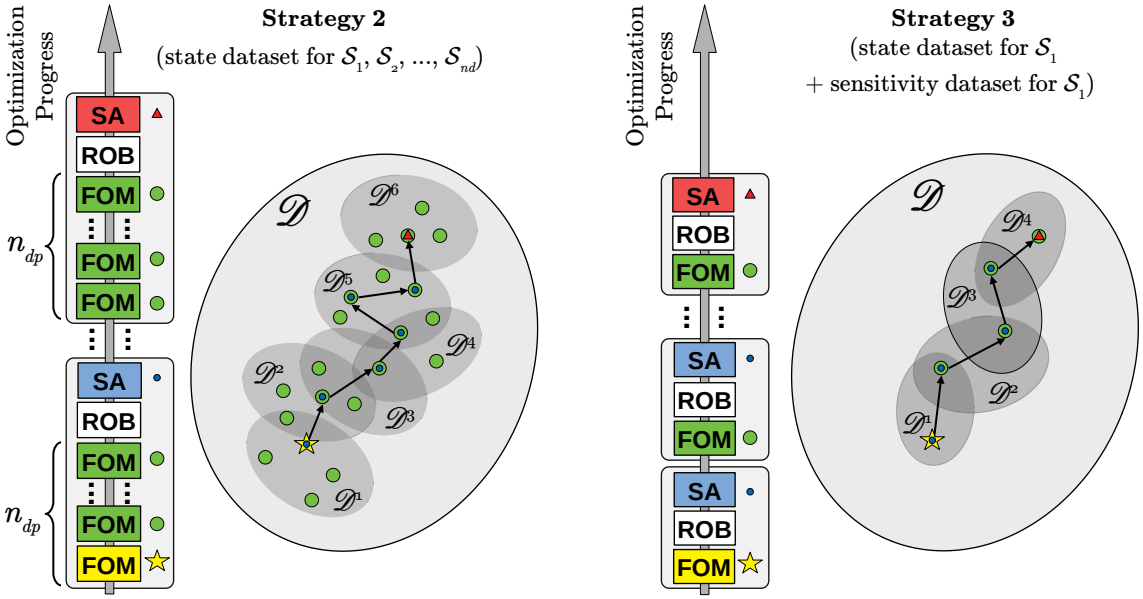


Figure 29: Comparing the performance of Strategies 2 and 3 during the optimization procedure.

the accuracy of the sensitivity solution increases by adding more information to the dataset. On the other hand, Strategy 3 only needs one FOM simulation for each subspace, followed by building the ROB, and performing SA. In optimization problems with many design parameters, this strategy will significantly reduce the computational cost. Although Strategy 3 may reflect a promising improvement in the accuracy of the ROM, this ROM may lose its accuracy for a different set of design parameters, $\{\mathcal{S}_2, \ldots, \mathcal{S}_{n_{dp}}\}$, in the same subspace. As a matter of fact, we do not need to use this ROM for different design parameters, since we only require those sensitivities at $\mathcal{S}_1$. However, in Strategy 2, we learned that adding more information to the dataset will help increase the accuracy of the ROB in each subspace. If we use all of the previous datasets, from $\mathscr{D}_1$ to $\mathscr{D}_{i-1}$, to build the ROB for $\mathscr{D}_i$, we may build a more efficient and effective ROM during the optimization procedure. Consequently, we can combine these two strategies to develop more accurate models. In section 7.3.3, we use these combinations to develop training algorithms. It is worth mentioning that we can change the number of FOMs in each subspace according to the required accuracy in the final design, physics of the problem, and available computational resources.

7.3 Physics-Constrained Data-Driven Approach

7.3.1 Architecture

Solving the weak form of the FOM, which is converted into a closure model, produces non-physical results unless optimal trial and test basis functions are defined. Therefore, we leverage data science techniques to find those optimal basis functions for the closure model. To this end, we propose a novel algorithm to find the overall structure of the manifolds, and shape them in Hilbert space. This proposed algorithm has four major phases. The first phase is the *preparing* phase, and in this phase, a closure model is developed to obtain the dynamics of the system. The second phase is the *building* phase, during which different data sampling methods are taken into account in order to generate a dataset. In the third phase, *hyper-reduction*, the dynamical behaviour of the reduced-order matrices are found by deep learning. This phase reduces the computational costs attributed to building the closure model for the sensitivity function. The fourth phase is the *shaping* phase, which refers to improvement of the ROB in lower dimensions through an iterative process, and solving for sensitivities. Figure 30 illustrates the proposed platform architecture schematically. First, the FOM is solved over $\Delta\mathbb{T}$, and snapshots are collected from it. These snapshots in physical space, $\boldsymbol{u} \in \mathscr{P}$, are projected into Hilbert space, $\mathscr{H}_1$. Next, in the *preparing* phase, Eq. (76) is solved as a closure model via LSPG, such that $\tilde{\boldsymbol{q}} \in \mathscr{H}_1$ lies in a lower-dimensional space. The red nodes contain the dynamics of the system in $\mathscr{H}_1$. In the *building* phase, the ROB are created with sensitivity solutions that are sampled at $\mathbb{N}_p(n) = \{\xi_p^{(1)}, \xi_p^{(2)}, \ldots, \xi_p^{(m_p)}\}$, where $\xi_p^{(n)}$ is index of each time interval (i.e., $\xi_p^{(n)}\Delta t \in \Delta\mathbb{T}$), and m_p denotes the number of samples in the *building* phase. Afterward, the overall structure and shape of manifolds in Hilbert space are obtained. Large-scale Jacobian and relative matrices are transformed from $\mathscr{P}$ into $\mathscr{H}_2$, where the sensitivity solutions evolve with time in this space. The second phase is optional, while it significantly reduces the computational burden for sensitivity analysis of large-scale chaotic problems. If the closure model does not suffer high computational cost, we can skip this phase. In the *hyper-reduction* phase, costly reduced-order matrices are only computed for a limited set of time intervals, $\mathbb{N}_s(n) = \{\xi_s^{(1)}, \xi_s^{(2)}, \ldots, \xi_s^{(m_{\text{samp}})}\}$, where m_{samp} is the number of sample intervals, and the remainder matrices are computed via deep learning. Furthermore, in the *shaping* phase, the trial basis functions

are deformed in all directions in $\mathscr{H}_2$, and this deformation is optimized with regard to another set of samples at $\mathbb{N}_q(n) = \{\xi_q^{(1)}, \xi_q^{(2)}, \ldots, \xi_q^{(m_q)}\}$, where m_q is the number of samples in the *shaping* phase. Additionally, LSS minimization is solved for sensitivities in this phase. Afterward, the orange nodes contain the sensitivity solutions of the corresponding dynamics in Hilbert space. At the end of the procedure, the solutions are lifted back from $\mathscr{H}_2$ into $\mathscr{P}$ with a proper mapping function. The output of this platform produces the sensitivity solution, $\boldsymbol{v}^{(n)} \in \mathscr{P}$, which reflects the solution of Eq. (103) over $\Delta\mathbb{T}$. In Sections 7.3.2 to 7.3.5, we provide further details to describe the algorithms and related numerical approaches.

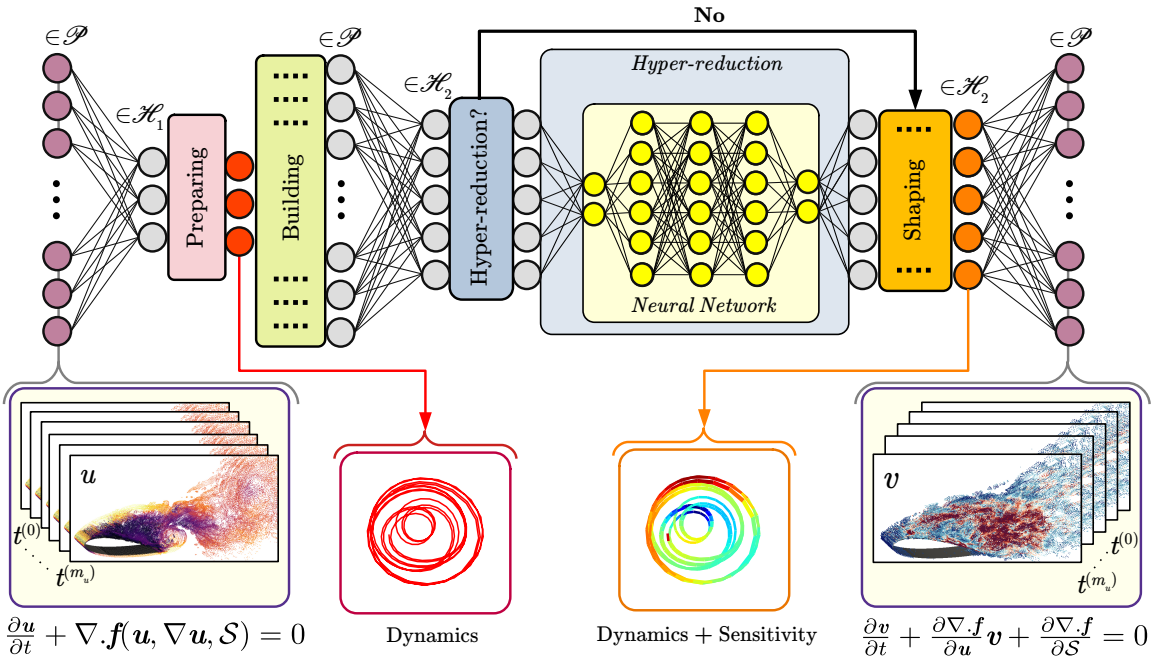


Figure 30: Schematic of the platform designed for sensitivity analysis.

7.3.2 Preparing a Closure Model for the Dynamics of the System

In model-reduction techniques, the trial basis functions are built by a set of ROB matrices. Generally speaking, a ROB is a subset of trial basis functions, containing prominent structures of the dynamical system. Each ROB is generated by different

datasets (e.g., state vector, sensitivity solution) that are collected over $\Delta\mathbb{T}$. Additionally, other quantities can help enrich the information we require to shape the manifolds in $\mathscr{H}$. These quantities can be the residuals, fluxes, or Krylov vectors [148]. High-dimensional datasets can be collected for different sets of design parameters, $\mathcal{S} \in \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_{dp}}\} \subset \mathcal{D}$, where n_{dp} denotes the total number of design sets (i.e., sample design parameters). Therefore, snapshots for each design set are assembled as follows

$$\mathbf{U}(\mathcal{S}_i) = \begin{bmatrix} | & | & & | \\ \Delta\boldsymbol{u}^{(1)}(\mathcal{S}_i) & \Delta\boldsymbol{u}^{(2)}(\mathcal{S}_i) & ... & \Delta\boldsymbol{u}^{(m_u)}(\mathcal{S}_i) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n_u \times m_u}, \tag{139}$$

where $\Delta\boldsymbol{u}^{(n)} = \boldsymbol{u}^{(n)} - \overline{\boldsymbol{u}}$, and $i = 1, 2, ..., n_{dp}$. We build all datasets according to Eq. (139), and then assemble all of them into a united matrix

$$\mathbf{X}_{state} = \bigoplus_{i=1}^{n_{dp}} \mathbf{U}(\mathcal{S}_i), \tag{140}$$

where, $\mathbf{X}_{state}$ denotes the dataset used to train the ROB. Subsequently, we build the ROB using

$$\tilde{\mathbf{\Phi}}_u = \mathbf{POD}(\mathbf{X}_{state}), \quad \text{and} \quad \mathbf{U}(\mathcal{S}_i) = \tilde{\mathbf{\Phi}}_u \mathcal{Q}_u(\mathcal{S}_i), \quad i = 1, 2, ..., n_{dp}, \tag{141}$$

where, the **POD** represents a function that returns the orthogonal basis (i.e., POD modes). This function is described in Algorithm 2. Additionally, $\mathcal{Q}_u(\mathcal{S}_i)$ denotes the generalized coordinates for each set of design parameters

$$\mathcal{Q}_u(\mathcal{S}_i) = \begin{bmatrix} | & | & & | \\ \tilde{\boldsymbol{q}}_i^{(1)} & \tilde{\boldsymbol{q}}_i^{(2)} & ... & \tilde{\boldsymbol{q}}_i^{(m_u)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{r_{state} \times m_u}, \tag{142}$$

where r_{state} is the rank of $\tilde{\mathbf{\Phi}}_u$. Note that $\tilde{\boldsymbol{q}}_i^{(n)} \in \mathbb{R}^{r_{state}}$ is actually the generalized coordinates of the weak form of the FOM in Hilbert space. In order to find these coordinates for the closure model, Eq. (76) sould be solved via LSPG to recompute $\tilde{\boldsymbol{q}}_i^{(n)}$ over $\Delta\mathbb{T}$, such that $\tilde{\boldsymbol{q}} \in \mathscr{H}_1$.

7.3.3 Building Manifolds for the Sensitivity Function

In order to improve the robustness of the trial basis function with respect to perturbations in entries, the steady-state forward sensitivity function is solved for a different

set of design parameters. At first, snapshots of the steady-state sensitivity solutions for all sets of design parameters are defined as

$$\mathbf{V}_{ss}(\mathcal{S}_i) = \begin{bmatrix} | & | & & | \\ \Delta\boldsymbol{v}_{ss}^{(\mathbb{N}_p(1))}(\mathcal{S}_i) & \Delta\boldsymbol{v}_{ss}^{(\mathbb{N}_p(2))}(\mathcal{S}_i) & \dots & \Delta\boldsymbol{v}_{ss}^{(\mathbb{N}_p(m_p))}(\mathcal{S}_i) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n_u \times m_p}, \tag{143}$$

where $\Delta\boldsymbol{v}_{ss}^{(n)} = \boldsymbol{v}_{ss}^{(n)} - \overline{\boldsymbol{v}_{ss}}$, and $\overline{\boldsymbol{v}_{ss}} \in \mathbb{R}^{n_u}$ represents a reference vector for sensitivity snapshots, and it usually satisfies $\frac{\partial \overline{\boldsymbol{v}_{ss}}}{\partial t} = 0$. Therefore, the dataset for all sensitivity solutions is represented as

$$\mathbf{X}_{sens} = \bigoplus_{i=1}^{n_{dp}} \mathbf{V}_{ss}(\mathcal{S}_i), \tag{144}$$

and then,

$$\tilde{\mathbf{\Phi}}_v = \mathbf{POD}(\mathbf{X}_{sens}) \quad \text{and} \quad \mathbf{V}_{ss}(\mathcal{S}_i) = \tilde{\mathbf{\Phi}}_v \mathcal{Q}_v(\mathcal{S}_i), \quad i = 1, 2, ..., n_{dp}. \tag{145}$$

It is worth pointing out that the ROB obtained by the sensitivity dataset could have a different number of subspaces, $\tilde{\mathbf{\Phi}}_v \in \mathbb{R}^{n_u \times r_{sens}}$, where r_{state} is the rank of the ROB that is built by the sensitivity data. Accordingly, the generalized coordinates for the sensitivity solutions can be obtained by

$$\mathcal{Q}_v = \begin{bmatrix} | & | & & | \\ \tilde{\boldsymbol{h}}_i^{(\mathbb{N}_p(1))} & \tilde{\boldsymbol{h}}_i^{(\mathbb{N}_p(2))} & \dots & \tilde{\boldsymbol{h}}_i^{(\mathbb{N}_p(m_p))} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{r_{sens} \times m_p}, \quad \mathbb{N}_p(n) = \{\xi_p^{(1)}, \xi_p^{(2)}, \dots, \xi_p^{(m_p)}\}. \tag{146}$$

Finally, we need to collect data for the unsteady sensitivity solutions to ensure that errors in the ROB for chaotic problems do not grow significantly. However, as mentioned earlier, it is impossible to solve the sensitivity function of chaotic systems using conventional methods. On the other hand, if the dynamical system is chaotic, the steady-state sensitivity datasets are not sufficient to build an accurate closure model. Therefore, we propose a new approach for data collection from the unsteady sensitivity function. The intuition behind this idea is that the solution of the steady-state sensitivity function only gives an acceptable approximation when the system is linear or weakly non-linear. On the other hand, detecting features of the unsteady sensitivity function, and adding them to the trial basis function improves the prediction of the closure model for strongly non-linear systems. Therefore, the unsteady

sensitivity solutions are collected according to a procedure provided in Algorithms 4 and 5. Let us assume that $\mathcal{S}_1$ is the baseline design parameter that reflects the properties of the current problem, and the rest of design parameters $\{\mathcal{S}_2, \ldots, \mathcal{S}_{n_{dp}}\}$ are sample designs for evaluation. Algorithms 4 and 5 are only applied to $\mathcal{S}_1$ since this part of data collection is computationally expensive. Keep in mind that Algorithm 4 sets a proper initial condition to proceed with data sampling using Algorithm 5. Therefore, after data sampling using the aforementioned algorithms, the data are collected and represented as

$$\mathbf{V}_{us}(\mathcal{S}_1) = \begin{bmatrix} | & | & & | \\ \Delta\boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(1))}(\mathcal{S}_1) & \Delta\boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(2))}(\mathcal{S}_1) & \ldots & \Delta\boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(\acute{m}_p))}(\mathcal{S}_1) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n_u \times \acute{m}_p}, \quad (147)$$

where $\Delta\boldsymbol{v}_{us}^{(n)} = \boldsymbol{v}_{us}^{(n)} - \overline{\boldsymbol{v}_{us}}$, and $\mathbf{V}_{us} \in \mathbb{R}^{n_u \times \acute{m}_p}$ represents the dataset for prototype of the unsteady sensitivity solutions, $\acute{m}_p \in \mathbb{R}_+$ is the total number of samples. Moreover, $\boldsymbol{v}_{us}^{(n)} \in \mathbb{R}^{n_u}$ and $\overline{\boldsymbol{v}_{us}} \in \mathbb{R}^{n_u}$ correspond to the discrete unsteady sensitivity solution and its corresponding reference vector, respectively. Since we only consider the dataset for $\mathcal{S}_1$ in the unsteady case, then $\acute{\mathbf{X}}_{sens} = \mathbf{V}_{us}(\mathcal{S}_1)$, where $\acute{\mathbf{X}}_{sens} \in \mathbb{R}^{n_u \times \acute{m}_p}$ represents the dataset of the unsteady sensitivity solutions. Therefore, the ROB for the unsteady sensitivity solutions is given by

$$\acute{\tilde{\mathbf{\Phi}}}_v = \mathbf{POD}(\acute{\mathbf{X}}_{sens}) \quad \text{and} \quad \mathbf{V}_{us}(\mathcal{S}_1) = \acute{\tilde{\mathbf{\Phi}}}_v \acute{\mathcal{Q}}_v(\mathcal{S}_1), \quad (148)$$

and the generalized coordinates for Eq. (148) are

$$\acute{\mathcal{Q}}_v = \begin{bmatrix} | & | & & | \\ \acute{\tilde{\boldsymbol{h}}}_1^{(\mathbb{N}_{\acute{p}}(1))} & \acute{\tilde{\boldsymbol{h}}}_1^{(\mathbb{N}_{\acute{p}}(2))} & \ldots & \acute{\tilde{\boldsymbol{h}}}_1^{(\mathbb{N}_{\acute{p}}(\acute{m}_p))} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{\acute{r}_{sens} \times \acute{m}_p}, \quad \mathbb{N}_{\acute{p}}(n) = \{\xi_{\acute{p}}^{(1)}, \xi_{\acute{p}}^{(2)}, \ldots, \xi_{\acute{p}}^{(\acute{m}_p)}\}, \quad (149)$$

where $\acute{r}_{sens}$ is the rank of $\acute{\tilde{\mathbf{\Phi}}}_v$. The relative scale of the dataset directly influences the effectiveness of the **POD** function. Therefore, the **POD** function can not find the most effective trial basis function if we use $\mathbf{POD}([\mathbf{X}_{state}, \mathbf{X}_{sens}, \acute{\mathbf{X}}_{sens}])$ [32, 64]. Consequently, according to [148], a Gram-Schmidt-like procedure is suggested, such that the ROB are embedded as a subset of trial basis function

$$\tilde{\mathbf{\Phi}} = \tilde{\mathbf{\Phi}}_u \oplus \tilde{\mathbf{\Phi}}_v \oplus \acute{\tilde{\mathbf{\Phi}}}_v, \quad \text{and} \quad \tilde{\mathbf{\Phi}} \in \mathbb{R}^{n_u \times r}, \quad (150)$$

where, $r = r_{state} + r_{sens} + \acute{r}_{sens}$. The procedure for building the trial basis functions from a different set of ROB is explained in Algorithm 6. It is worth mentioning that the LSPG approach, with which the solutions of the ROM are obtained in a minimization framework at each time interval, has the minimum-residual property. In other words, $\tilde{\mathbf{\Phi}}_u$ would not influence the results of the sensitivity functions significantly. Furthermore, the POD modes embedded in $\tilde{\mathbf{\Phi}}$ can be rearranged based on their priority. Therefore, the OΔEs in the closure model are solved based on the priority of prominent features in the dynamical system.

7.3.4 Hyper-Reduction

Developing a closure model for a large-scale chaotic FOM is often computationally expensive. Dimensionality reduction of the Jacobian and relative matrices needs many scalable computational resources. Although the Jacobian of the dynamical system is represented as a sparse matrix, the trial and test basis functions of this dynamical system are dense matrices and, hence, their mathematical operations are computationally expensive. Therefore, building those terms of the closure model, given by Eq. (136), is still limited to available computational resources. These terms are $\tilde{\mathbf{\Psi}}^{(n)\top}\tilde{\mathbf{\Psi}}^{(n)}$, $\tilde{\mathbf{\Psi}}^{(n)\top}\tilde{\mathbf{\Phi}}$, $\tilde{\mathbf{\Psi}}^{(n)\top}\frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}}$, and $\tilde{\mathbf{\Psi}}^{(n)\top}\Big(\sum_{j=1}^{k}\zeta_j\mathcal{I}+\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\Big)$, which are required to solve Eq. (136) for the sensitivity solutions in Hilbert space. The remainder of this section shows how to reduce this computational burden using another dimensionality reduction, which is referred to as hyper-reduction.

Deep learning effectively captures non-linear patterns, and characteristics in a dataset. We construct neural network surrogate models to predict patterns in the aforementioned terms. To this end, we use Multi-Layer Perceptron (MLP) architecture to develop a surrogate model using a neural network. First, we compute the aforementioned terms in Eq. (136) over a limited set of time intervals, as the sample dataset. In the next step, using dimensionality reduction techniques, the generalized coordinates of this dataset are obtained. Later, we use these coordinates to train the network. The training objective function, RMSE, is minimized with respect to the sample dataset

$$\text{RMSE} = \sqrt{\frac{1}{m_{\text{samp}}}\sum_{n=1}^{m_{\text{samp}}}\left(q_{\text{pred}}^{(\mathbb{N}_s(n))} - q_{\text{samp}}^{(\mathbb{N}_s(n))}\right)^2}, \tag{151}$$

where, q_{samp}, and q_{pred} are scalar generalized coordinates for sample and prediction,

respectively. Since the objective is an accurate prediction of the generalized coordinates over $\Delta\mathbb{T}$, we train the surrogate model in $\mathbb{N}_s(n) = \{\xi_s^{(1)}, \xi_s^{(2)}, \ldots, \xi_s^{(m_{\text{samp}})}\}$. Consequently, developing surrogate models via deep learning allows us to predict the aforementioned terms in the remainder of time intervals over $\Delta\mathbb{T}$ with a minimal computational cost.

7.3.5 Shaping the Trial Subspaces

As the FOM becomes non-linear, the mapping functions will be highly sensitive to projection errors. This issue influences the accuracy and robustness of ROMs. Although closure models developed by LSPG are unconditionally stable over $\Delta\mathbb{T}$, the accuracy of the sensitivity solutions are not guaranteed. Most errors in the sensitivity solutions come from the trial basis function, which causes the sensitivity solution diverges from the exact one, i.e., $\forall t^{(n)} \in \Delta\mathbb{T}\colon \boldsymbol{v}^{(n)} \neq \overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}}^{(n)}$. Additionally, a combination of different ROB may deviate the manifolds in $\mathscr{H}$. However, we should note that this is not the usual case.

In the *shaping* step, we perform LSS to compute sensitivities, and if those residuals in LSS are large, we minimize the ℓ^2-norm of them after embedding the updated sensitivity solution, $\tilde{\boldsymbol{h}}_c := \tilde{\boldsymbol{h}}(\boldsymbol{\Phi}_v\boldsymbol{\Theta}_v)$, where $\boldsymbol{\Theta}_v \in \mathbb{R}^{r\times r}$ is a linear operator that reshapes the manifolds for all sets of subspaces. In this study, we set $\boldsymbol{\Theta}_v = \theta_{corr}\mathcal{I}$ to make it a scaling matrix at which $\theta_{corr} \in \mathbb{R}$ is a factor to scale the manifolds in each subspace. The updated manifold results in changes to the mapping function, such that $\boldsymbol{\Phi}\boldsymbol{\Theta}_v\tilde{\boldsymbol{h}}_c^{(n)} \mapsto \boldsymbol{v}^{(n)}$, which indicates that changes in the updated manifolds will influence the sensitivity solution in lower dimensions. Therefore, this numerical procedure can be defined as a minimization problem

$$\tilde{\boldsymbol{h}}_c, \boldsymbol{\Theta}_v \in \underset{\substack{\boldsymbol{z}\in\mathbb{R}^{n_u},\\ \mathbf{Y}\in\mathbb{R}^{r\times r}}}{\operatorname{argmin}} \quad \sum_{n=1}^{m_q} \left\| \frac{\partial\big(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}_u\tilde{\boldsymbol{q}}^{(\mathbb{N}_q(n))}\big)}{\partial\mathcal{S}} - \tilde{\boldsymbol{\Phi}}\mathbf{Y}\boldsymbol{z}^{(\mathbb{N}_q(n))} \right\|_2^2, \tag{152}$$

where $\mathbb{N}_q(n) = \{\xi_q^{(1)}, \xi_q^{(2)}, \ldots, \xi_q^{(m_q)}\}$ are a set of random numbers, such that $\xi^{(n)}\Delta t \in \Delta\mathbb{T}$, and m_q shows the number of samples chosen randomly for this minimization. We note that $\boldsymbol{z}$ is updated using LSS after each minimization iteration in Eq. (152) to ensure the sensitivity solution remains optimal in $\mathscr{H}$. As mentioned earlier, Eq. (152) can be considered to ensure the updated sensitivity results are not far from the actual values. The samples we use in Eq. (152) could be the steady-state sensitivity

solutions. We acknowledge that the steady-state sensitivity solutions are not exact for non-linear dynamical systems. However, it gives us useful insights into what the unsteady sensitivity solutions would be, and having these insights may improve the final sensitivity results. Moreover, the *shaping* phase helps fix $\tilde{\mathbf{\Phi}}_v$ if it is contaminated with inaccurate results. Nevertheless, we consider Eq. (152) as an optional step for sensitivity analysis and optimization.

Chapter 8

Computational Platform

8.1 Overview

In Chapters 4, 5, 6, and 7, we discussed the fundamentals required to proceed with convex or non-convex optimization in the presence of the chaotic dynamical system. However, in this chapter, we focus on how to solve the ROM-constrained optimization for large-scale problems. Figure 31 displays this procedure for computing sensitivities

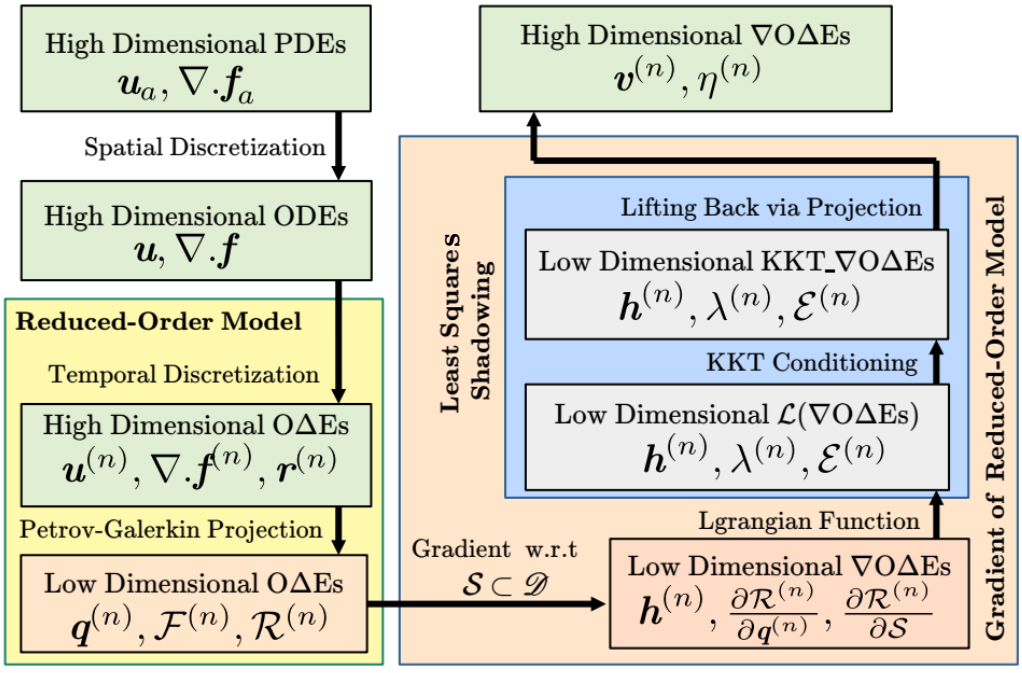


Figure 31: Flow-chart describing the proposed platform for sensitivity analysis.

of the objective function with respect to all design parameters, $\mathcal{S}$. At first, the high-dimensional Navier-Stokes equations, as a set of PDEs, $\nabla.\boldsymbol{f}_a$, are spatially discretized using the FR approach, and then converted into a set of ODEs, $\nabla.\boldsymbol{f}$. To develop a closure model using *projection-based* approaches, we discretize these high-dimensional ODEs in the time domain, referred to as OΔEs, using an implicit temporal scheme, such as BDF scheme. Next, we project OΔEs into lower-dimensional spaces, $\mathscr{H}$, using the Petrov-Galerkin projection. This projection yields a lower-dimensional set of OΔEs that can be used as a reference to derive the sensitivity function. Derivatives of this closure model with respect to $\mathcal{S}$ are referred to as ∇OΔEs, which is the main challenge in the present study. This is because the Jacobian matrices should be significantly reduced in size, such that they provide sufficient information regarding the dynamics of the corresponding FOM in $\mathscr{H}$. Finally, applying the Lagrange function, $\mathcal{L}(\nabla\text{O}\Delta\text{Es})$, accompanied by Karush-Kuhn-Tucker (KKT) conditioning leads to KKT-∇OΔEs, which is the final set of equations that reflects the sensitivity solution in $\mathscr{H}$. Finally, these sensitivities from $\mathscr{H}$ are lifted back into physical space, $\mathscr{P}$, resulting in the solution of ∇OΔEs in high-dimensional space.

8.2 Discrete Forward Minimization Problem

In ROM-constrained optimization, we define different categories due to having several inner minimization problems. The first category determines the rank of the ROB, which impacts the computational cost, accuracy, stability of the ROM, and consistency of the sensitivity solutions. Therefore, in the optimization procedure, these minimizations are given by

$$r_{state} \in \underset{z\in\mathbb{R}_+\leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=0}^{m_u} \boldsymbol{u}^{(n)} - \overline{\boldsymbol{u}} - \tilde{\boldsymbol{\Phi}}_u \tilde{\boldsymbol{q}}^{(n)} \right\|_2^2 \right), \tag{153}$$

$$r_{sens} \in \underset{z\in\mathbb{R}_+\leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=1}^{m_p} \boldsymbol{v}_{ss}^{(\mathbb{N}_p(n))} - \overline{\boldsymbol{v}_{ss}} - \tilde{\boldsymbol{\Phi}}_v \tilde{\boldsymbol{h}}^{(\mathbb{N}_p(n))} \right\|_2^2 \right), \tag{154}$$

$$\acute{r}_{sens} \in \underset{z\in\mathbb{R}_+\leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=1}^{\acute{m}_p} \boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(n))} - \overline{\boldsymbol{v}_{us}} - \acute{\tilde{\boldsymbol{\Phi}}}_v \acute{\tilde{\boldsymbol{h}}}^{(\mathbb{N}_{\acute{p}}(n))} \right\|_2^2 \right), \tag{155}$$

where $\mathbb{N}_p(n) = \{\xi_p^{(1)}, \ldots, \xi_p^{(m_p)}\}$ and $\mathbb{N}_{\acute{p}}(n) = \{\xi_{\acute{p}}^{(1)}, \ldots, \xi_{\acute{p}}^{(\acute{m}_p)}\}$ are those set of time intervals at which data are sampled. Each minimization problem contains ℓ^2-norm

plus another term that represents the rank of the ROB. Moreover, β is a weighting factor that affects the truncation. Note that ℓ^2-norm computes errors of dimensionality reduction. As the rank increases, ℓ^2-norm reduces, and at a specific rank, the summation of ℓ^2-norm and βz will become minimum.

Another category that attributes the least-squares minimization to obtain the sensitivity solution in the training procedure is defined as

$$\boldsymbol{v}_{ss}^{(\mathbb{N}_p(n))} \in \underset{\boldsymbol{z}\,\in\,\mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \nabla.\boldsymbol{r}^{(\mathbb{N}_p(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_p(n))}}\boldsymbol{z} + \frac{\partial \nabla.\boldsymbol{r}^{(\mathbb{N}_p(n))}}{\partial \mathcal{S}} \right\|_2^2, \tag{156}$$

$$\boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(n))} \in \underset{\boldsymbol{z}\,\in\,\mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \boldsymbol{r}^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_{\acute{p}}(n))}}\boldsymbol{z} + \sum_{j=1}^{k} \frac{\partial \boldsymbol{r}^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_{\acute{p}}(n)-j)}}\boldsymbol{v}_{us}^{(\mathbb{N}_{\acute{p}}(n)-j)} + \frac{\partial \boldsymbol{r}^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \mathcal{S}_1} \right\|_2^2, \tag{157}$$

$$\tilde{\boldsymbol{h}}_c, \boldsymbol{\Theta}_v \in \underset{\substack{\boldsymbol{z}\,\in\,\mathbb{R}^{n_u},\\ \mathbf{Y}\,\in\,\mathbb{R}^{r\times r}}}{\operatorname{argmin}} \quad \sum_{n=1}^{m_q} \left\| \boldsymbol{v}_{ss}^{(\mathbb{N}_q(n))} - \tilde{\boldsymbol{\Phi}}\mathbf{Y}\boldsymbol{z}^{(\mathbb{N}_q(n))} \right\|_2^2, \tag{158}$$

where $\mathbb{N}_q(n) = \{\xi_q^{(1)}, \ldots, \xi_q^{(m_q)}\}$ is a set of time intervals chosen randomly for the *shaping* phase. We defined this category because of its impact on the accuracy of the solutions collected in the training step. Finally, the fully discrete form of the ROM-constrained optimization is represented as follows

$$\begin{aligned}
&\underset{\boldsymbol{q}\in\mathbb{R}^r, \mathcal{S}\in\mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} \approx \frac{1}{m_u}\sum_{n=0}^{m_u} \mathcal{J}^{(n)}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) \\
&\text{subject to:} \\
&\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\sum_{j=0}^{k}\zeta_j \boldsymbol{q}^{(n-j)} + \Delta t^{(n)}\beta_0 \tilde{\boldsymbol{\Psi}}^{(n)\top}\nabla.\boldsymbol{f}^{(n)}\Big(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\boldsymbol{q}^{(n)}, t^{(n)}, \mathcal{S}\Big) = 0, \\
&\text{Apply Eq. (136),} \\
&\text{Apply Eq. (153), Eq. (154), Eq. (155),} \\
&\text{Apply Eq. (156), Eq. (157), Eq. (158),} \\
&\mathbf{C}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) \leq 0.
\end{aligned} \tag{159}$$

The time-averaged sensitivities can also be found by

$$\begin{aligned}
\frac{\overline{\Delta\mathcal{J}}}{\Delta\mathcal{S}} \approx \frac{1}{m_u}\sum_{n=0}^{m_u}\left[\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^\top \big(\overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}}\boldsymbol{\Theta}_v\tilde{\boldsymbol{h}}_c^{(n)}\big) + \frac{1}{m_u}\sum_{n=0}^{m_u}\mathcal{E}^{(n)}\big(\mathcal{J}^{(n)} - \overline{\mathcal{J}}\big) \\
+ \frac{1}{m_u}\sum_{n=0}^{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \mathcal{S}}.
\end{aligned} \tag{160}$$

8.3 Platform Steps

In this section, all parts of the proposed platform are explained briefly to clarify the essential steps taken in the ROM-constrained optimization.

8.3.1 Input

Let us suppose that we want to consider sensitivity analysis for any specific set of design parameters, such as boundary conditions or geometry of a solid object in the computational domain. In this case, these parameters are given manually to the platform. However, in the optimization procedure, the optimizer delivers these parameters automatically to the platform. The input parameters are the initial conditions for the FOM, denoted via $\boldsymbol{u}(0, \mathcal{S}) = \boldsymbol{u}_0$, and a set of design parameters, $\mathcal{S} = [s_1, s_2, \ldots, s_{n_s}]^\top$.

8.3.2 Simulation

In this step, the primal PDEs, in the form of a FOM simulation, are solved in time until a desire convergence criterion (e.g., we usually solve them until $20t^* \sim 40t^*$). Afterward, the rest of this FOM simulation proceed with data collection. First, the state vector is computed, converted into a snapshot, and stored to hard disk memory for a certain number of time intervals, m_u. The time-averaged objective function, $\overline{\mathcal{J}}$, is also computed, and stored.

8.3.3 POD Function

If the aforementioned FOM simulation requires a significant memory in the hard disk, we can build $\mathbf{X}_{state}$ at the end of the simulation, and then built the POD modes. This attempt lets us compress these massive data, and use them with less computational effort. As a matter of fact, we do not need these massive snapshots that are already stored to the hard disk memory anymore, since we already compressed them using the POD function. Therefore, we can delete them, and reconstruct the solution at each discrete time interval, $t^{(n)} \in \Delta\mathbb{T}$, when it is required to compute derivatives (i.e., Jacobian matrices). Note that r_{state} should be determined via Eq. (153) to ensure all

the information in the system is captured, such that $99\% \sim 99.9\%$ of the total energy of flow can be reconstructed.

8.3.4 Data Sampling

According to Chapter 7, we need to solve Eq. (156) and Eq. (157), and build other datasets to find prominent structures that exist in the solution of the sensitivity function. These structures are a critical part of sensitivity analysis, since they determine the final shape of the corresponding manifolds in $\mathscr{H}$. Strategies for data sampling have a significant effect on the computational cost, and those accuracies in sensitivity analysis. One of the factors that come through the experience is the number of samples that need to be collected over $\Delta\mathbb{T}$. There is an optimum point between the accuracy, and the computational cost investigated before starting any optimization procedure. Additionally, after taking all sampling steps, $\mathbf{X}_{sens}$ and $\acute{\mathbf{X}}_{sens}$ are built to obtain the ROB using the **POD** function. Accordingly, Eq. (154) and Eq. (155) are solved to minimize the number of POD modes needed to describe the accurate sensitivity solutions. Finally, we build the trial basis function, $\tilde{\mathbf{\Phi}}$, using those ROB built for all design parameters.

8.3.5 ROM

We can develop the closure model by applying the Petrov-Galerkin projection to the discretized form of the FOM using $\tilde{\mathbf{\Phi}}$, yielding two sets of OΔEs and ∇OΔEs. The residual form of the governing equations is a significant constraint to be applied to the closure model. Therefore, considering this constraint is referred to as the physics-constrained ROM. The residuals of OΔEs and ∇OΔEs are minimized to ensure the optimality condition is applied to the closure model. Note that the Petrov-Galerkin protection is applied at each discrete time interval to guarantee the minimum-residual property over $\Delta\mathbb{T}$.

8.3.6 Derivatives

After developing the closure model, we compute lower-dimensional Jacobian matrices, and other related derivatives according to Eq. (135). These derivatives are stored to disk with a considerable reduction in memory requirements than the FOM. In the

next step, the KKT system in Eq. (51) is built in the form of LSS problem. This LSS can be set into a loop with Eq. (158) to proceed with the *shaping* phase discussed in Chapter 7. As mentioned before, the *shaping* phase is an optional step in the optimization process. In other words, if there is any intrusive mode (i.e., poor-defined mode due to uncertainties or poor data collection), which may influence the resulting sensitivities significantly, this *shaping* phase reshape manifolds in such a way that the corresponding errors in those sensitivities become minimum. Finally, the sensitivity solutions in $\mathscr{H}$ are lifted back into physical space, $\mathscr{P}$, via $\tilde{\mathbf{\Phi}}$. These results can be given to the optimizer in order to update design parameters in design space, $\mathscr{D}$.

Chapter 9

Sensitivity Analysis

9.1 Overview

In this chapter, different examples are provided to demonstrate sensitivity analysis of large-scale fluid dynamic problems using the proposed approach. The main objective of this chapter is to show how dimensionality reduction accelerates sensitivity analysis of high-fidelity CFD problems. First, we provide the computational setup for each case study, and then the results are interpreted in detail.

9.2 Computational Setup

In this section, three different case studies are selected to validate the proposed approach for sensitivity analysis. The first case is flow past a circular cylinder at $Re = 40 \sim 250$. We chose this case since it consists of steady-state and periodic flow regimes (non-chaotic) at different Reynolds numbers. So we can show that the proposed approach can accurately predict the sensitivity of the non-chaotic dynamical system. The second case is flow past a 2D NACA 0012 airfoil at $Re = 2400$, where the airfoil is at high angles of attack, near stall and post-stall. This problem is chaotic, which makes sensitivity analysis more challenging. The third case study presents sensitivity analysis and uncertainty quantification of flow past a 3D NACA 0012 blade at $Re = 2 \times 10^4$. This problem is inherently chaotic since the flow in the wake is fully turbulent.

FOM simulations are performed using the High-ORder Unstructured Solver (HORUS), an in-house CFD package, for the compressible Navier-stokes equations. The Navier-Stokes equations are discretized, and converted into a set of ODEs using the Flux-Reconstruction (FR) approach [65]. The second-order Backward Differentiation Formula (BDF2) is used as a temporal scheme to obtain the fully-discrete $O\Delta Es$. The residual tolerance is set to 10^{-7} with a maximum of 11 inner iterations for each time step. The $O\Delta Es$ at each time interval are solved using Gauss-Newton GMRES with a maximum of 10^3 iterations. The Additive Schwarz method is used for preconditioning. An in-house scientific software, OPtimization Toolkit for Highly NOn-linear Systems (OPTHiNOS), was developed to solve forward sensitivity functions using the proposed approach. Dimensionality reduction is applied to the high-dimensional solutions using Singular Value Decomposition (SVD). A different set of eigenvalue problems for SVD are solved via SLEPc [59]. Furthermore, PETSc [9, 40] compiled with OpenMPI [41] is employed to parallelize linear solvers. The test basis function is built by the Petrov-Galerkin approach with the discrete optimality condition, achieved by finding proper directions in subspaces to minimize the residual of $O\Delta Es$. Here the optimality condition is guaranteed by the LSPG approach. A series of least-squares minimizations are sorted by a feed-forward auto-encoder that is accompanied by two basic steps (i.e., *building* and *shaping*). Furthermore, sensitivity functions are solved via conventional LSS, such that the KKT system is a large sparse multi-block matrix. In order to reduce computational costs, this KKT system is built using first-order BDF1. More details for each case study will be provided in the following sections.

9.3 Test Case 1: Non-Chaotic Flow Past a Circular Cylinder

In this example, the sensitivity of aerodynamic loads are investigated for the flow past a circular cylinder at different Reynolds numbers $Re = 20 \sim 250$ and a constant $M_\infty = 0.1$, where M_∞ indicates the free-stream Mach number. A 2D computational domain with a multi-block structured mesh is used. Inlet and outlet boundaries are placed at $20D$ and $40D$, respectively, where D is diameter of the cylinder. The time-step is set to $\Delta t = 0.05t^*$, where t^* is a convective time ($t^* = l_c/u_\infty$, where l_c

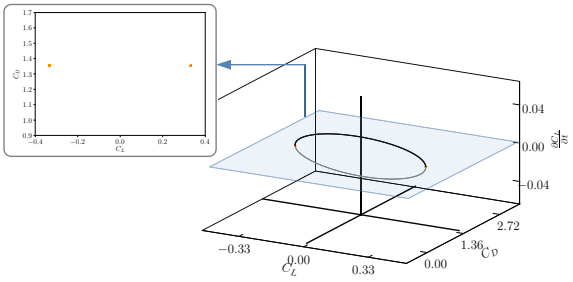
and u_∞ are characteristic length and free-stream velocity, respectively). Note that the characteristic length for this example is set to $l_c = D$. Table 6 compares the results of the present simulation at $Re = 150$ with reference data [28, 66]. Lift and drag (F_L and F_D) are normalized into lift and drag coefficients, $C_L = \frac{F_L}{0.5\rho_\infty l_c u_\infty^2}$ and $C_D = \frac{F_D}{0.5\rho_\infty l_c u_\infty^2}$, where ρ_∞ is free-stream fluid density. Numerical simulations with different solution polynomial degrees, $p_s = \{1, 2, 3\}$, are tested to evaluate accuracy in space. The present results with $p_s = 2$ and $p_s = 3$ are in good agreement with the results of literature [28, 66]. Hence, the rest of this study continues with $p_s = 2$, since the results with this solution polynomial degree agree with those of $p_s = 3$.

p_s	Order of accuracy	DOF	$\overline{C_D}$	ΔC_L	ΔC_D
1	2^{nd}	20,256	1.219	0.331	0.008
2	3^{rd}	45,576	1.328	0.522	0.024
3	4^{th}	81,024	1.331	0.516	0.026
Reference [28]	5^{th}	42,150	1.324	0.516	0.0258
Reference [66]			1.32	0.52	0.026

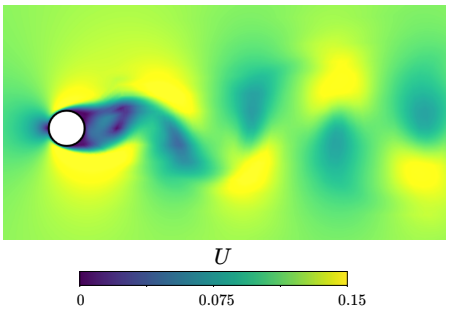
Table 6: Comparison of the current results with forces in other literature.

Figure 32 shows simulation results at $Re = 150$. Figure 32a displays a Poincare map built by projection of $C_L - C_D$ on $\frac{\partial C_L}{\partial t} = 0$. A Poincare map is a technique to detect the evolutional behaviour of dynamical systems. This Poincare map consists of only two points, which indicates periodic behaviour of the dynamical system for this Reynolds number. Figure 32b shows velocity contours, which appear as a von-Karman street in the wake of the cylinder. Since the von-Karman vortex street for this circular cylinder is symmetric, the projected points in the Poincare map are symmetric also. Consequently, we expect that all but one of the LEs for this problem are negative. Figure 33 displays the first five leading LEs, which indicates that theoretical and computed LEs are asymptotic to each other.

For sensitivity analysis, we provide the time-averaged sensitivity of the aerodynamic loads with respect to the Reynolds number. To change the Reynolds number, the free-stream Mach number is kept constant, while the fluid viscosity, μ_u, is changed. Subsequently, the sensitivity of the state vector with respect to the Reynold number, $\frac{\partial \boldsymbol{u}}{\partial Re}$, is computed. For this example, the rank is set to $r_{state} = 32$ to ensure at least %99 of flow energy is captured by $\tilde{\boldsymbol{\Phi}}_u$. Additionally, $\tilde{\boldsymbol{\Phi}}_v$ has the same rank, and we did not use $\acute{\tilde{\boldsymbol{\Phi}}}_v$ since the dynamical system in this example is not complex (i.e.,



(a) Poincare map for a periodic.



(b) Time-averaged aerodynamic loads.

Figure 32: Poincare map and aerodynamic loads for a periodic (non-chaotic) flow past a circular cylinder at $Re = 150$.

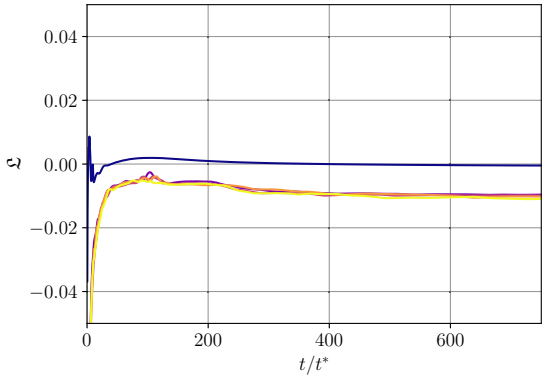


Figure 33: Time-history of the first five leading LEs at $Re = 150$.

$\tilde{\mathbf{\Phi}} = \tilde{\mathbf{\Phi}}_u \oplus \tilde{\mathbf{\Phi}}_v$). The perturbation for the Reynolds number is set to $\epsilon = 1$. In the end, $\overline{C_D}$ and $\overline{C_{L,rms}} = (\frac{1}{T}\int_0^T (\Delta C_L)^2 dt)^{0.5}$ are selected to be objective functions for this example.

Figure 34 shows the time-averaged sensitivity of C_D curve versus Re, which is found by $\overline{\frac{\Delta C_D}{\Delta Re}}$. The second part of Figure 34 presents the time-averaged sensitivity of $C_{L,rms}$ with respect to Re, which is defined by $\overline{\frac{\Delta C_{L,rms}}{\Delta Re}}$. The yellow area in ΔC_D and ΔC_L^2 indicates the range of unsteady fluctuations. As shown, the proposed approach can precisely determine sensitivities for this weakly non-linear dynamical system. It is worth mentioning that strong diffusion in this system, which leads to negative LEs, dampens perturbations to the flow. Therefore, the closure model can exactly map this dynamical system from $\mathscr{P}$ to $\mathscr{H}$, such that $\mathcal{F} \to \mathbb{F}$ for all $t^{(n)} \in \Delta\mathbb{T}$.

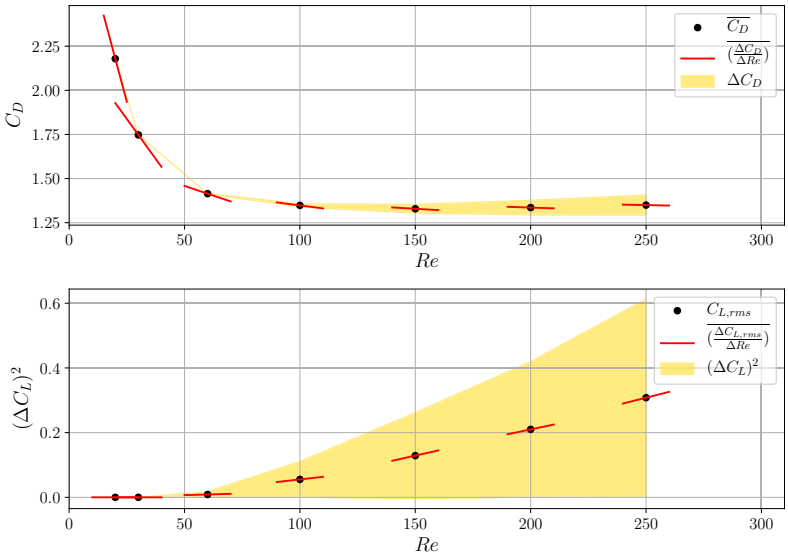


Figure 34: Sensitivity analysis for flow past a circular cylinder.

9.4 Test Case 2: Chaotic Flow Past a 2D NACA 0012 at $Re = 2400$

In the second example, we show how a dynamical system's chaoticity influences the results of the sensitivity function. A 2D computational domain with a NACA 0012 airfoil is selected, and the flow conditions are set, such that $Re = 2400$ and $M_\infty = 0.2$. The computational domain is discretized using unstructured elements, with a clustered boundary mesh around the airfoil. The flow passes around the airfoil until $20t^*$ to ensure the flow field is fully developed in the computational domain. Afterward, data are collected over another $20t^*$. The time-step is set to $\Delta t = 0.01t^*$.

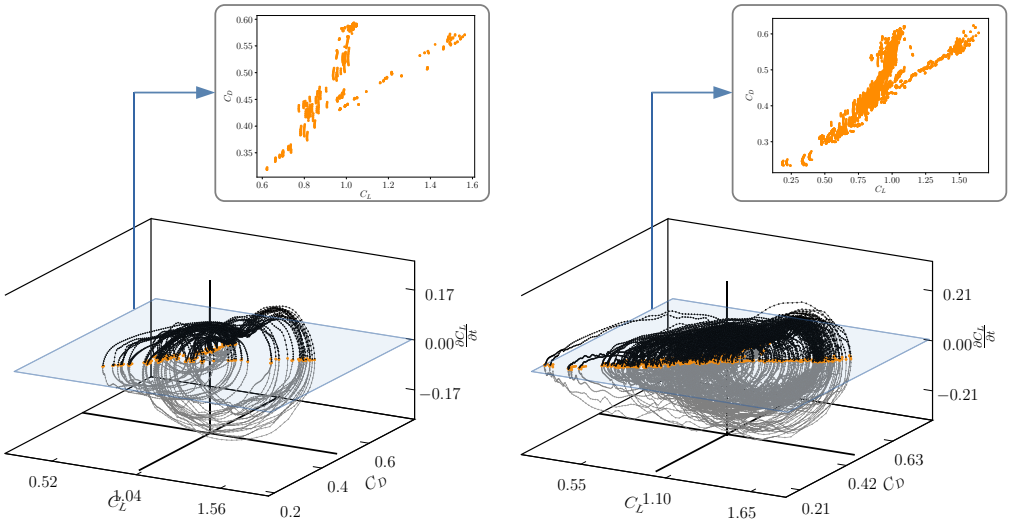
For validation, the effective angle of attack is set to $\alpha_{eff} = 20°$. Table 10 compares the present results with forces available in the literature [45]. The non-dimensional normal and tangential forces are $C_n = \frac{F_n}{0.5\rho_\infty l_c u_\infty^2}$ and $C_t = \frac{F_t}{0.5\rho_\infty l_c u_\infty^2}$, where F_n and F_t correspond to normal and tangential forces, respectively. In this example, the characteristic length is $l_c = c$, where c is chord length of the airfoil. Additionally, the Strouhal number is defined as $St = f_s l_c \cos(\alpha_{eff})/u_\infty$, where f_s is frequency. Please note that $l_c \cos(\alpha_{eff})$ corresponds to the surface normal to u_∞. As shown in Table 10, the current results are in good agreement with the reference solution.

To investigate chaoticity, the flow field should be simulated over a sufficiently

p_s	Order of accuracy	DOF	$\overline{C_n}$	$\overline{C_t}$	$St/\cos(\alpha_{eff})$
2	3^{rd}	134,784	1.138	0.097	0.26
3	4^{th}	239,616	1.158	0.100	0.26
Reference [45]	4^{th}	726,240	1.146	0.102	0.26

Table 7: Comparing the present results at $Re = 2400$ with those in [45].

large $\Delta\mathbb{T}$. Fig. 35 displays Poincare maps for two different solution polynomial degrees, which indicates that the dynamics of the flow are continuously changing. Furthermore, the Poincare map for $p_s = 2$ shows that flow structures are not captured properly compared to those of $p_s = 3$. Therefore, we will consider the flow field simulations with $p_s = 3$ for the rest of this case. In order to compute the leading LEs, the flow field is simulated over $\sim 150t^*$ to ensure that the attractor of the dynamical system is complete. Afterward, the first ten leading LEs are computed, as shown in Fig. 36. There are four positive LEs, $\mathfrak{L}_i > 0 \in n_+$ with $1 \leq i \leq 4$, which are



(a) $p_s = 2$. (b) $p_s = 3$.

Figure 35: Poincare map for chaotic flow at $Re = 2400$ and $\alpha_{eff} = 20°$.

responsible for strong non-linearities in this dynammical system. In this chaotic flow, the first leading LE is $\mathfrak{L}_1 = 0.225$, which governs exponential growth of perturbations. Moreover, $\mathfrak{L}_5 \approx 0$ and all the rest are negative, $\mathfrak{L}_i < 0$ with $6 \leq i \leq 10$, which is in agreement with aforementioned descriptions of chaotic systems.

Figure 37a shows the instantaneous lift and drag coefficients over $325t^*$. These values do not follow any certain cyclic pattern, which indicates the behaviour of

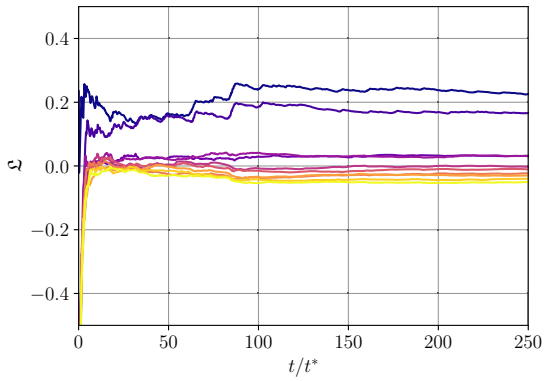
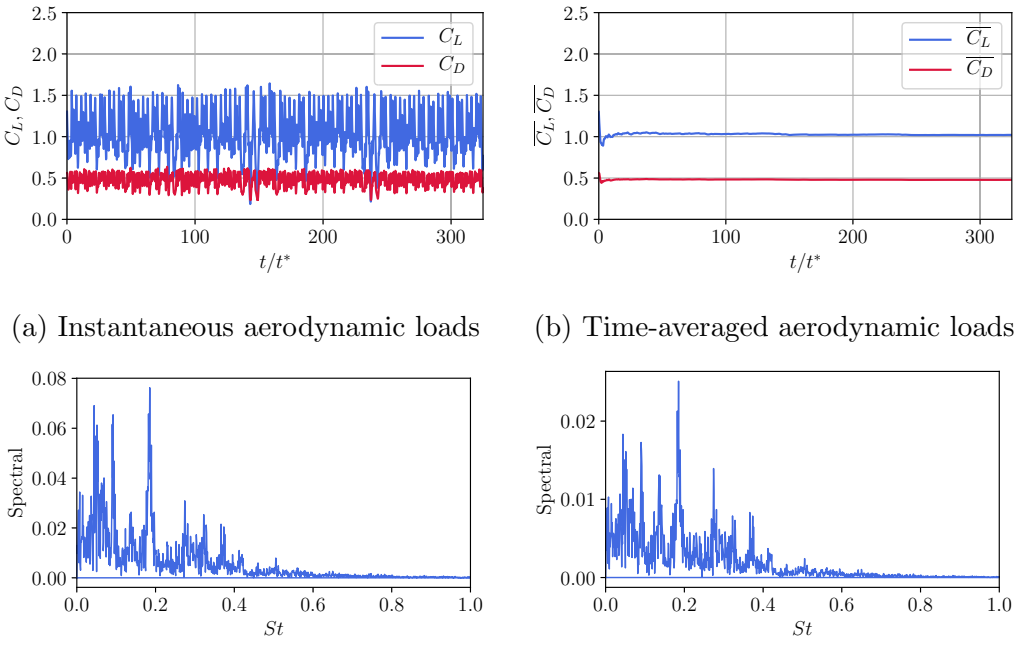


Figure 36: Time-history of the first ten leading LEs at $Re = 2400$ and $\alpha_{eff} = 20°$.

chaotic flows. Fig. 37b also displays the time-averaged lift and drag coefficients over the same time domain. Furthermore, a Fast Fourier Transform (FFT) is applied to the aerodynamic loads to investigate the vortex shedding frequencies. Figure 37c and 37d show the spectral amplitude versus the Strouhal number. In contrast to periodic flows, with a constant shedding frequency, the Strouhal number for this case covers a wide range of frequencies, $St \in [0, 1]$. However, there is one frequency with high spectral amplitude, $St \approx 0.2$, indicating the existence of a dominant vortex shedding pattern. This type of dominant pattern also exist in chaotic flows, also observed by [63, 147].

Sensitivity analysis for this example is done over a range of $2 \times 10^3 \Delta t$ in time domain $\Delta\mathbb{T}$. Statistical analysis is also performed with 99% confidence intervals. The sensitivity function is solved for a wide range of the effective angles of attack, $\alpha_{eff} \in [19°, 25°]$, which covers pre-stall, stall, and post-stall conditions. Figure 38 displays the time-averaged lift and drag coefficients, $\overline{C_L}$ and $\overline{C_D}$, where green error-bars denote 99% confidence intervals. Sensitivity of the lift and drag coefficients with respect to the effective angle of attack, $\frac{\Delta C_L}{\Delta \alpha_{eff}}$ and $\frac{\Delta C_D}{\Delta \alpha_{eff}}$, are shown by red wedges. These results are compared with second-order FD approximations, where the objective functions are computed using FOMs. The sensitivities obtained by the present approach reflect correctly the trend of $\overline{C_L}$ and $\overline{C_D}$ with α_{eff}. The challenging part of sensitivity analysis is usually in stall and post-stall conditions, where flow structures are highly complex due to strong non-linearity of the dynamical system. Interestingly, the computed sensitivities also capture the lift reduction with α_{eff} around the stall angle. The same trend is also reflected by the sensitivities computed for the drag



(a) Instantaneous aerodynamic loads

(b) Time-averaged aerodynamic loads

(c) Spectral versus St for C_L

(d) Spectral versus St for C_D

Figure 37: Aerodynamic loads and Fast Fourier Transform (FFT) results at $Re = 2400$ and $\alpha = 20°$.

coefficient. Due to this instability of the flow in stall and post-stall conditions, the uncertainty of the computed results increases. There are some discrepancies between the FD approximations and the present results at $\alpha_{eff} = 21°$ and $22°$, which is around stall angle. In this condition, computing exact sensitivities remains highly challenging. At the stall angle, the FD approximations themselves may not be suitable for predicting sensitivities. Furthermore, as we will show later in the next example, FD approximations completely fail when strongly non-linear dynamical systems govern vortical structures in the wake. Therefore, we can not judge the results in these two angles with confidence. Furthermore, small sensitivity magnitudes near transitional conditions, such as stall, lead to statistical difficulties when approximating the exact time-averaged values [100].

In conventional sensitivity analysis, the high-dimensional forward sensitivity function at $\alpha_{eff} = 21°$ is directly solved over $10t^*$ time domain. To this end, Eq. (103) is discretized using a BDF2 scheme, and advanced in time with the same time-step already used for FOM simulations. Figure 39 compares the results of the present approach with those of a conventional method. The first column in Figure 39 shows

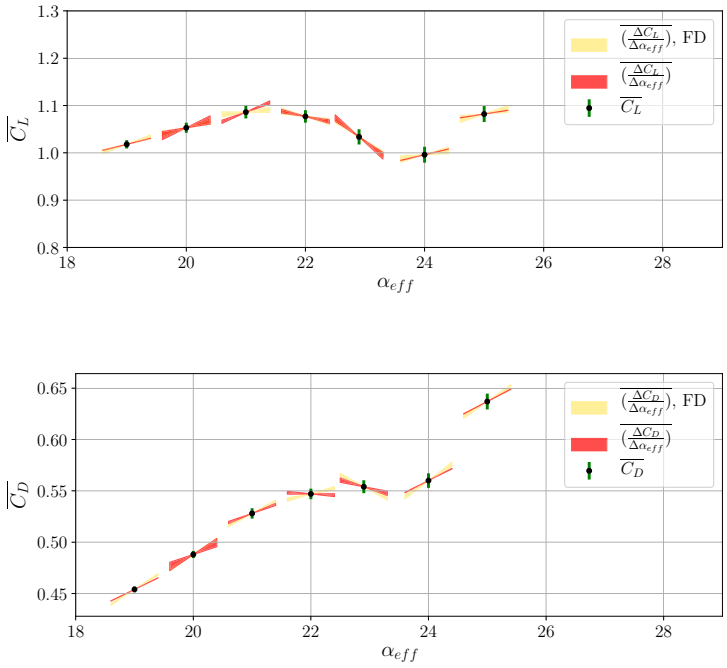


Figure 38: Sensitivity analysis of the aerodynamic loads with respect to α_{eff} near stall and post-stall regions for chaotic flow at $Re = 2400$.

velocity contours at different instants $t = \{2t^*, 6t^*, 10t^*\}$. The second and third columns display contours of the sensitivity solution with respect to the effective angle of attack, $\frac{\partial \boldsymbol{u}}{\partial \alpha_{eff}}$, for both the conventional and present approaches. In the second column, the sensitivity solution blows up at $t = 6t^*$. Moreover, at $t = 10t^*$, errors propagate through the entire domain, particularly in the wake. However, in the third column, the sensitivity solution given by the present approach remains stable over the evaluated time domain. Here, positive eigenvalues in the Jacobian matrix of the linearized FOM correspond to local unstable modes in the dynamical system, and implicit schemes can only preserve instabilities that are caused by these unstable modes. However, positive LEs correspond to global unstable modes in the non-linear dynamical system, making the sensitivity function unconditionally unstable. However, this example demonstrates the potential of the present approach to tackle global unstable modes, and solve for the sensitivity function for strongly non-linear dynamical systems.

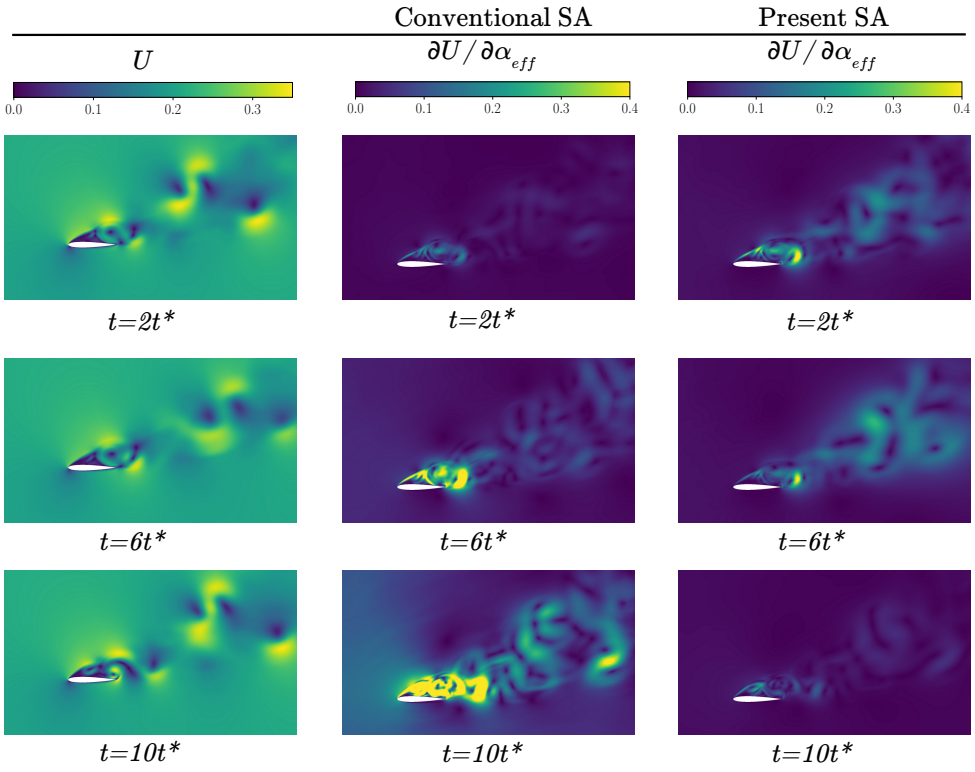


Figure 39: Contours of velocity magnitude, sensitivity solution obtained via a conventional approach, and the sensitivity solution computed using the proposed approach. Note: SA is the abreviation of sensitivity analysis.

9.5 Test Case 3: Chaotic Flow Past a 3D NACA 0020 at $Re = 2 \times 10^4$

In this example, sensitivity analysis of a 3D NACA 0020 airfoil in the presence of massive flow separation at $Re = 2 \times 10^4$ and $M_\infty = 0.1$ is investigated. A C-topology unstructured mesh with hexahedral elements is used. The aspect ratio of the blade is set to $0.45c$ with periodic boundary conditions for the side-walls. The outer boundaries at upstream and downstream are placed $15c$ and $24c$, respectively. The time-step is set to $\Delta t = 0.01t^*$, where t^*. The initial FOM simulation is run to $40t^*$. Then another $7.5t^*$, corresponding to $m_u = 750$, was used for data collection. The rank of the solution in the closure model is set to $r = 200$, and 10% of the time domain is dedicated to sequential sampling from steady-state sensitivity solutions. Additionally, 100% of the time domain, $m_p = m_u$, is considered for sampling from the unsteady sensitivity solutions.

For validation, three different meshes (coarse, medium and fine) are compared with available literature, as shown in Table 8. The ILES [135] and DNS [117] results are time-averaged without including uncertainty bounds, while we provide 99% confidence intervals for the present time-averaged results. In general, the present results for medium and fine meshes are in good agreement with those in the literature. The DNS [117] results are in the range of the 99% confidence intervals for both medium and fine meshes. Since the medium mesh yields suitable results with less computational costs, we chose it for sensitivity analysis.

Test	DOF	$\overline{C_L}$	$\overline{C_D}$
Coarse mesh	6.48×10^5	0.74±0.08	0.40±0.033
Medium mesh	2.09×10^6	0.67±0.06	0.37±0.031
Fine mesh	2.71×10^6	0.65±0.09	0.37±0.038
ILES [135]	-	0.59	0.33
DNS [117]	-	0.64	0.35

Table 8: Comparing the aerodynamic results of a 3D NACA 0020 blade at $Re = 2{\times}10^4$ with those in [117, 135].

To evaluate the chaoticity of this example, the first fifty leading LEs for this blade at $\alpha_{eff} = 20^\circ$ are shown in Figure 40. The initial simulation is performed until $20t^*$ to ensure transitional effects due to the initial condition are omitted from the domain. Afterward, LEs are computed over $10t^*$ with the 99% confidence intervals. As shown, all of these LEs are positive, indicating dominant chaotic behaviour of the flow. Therefore, sensitivity analysis and uncertainty quantification for this case are challenging.

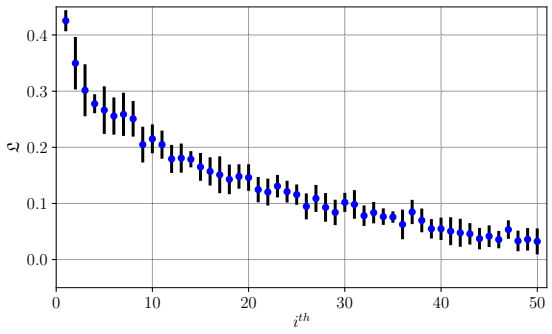
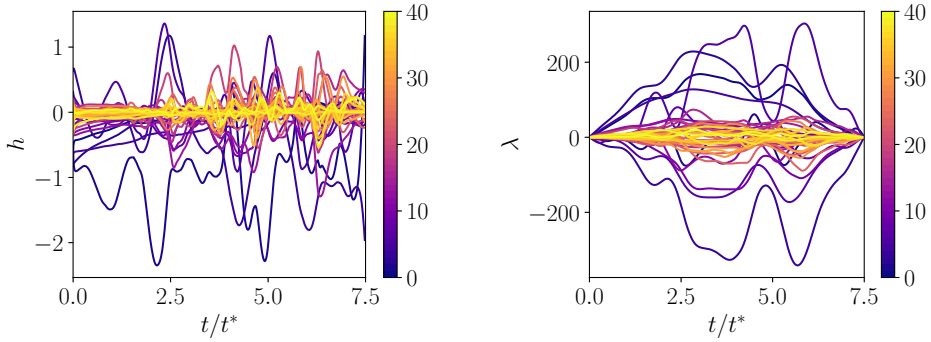


Figure 40: The first fifty leading LEs at $\alpha_{eff} = 20^\circ$.

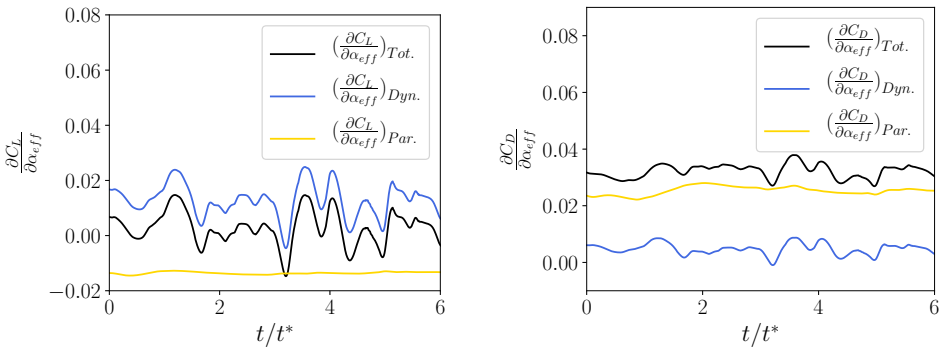


(a) Generalized coordinates. (b) Lagrange multiplier.

Figure 41: Instantaneous results of the sensitivity solutions obtained by the present approach at $\alpha_{eff} = 20°$.

Figure 41 shows instantaneous results obtained by the LSS for the first forty modes. Figure 41a shows the sensitivity of the generalized coordinates. The colour bar shows the rank of each mode associated with its generalized coordinate. Lower rank solutions have more impact on the fluid physics while, as the rank increases, the sensitivity magnitude decreases. This shows that lower rank modes are influenced more by perturbations in the effective angle of attack. This is because these lower rank modes contain the most energy and, hence, are the basis of the dynamical system. Although higher rank modes, containing the small vortical structures and eddies, could be more sensitive to perturbations, their natural effect on the dynamical system remains small. Figure 41b displays the instantaneous Lagrange multipliers. Since LSS solves the sensitivity function by converting it into a boundary-value problem, the Lagrange multiplier at the initial and final time intervals is set to zero. The Lagrange multiplier is notably high for the lower rank modes, indicating a strong relation with the sensitivity solution.

Figure 42 shows the instantaneous sensitivities of aerodynamic loads with respect to the effective angle of attack $\alpha_{eff} = 20°$. The dynamic sensitivity solutions, $\left(\frac{\partial C_L}{\partial \alpha_{eff}}\right)_{Dyn.}$ and $\left(\frac{\partial C_D}{\partial \alpha_{eff}}\right)_{Dyn.}$, are obtained by the first two terms in Eq. (138). Moreover, $\left(\frac{\partial C_L}{\partial \alpha_{eff}}\right)_{Par.}$ and $\left(\frac{\partial C_D}{\partial \alpha_{eff}}\right)_{Par.}$ are the parametric sensitivity solutions, which are given by the last term in Eq. (138). Finally, a summation of these dynamic and parametric sensitivity solutions yields the total sensitivity solutions, $\left(\frac{\partial C_L}{\partial \alpha_{eff}}\right)_{Tot.}$ and $\left(\frac{\partial C_D}{\partial \alpha_{eff}}\right)_{Tot.}$. It is noteworthy that the instantaneous results do not exhibit any periodic behaviour, which is a sign of chaotic dynamical systems.



(a) Sensitivity of lift coefficient (b) Sensitivity of drag coefficient

Figure 42: Instantaneous sensitivity of the aerodynamic loads with respect to a perturbation in the effective angle of attack at $\alpha_{eff} = 20°$.

Figure 43 is also provided to interpret the time-averaged sensitivity of the aerodynamic loads at different effective angles of attack, $\alpha_{eff} = \{5°, 10°, 15°, 20°\}$. The time-averaged lift and drag coefficients are shown with green errorbars, indicating 99% confidence intervals. Moreover, red wedges display 99% confidence intervals for the present results. Furthermore, yellow wedges belong to the second-order FD approximations with 99% confidence intervals. The present sensitivities are in good agreement with the results computed by FOMs. Additionally, the present results reflect proper trends for the lift and drag curves with α_{eff}. On the other hand, the uncertainty level of the FD approximations increases significantly as the effective angle of attack increases. This uncertainty becomes dominant after $\alpha_{eff} = 10°$, and the FD approximations completely fail to provide any practical sensitivity. These notable augmentations in the objective function with high uncertainties are the primary challenge in non-convex optimizations. The present approach could be a good replacement for these types of optimizations.

On the left-hand side of Figure 44, Q-criterion is obtained by the state vector. Also, on the right-hand side, Q-criterion is given by the sensitivity solution vector at $\alpha_{eff} = 20°$. At an instant $t = t^*$, the Leading Edge Vortex (LEV) is fully developed in size, and is about to pinch-off. At this instant, it is observed that the sensitivity solution is not very high. At instant $t = 4t^*$, the Trailing Edge Vortex (TEV) rolls up, and grows in strength. Therefore, the flow in the wake becomes highly sensitive to the perturbations in the effective angle of attack.

Overall, each Jacobian matrix for this example allocates about 7 GB of hard disk

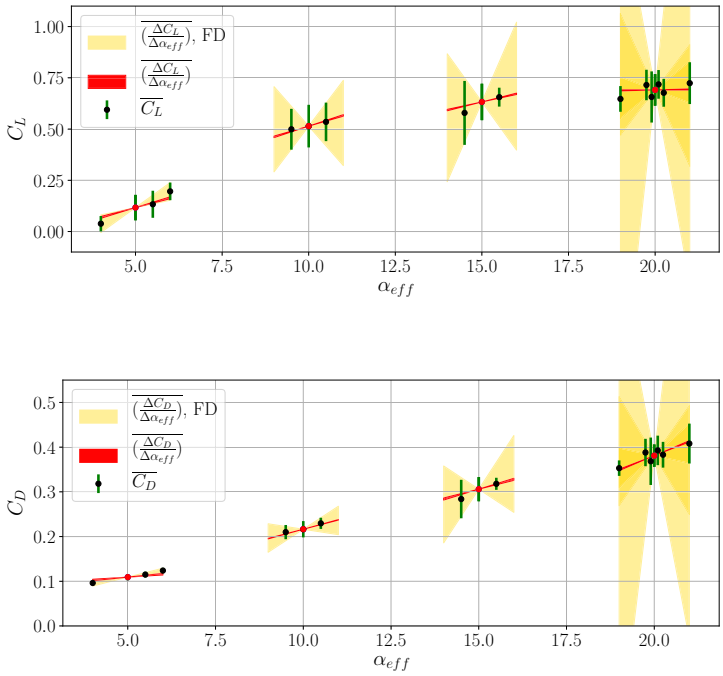


Figure 43: Time-averaged sensitivity of the aerodynamic loads with respect to different angles of attack, where the flow is fully separated.

memory. Using conventional LSS, we would need to store all the Jacobian matrices and other derivatives to disk for all time intervals, and load all of them to a computing device simultaneously. In this case, the KKT system would need 1500 residual matrices, $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \in \mathbb{R}^{n_u \times n_u}$, where $n_u = 2.09 \times 10^6$. Hence, solving the conventional LSS problem would need about 10.517 TB of memory, making it prohibitively expensive. However, the present approach reduces this memory allocation to only 1.3 GB, indicating a $8,087$ times reduction. Besides remarkable dimensionality reduction, and reduced requirement in data storage, it is shown that the accuracy of the time-averaged sensitivity approximation is still preserved.

9.6 Importance of Hyper-Reduction

In the previous examples, we showed how ROMs could be employed to perform sensitivity analysis and uncertainty quantification of either chaotic or non-chaotic problems. These ROMs provided notable reductions in both computation and memory

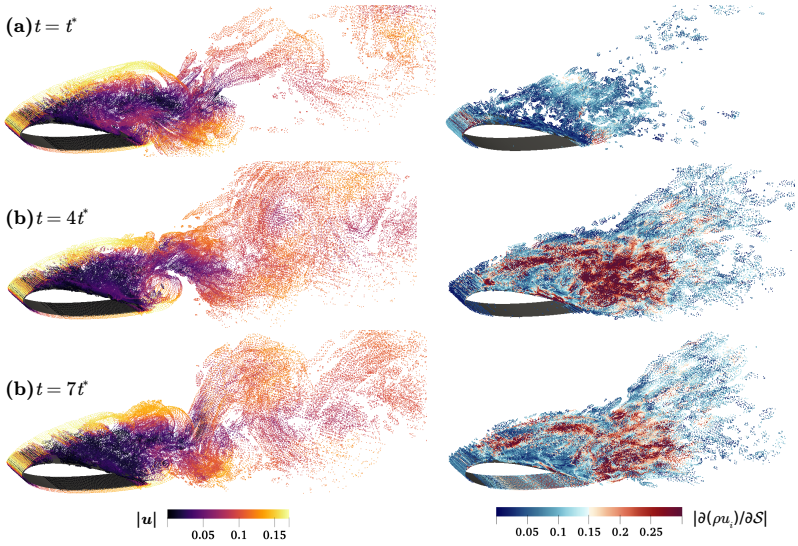


Figure 44: Q-criterion superimposed by the velocity magnitude (left), and corresponding momentum sensitivity magnitude (right) at $\alpha_{eff} = 20°$.

requirements, and paved the way to analyze non-linear systems. In this section, we consider the hyper-reduction only for the third case, since it requires a notable computational cost compared to the other two cases. All computations were carried on twenty Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz cores. To perform the hyper-reduction, we selected only 10% of time intervals in $\Delta\mathbb{T}$, $m_{\text{samp}} = 75$ with equidistance time steps, and computed the aforementioned terms in Eq. (136) at these sample intervals. The time needed to compute these terms at all of these sample intervals is approximately 4.22hr. We now use deep learning to compute the aforementioned terms for the remainder of time intervals (i.e., 675 out of 750).

To develop a surrogate model using a neural network, the first hidden layer has a linear activation function with 80 neurons. This layer is followed by three consecutive Rectified Linear Unit (ReLU) layers with 160 neurons. The last hidden layer also has a linear activation function, containing 80 neurons. Input and output layers are also set to the linear activation function. We used Tensorflow scientific package [1] to develop this neural network architecture. The optimizer was set to Adam with a learning rate of 0.001. Data compression for each term was also performed via the POD technique. After several examinations, these configurations were found to be appropriate for hyper-reduction, and we used this network for the rest of this study.

Table 9 shows the results of two different hyper-reduction cases, where each

case has different recovery criteria. We also compare these cases with the one performed without hyper-reduction, Non-Hyper. The recovery criteria is calculated via $\sum_{i=1}^{r_h} \mathfrak{E}_i / \sum_{i=1}^{r} \mathfrak{E}_i$, where $\mathfrak{E}_i$ is eigenvalue of i^{th} mode, and r_h is the truncation rank for hyper-reduction. As it is shown, in the Non-Hyper case, it took about 38.06hr to prepare the aforementioned terms in Eq. (136) for the remainder of time intervals. However, Hyper-2 with 0.99 of recovery criteria has almost the same time-averaged dynamical sensitivities (i.e., 0.16% error for the lift), and it only needs 7.705hr to compute these remainder terms. With decreasing the recovery criteria to 0.95, the computational time reduces notably, while the error in the sensitivities increases to 0.49% and 0.87% for the lift and drag coefficients, respectively.

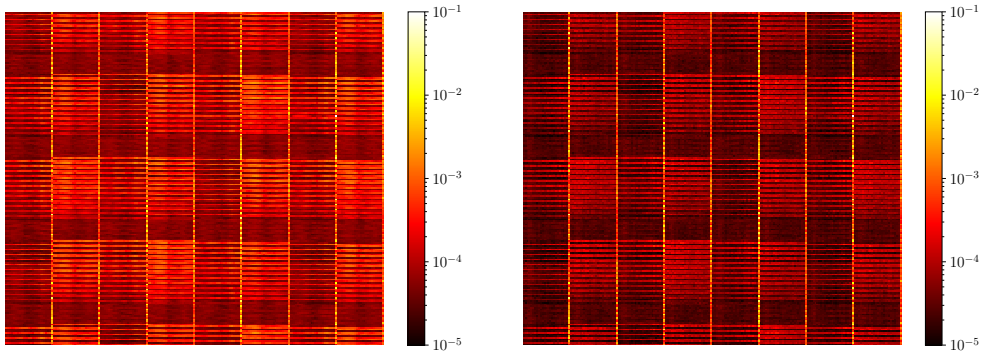
Hyper-reduction	Recovery criteria	r_h	$\overline{\left(\frac{\partial C_L}{\partial \alpha_{eff}}\right)}_{Dyn.}$	$\overline{\left(\frac{\partial C_D}{\partial \alpha_{eff}}\right)}_{Dyn.}$	Time (hr)
Hyper-1	0.95	2	0.01214	0.00454	0.45
Hyper-4	0.99	18	0.01222	0.00458	7.705
Non-Hyper	-	-	0.01220	0.00458	38.06

Table 9: Hyper-reduction considered for the test case 3 at $\alpha_{eff} = 20^{\circ}$.

Among the aforementioned terms in Eq. (136), we selected $\tilde{\mathbf{\Psi}}^{\top}\tilde{\mathbf{\Psi}}$ for further study because of its importance in the evolutional behaviours of the sensitivities in Hilbert space. Figure 45 shows the RMSE computed for each element of $\tilde{\mathbf{\Psi}}^{\top}\tilde{\mathbf{\Psi}}$. As seen, Hyper-1 has larger errors compared to Hyper-2, and as the truncation rank increases, these errors reduce monotonically. In general, Figure 45 indicates that most of the prominent dynamical structures for sensitivities are embedded in lower truncation rank modes. Consequently, we conclude that, in this case, deep learning can predict the aforementioned terms accurately over the remainder of time intervals. This fact causes hyper-reduction to significantly reduce computational time, with less impact on the dominant structures embedded in the aforementioned terms.

9.7 Remarks

Conventional sensitivity analysis fails to compute sensitivities of chaotic dynamical systems. This arises from the fact that chaotic dynamical systems have at least one unstable mode with a positive Lyapunov exponent (LE). Therefore, any small perturbation to the system induces errors in the direction this mode grows exponentially



(a) Hyper-1 (b) Hyper-2

Figure 45: RMSE computed for each element of $\tilde{\mathbf{\Psi}}^{\top}\tilde{\mathbf{\Psi}} \in \mathbb{R}^{r\times r}$.

in time. From a mathematical standpoint, the *shadowing lemma*, in the form of the LSS minimization problem, can be employed to compute accurate sensitivity solutions for chaotic systems. However, this approach is computationally expensive for large-scale fluid systems (i.e., $10^4 \sim 10^5$ times more expensive than FOM simulations) [20]. Although several efforts have been made to reduce the computational cost of conventional LSS, it still suffers from a prohibitive computational burden and large memory requirement, which make LSS impractical for large-scale chaotic problems.

In this study, we proposed a novel physics-constrained data-driven approach to solve the sensitivity function for strongly non-linear dynamical systems. Here we explicitly derived a closure model, in the form of a ROM, from the high-dimensional FOM. We fully discretized the Navier-Stokes equations and forward sensitivity functions, and then transformed them from physical space to an unphysical space (i.e., Hilbert space), creating a low-dimensional model represented as a set of the OΔEs. This transformation approach applies strong physical constraints to the developed closure model. Also, the residual form of the OΔEs allows us to apply them as a series of least-squares minimization problems in order to keep projection errors minimal. Consequently, a platform for this physics-constrained data-driven approach was developed. The closure model leveraged from the minimum-residual property by applying the Petrov-Galerkin approach to the weak form of the Navier-Stokes equations, such as the LSPG approach, described in [29]. Furthermore, a hyper-reduction approach was also proposed to reduce computational burden for sensitivity analysis. We leveraged deep learning to approximate those matrices in Hilbert space, which

are often costly to compute, yielding a significant reduction in computational costs. Consequently, we implemented the proposed approach into an in-house scientific software, called OPtimization Toolkit for Highly non-linear Systems (OPTHiNOS), with parallel algorithms for multicore architectures.

To validate the proposed approach, three canonical fluid dynamics problems were considered. In the first case, flow past a circular cylinder was simulated in the range of $Re = 20 \sim 250$, such that the flow is steady-state or periodic. In this case, the present approach could accurately predict sensitivity solutions. This case was chosen to show that the proposed approach can be considered as an efficient tool for sensitivity analysis of weekly non-linear systems. In the second case, a 2D NACA 0012 airfoil with massive flow separations at high angles of attack, near the stall and post-stall zones, was considered. The Reynolds number for this case was set to $Re = 2.4 \times 10^3$. It was shown that this problem at $\alpha_{eff} = 20^\circ$ has four unstable modes with positive LEs. Sensitivity analysis of this chaotic problem using conventional LSS needs at least 861 GB of memory. However, the current approach used only 236 MB, which indicates approximately $3,650$ times less memory usage than the conventional LSS. In the third case, chaotic flow past a 3D NACA 0020 airfoil at $Re = 2 \times 10^4$ was investigated. It was shown that the present approach can predict the sensitivities at stall and post-stall angles, while FD approximations completely fail in these conditions. The present approach provides an $8,087$ times memory reduction, from 10.517 TB to 1.3 GB. Additionally, it was shown that dimensionality reduction has no notable effect on the accuracy of the approximated sensitivities. Interestingly, the proposed approach can be implemented on a regular desktop, regardless of the number of cores, and LSS can be solved with the tolerance of machine precision. Additionally, hyper-reduction was used for the third case. It was shown that the hyper-reduction accelerates computations approximately 5 to 90 times faster than the case without employing it. Therefore, the utility of the present approach appears promising for future analysis and optimization of complex systems.

Chapter 10

PDE-Constrained Optimization

10.1 Optimization of Stationary Airfoil

In this section, the numerical model developed for the ROM is used for unsteady aerodynamic shape optimization. The objective is drag minimization of a NACA 0012 airfoil at $Re = 1000$ and $M_\infty = 0.1$. This is performed with multiple constraints to ensure the optimization is supervised in $\mathscr{D}$. Two different effective angles of attack, α_{eff}, are selected to investigate the performance of the optimization procedure for pre-stall and post-stall conditions. In this study, $\alpha_{eff} = 8°$, and $\alpha_{eff} = 25°$ correspond to moderate (pre-stall) and massive (post-stall) flow separation, respectively. We deliberately chose these conditions, where the Navier-Stokes equations show strong non-linear dynamics, to demonstrate the robustness of the optimization approach proposed in this study. The ROM-constrained minimization in continuous form can

be written as

$$\underset{\tilde{\boldsymbol{q}}\in\mathbb{R}^r,\mathcal{S}\in\mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} = \frac{1}{T}\int_0^T \beta(C_D)^2 dt + \frac{1}{T}\int_0^T \left(C_L - \overline{C}_{L,target}\right)^2 dt,$$

subject to

$$\begin{aligned}
&\mathcal{R}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}) = 0,\\
&\frac{d\mathcal{R}}{d\mathcal{S}}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}) = 0,\\
&\boldsymbol{h} = \underset{\boldsymbol{z}\in\mathbb{R}^r}{\text{argmin}} \left\| \frac{\partial(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}})}{\partial\mathcal{S}} - \overline{\boldsymbol{v}} - \tilde{\boldsymbol{\Phi}}\boldsymbol{z} \right\|_2^2,\\
&\mathbf{C}_{geom.}(\mathcal{S}) \leq 0,\\
&B_l \leq \mathbf{C}_{boud}(\mathcal{S}) \leq B_u,
\end{aligned} \tag{161}$$

where $C_L = \text{Lift}/(0.5\rho c u_\infty^2)$ and $C_D = \text{Drag}/(0.5\rho c u_\infty^2)$ are the lift and drag coefficients, respectively, ρ is the fluid density, u_∞ is the free-stream velocity, and c is the airfoil chord length. Moreover, $\mathbf{C}_{geom.}(\mathcal{S})$ and $\mathbf{C}_{boud}(\mathcal{S})$ correspond to geometrical constraints and bounds included in the constraint function $\mathbf{C}(\mathcal{S})$. Additionally, B_u and B_l are the upper and lower limits of $\mathbf{C}_{bound}$, respectively, and β is a weighting factor. This weighting is important when using a united objective function. The second term of the objective function guides the optimization toward a target lift

coefficient, $\overline{C}_{L,target}$. The fully discrete form of Eq. (161) is given by

$$\begin{aligned}
\underset{\tilde{\boldsymbol{q}}\in\mathbb{R}^r,\mathcal{S}\in\mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} &= \frac{1}{m_u}\sum_{n=0}^{m_u}\beta\Big(C_D(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)},t^{(n)},\mathcal{S})\Big)^2 \\
&\quad + \frac{1}{m_u}\sum_{n=0}^{m_u}\Big(C_L(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)},t^{(n)},\mathcal{S})-\overline{C}_{L,target}\Big)^2,
\end{aligned}$$

subject to

$$\begin{aligned}
&\mathcal{R}^{(n)}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)},t^{(n)},\mathcal{S})=0,\\
&\frac{d\mathcal{R}^{(n)}}{d\mathcal{S}}(\overline{\boldsymbol{u}}+\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)},t^{(n)},\mathcal{S})=0,\\
&\boldsymbol{v}_{ss}^{(\mathbb{N}_p(n))}\in\underset{\boldsymbol{z}\,\in\,\mathbb{R}^{n_u}}{\text{argmin}}\left\|\frac{\partial\boldsymbol{r}^{(\mathbb{N}_p(n))}}{\partial\boldsymbol{u}^{(\mathbb{N}_p(n))}}\boldsymbol{z}+\frac{\partial\boldsymbol{r}^{(\mathbb{N}_p(n))}}{\partial\mathcal{S}}\right\|_2^2,\\
&r_{state}\in\underset{z\in\mathbb{R}_+\le n_u}{\text{argmin}}\left(\beta z+\left\|\sum_{n=0}^{m_u}\boldsymbol{u}^{(n)}-\overline{\boldsymbol{u}}-\tilde{\boldsymbol{\Phi}}_u\tilde{\boldsymbol{q}}^{(n)}\right\|_2^2\right),\\
&r_{sens}\in\underset{z\in\mathbb{R}_+\le n_u}{\text{argmin}}\left(\beta z+\left\|\sum_{n=1}^{m_p}\boldsymbol{v}_{ss}^{(\mathbb{N}_p(n))}-\overline{\boldsymbol{v}}_{ss}-\tilde{\boldsymbol{\Phi}}_v\tilde{\boldsymbol{h}}^{(\mathbb{N}_p(n))}\right\|_2^2\right),\\
&\mathbf{C}_{geom.}(\mathcal{S})\le 0,\\
&B_l\le\mathbf{C}_{boud}(\mathcal{S})\le B_u,
\end{aligned}\tag{162}$$

where argmin in Eq. (161) is expanded to show the training process and the rank selection criteria in Eq. (162).

The shape of the airfoil is changed using "B-spline" shape functions, using control points shown in Figure 46. These control points define smooth perturbations to the suction and pressure surfaces of the airfoil. Additionally, three fixed points at the leading edge, and another at the trailing edge, are used as constraints to avoid defective geometries in these regions. These fixed points are also responsible for keeping the angle of attack constant. As mentioned above, all of these constraints are included in $\mathbf{C}(\mathcal{S})$.

10.1.1 Computational Setup

A 2D structured computational domain with a C-topology is used. To solve a compressible flow field, an in-house CFD software, the High-ORder Unstructured Solver (HORUS) Version 0.2.0, is used. The spatial discretization in HORUS uses the Flux Reconstruction (FR) approach [65], and we set the solution polynomial degree to

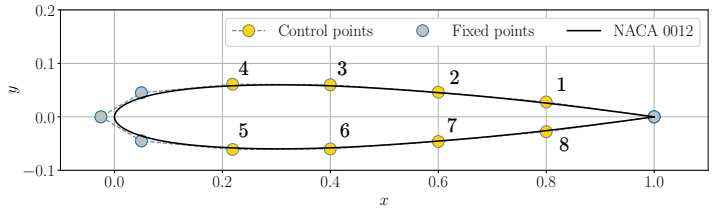
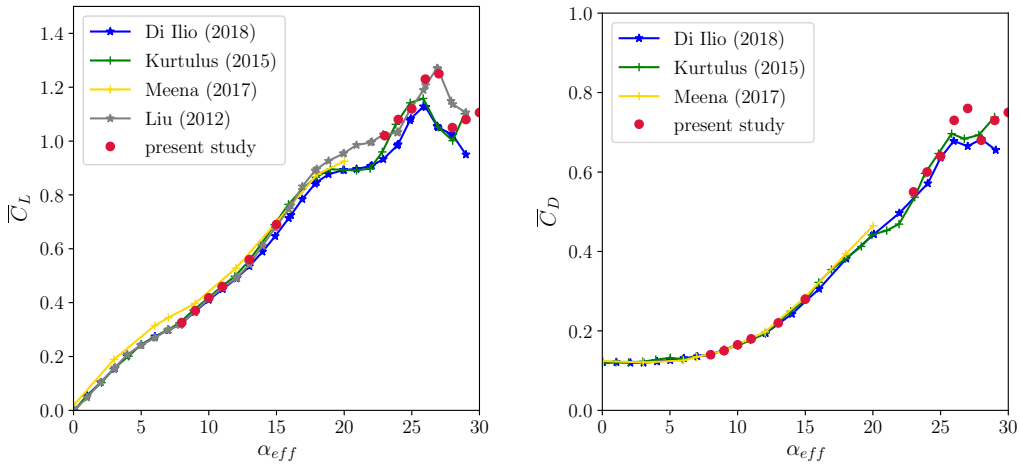


Figure 46: Geometrical definition of the airfoil using control and fixed points.



(a) Time-averaged lift coefficient.

(b) Time-averaged drag coefficient.

Figure 47: Comparison of present numerical simulations with the data from literature [42, 52, 77, 81].

$p_s = 2$, which recovers third-order accuracy. The second-order Backward Differentiation Formula (BDF2) is used for temporal discretization. The Prandtl number and Mach number are set to $Pr = 0.72$ and $M_\infty = 0.1$, respectively. Since the governing equations are solved implicitly, the time step is chosen to be $\Delta t = 0.05t^*$, where $t^* = c/u_\infty$ is the convective time.

The number of elements for the initial mesh is approximately 3.7×10^3. With a solution polynomial degree $p_s = 2$ the total number of solution points is $n_u = 1.35 \times 10^5$. Figure 47 shows a comparison of time-averaged lift and drag coefficients at several angles of attack, showing good agreement with available reference data. The baseline NACA 0012 airfoil was then modified using Sequential Least-Squares Programming (SLSP) as an optimizer from the Scipy package [137]. Additionally, the SVD, as an eigenvalue problem is solved via SLEPc [59] in parallel, and PETSc [9, 40], compiled with OpenMPI [41], is used for parallelization. We simulated each FOM

case until $40t^*$. It was observed that this period is sufficient to obtain statistically converged results. Data snapshots were then collected over a period of $T = 15t^*$. The number of subspaces (the rank of the ROB) is a crucial parameter that should be assessed before starting optimization. Having an inappropriate rank may cause a bias in the ROM, creating a poor dynamical model. Therefore, a ROB was built for the reference case (NACA 0012 airfoil at $Re = 1000$). Then the arrangement and accumulation of singular values were analyzed according to Eq. (162). It is observed that if the rank of the ROM is selected to be $r_{state} = 50$, the ROB contains most of the interpretive information of flow structures. Additionally, $m_p = 20$ samples for each control point, P(1) to P(8), are collected to develop the ROB of the sensitivity function with the rank of $r_{sens} = 80$. These derivatives are computed with a second-order central finite difference scheme. The sensitivity function is also discretized via the first-order BDF1 scheme for sensitivity analysis. It is worth mentioning that r_{sens} should be modified since some modes in $\tilde{\mathbf{\Phi}}_v$ might be unnecessary, leading to an increase in the bias of the sensitivity. On the other hand, $\tilde{\mathbf{\Phi}}_v$ should have a high enough rank to minimize the detrimental effect of variance on the results. Therefore, the residual of the sensitivity function was monitored to ensure it was lower than 10^{-4}. Using this approach, we will now consider shape optimization at two different angles of attack.

10.1.2 Shape Optimization of a NACA0012 at $\alpha_{eff} = 8^\circ$

Setup

A NACA 0012 airfoil at $\alpha_{eff} = 8^\circ$ is in the pre-stall condition with $\overline{C}_L = 0.323$ and $\overline{C}_D = 0.14$. For drag minimization we select $\overline{C}_{L,target} = 0.32$ and $\beta = 10$ in Eq. (162). We set the effective angle of attack as a variable that can be changed by unlocking the fixed points at the leading edge. In this case, $\mathbf{C}(\mathcal{S})$ includes ten geometrical constraints with sixteen bounds. Additionally, one extra constraint is considered to limit $\overline{C}_L$. The minimization is then advanced for ten design iterations.

Optimization Results

Figure 48 shows the optimization results. The time-averaged objective function, $\overline{\mathcal{J}}$, continuously decreases until the fourth iteration. After that, the constraints slow

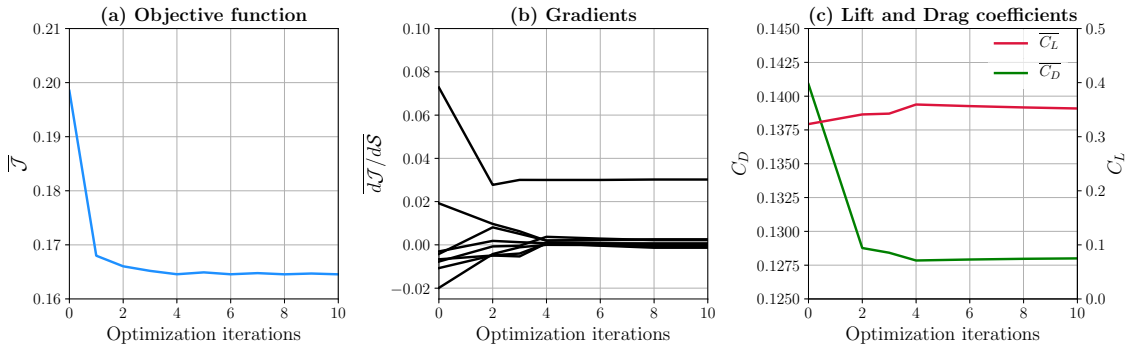


Figure 48: Optimization progress in the shape optimization of the airfoil at $Re = 1000$ and $\alpha_{eff} = 8°$.

down the reduction in $\overline{\mathcal{J}}$. Moreover, the time-averaged gradients of the objective function are displayed in Figure 48 (b). As shown, all gradients converge to zero, except one that remains constant at approximately 0.03. This cannot be driven lower due to the imposed constraints. Figure 48 (c) illustrates how the time-averaged lift and drag coefficients are changed during optimization. The drag coefficient is reduced by approximately 9.35%, while producing 9% higher lift than the reference airfoil.

To compare the reference and optimized designs in terms of geometry and flow characteristics, Figure 49 is provided. The pressure changes rapidly at the leading edge of the optimized airfoil. These variations on the suction side produce additional increased lift, while the rest of the optimized airfoil has no notable contribution to lift generation. Furthermore, the optimized airfoil has a nose slightly lower than the reference airfoil, which influences the effective angle of attack. Interestingly, the optimized airfoil produces higher lift at lower effective angle of attack.

Flow Structures

The slender leading edge of the optimized airfoil has a notable role for the separation point on the suction side. As shown in Figure 50 (a), the separation point, where the flow detaches from the surface due to an adverse pressure gradient, is at $x \simeq 0.5c$. After the separation point, the flow confined on the suction side generates a large pressure difference, and subsequently, higher drag. This confined area is also directly related to the wake of the airfoil. However, the optimized airfoil's nose induces a sudden adverse pressure gradient near the leading edge, which leads to Laminar Separation Bubble (LSB) formation, as shown in Figure 50 (b). Consequently, it

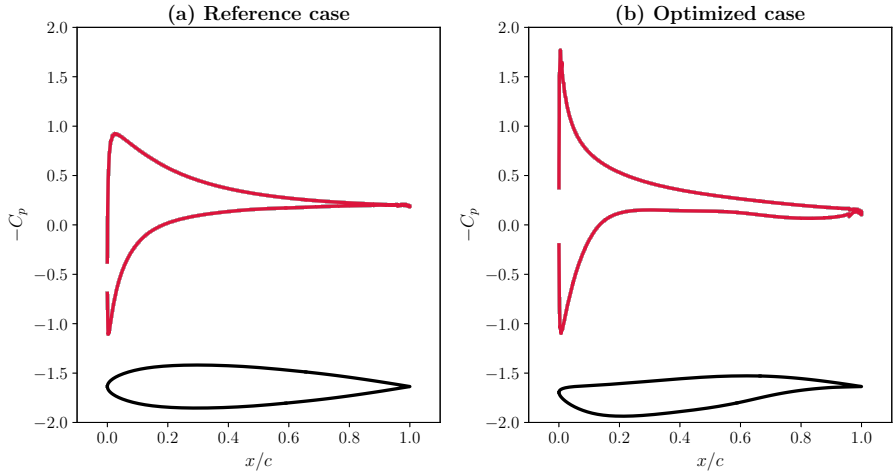


Figure 49: The geometry of the airfoil and the time-averaged pressure coefficient along the airfoil surface.

leads to a delayed separation point located at approximately $x \sim 0.8c$. This delayed flow separation leads to a smaller wake behind the airfoil, which reduces drag.

10.1.3 Shape Optimization of a NACA0012 at $\alpha_{eff} = 25°$

Setup

There are different reports on the stall point of a NACA 0012 at $Re = 1000$. Liu et al. [81] reported that the stall angle of attack is 27° with $\overline{C}_L = 1.28$. Kurtulus [77] also showed the stall lift coefficient as $\overline{C}_L = 1.25$ with a stall angle of 26°, and Di Ilio et al. [42] detected the same stall angle, but with $\overline{C}_L = 1.1$. The present study shows that stall occurs at 27° with $\overline{C}_L = 1.25$. Irregular vortex shedding is expected due to massive flow separation on the airfoil's suction side. Our objective is to minimize the drag coefficient of the airfoil at $\alpha_{eff} = 25°$. The lift and drag coefficients for the NACA 0012 at this angle of attack are $\overline{C}_L = 1.17$ and $\overline{C}_D = 0.66$, respectively.

To define the optimization problem, we selected $\beta = 2$, and $\overline{C}_{L,taget} = 1.25$ to force the objective function to keep the lift coefficient relatively constant while reducing drag. Having a slightly higher $\overline{C}_{L,target}$ than $\overline{C}_L = 1.17$ was found to prevent the magnitude of the lift coefficient dropping significantly in early design iterations. Additionally, $\mathbf{C}(\mathcal{S})$ includes twelve geometrical constraints and sixteen bounds for the control points. Limiting the lift coefficient and the angle of attack are two other

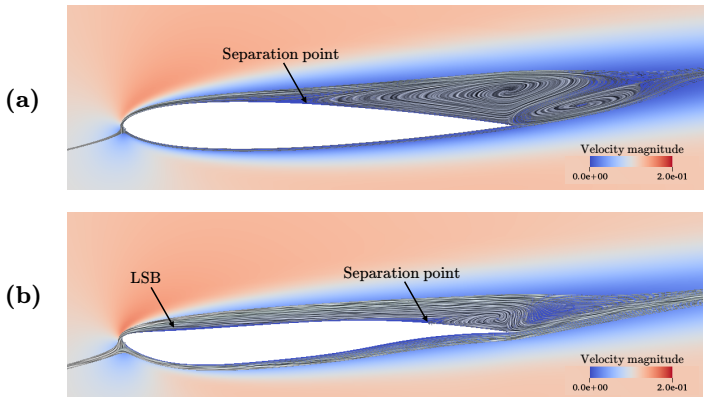


Figure 50: Velocity contours with superimposed streamlines for $Re = 1000$ and $\alpha_{eff} = 8°$; (a) reference, and (b) optimized airfoils.

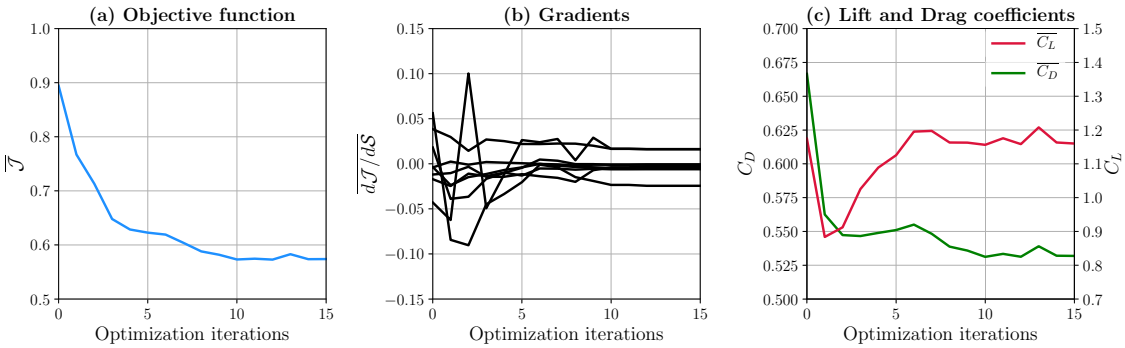
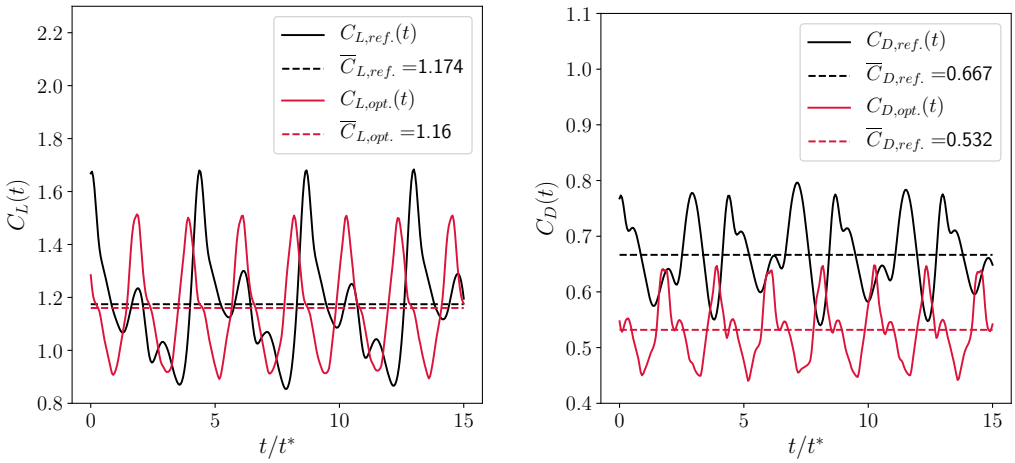


Figure 51: Optimization progress in the shape optimization of the airfoil at $Re = 1000$ and $\alpha_{eff} = 25°$.

constraints. Therefore, this optimization problem contains thirty constraints in total.

Optimization Results

To show how the design changes during optimization, Figure 51 and Figure 52 are provided. In Figure 51 (a), the time-averaged objective function, $\overline{\mathcal{J}}$, begins to reduce continuously until the tenth optimization iteration. Later the objective function reduces slightly or oscillates when the design parameters are close to the local optimum point in the design space. As shown in Figure 51 (b), gradients of the objective function either plateau or converge to zero. According to the gradient-based formulation, gradients of the objective function ideally should be zero at the local/global



(a) Instantaneous lift coefficient (b) Instantaneous drag coefficient

Figure 52: Instantaneous lift and drag coefficients for both reference and optimized cases.

optimum. However, bounding the control points with geometrical constraints may restrict gradients from reaching identically zero. Figure 51 (c) also represents the variation of time-averaged lift and drag coefficients. The lift coefficient drops at the first optimization iteration, and later, it rises toward the target value $\overline{C}_{L,target}$. The drag coefficient diminishes significantly.

Figure 52 depicts the lift and drag coefficients versus non-dimensional time t/t^*. From Figure 52 (a), the time-averaged lift coefficient reduces slightly from $\overline{C}_{L,ref.} = 1.17$ to $\overline{C}_{L,opt.} = 1.16$. However, a significant difference between the time-averaged drag coefficients yields an approximately 20% reduction in drag. Figure 53 shows the distribution of the time-averaged pressure coefficient along the airfoil surface, where the shaded area shows pressure variations over time. From Figure 53 (a), the pressure on the reference airfoil suction side fluctuates across the majority of the chord length, particularly near the trailing edge. However, pressure fluctuations for the optimized design are significantly reduced, as shown in Figure 53 (b). Because the intensity of large-scale coherent structures in the flow field tends to amplify pressure fluctuations on the suction side, this indicates that the optimized airfoil creates weaker wake vortices.

The spatial integral of the time-averaged pressure on the surface yields the time-average pressure force exerted on the airfoil. The majority of this pressure force

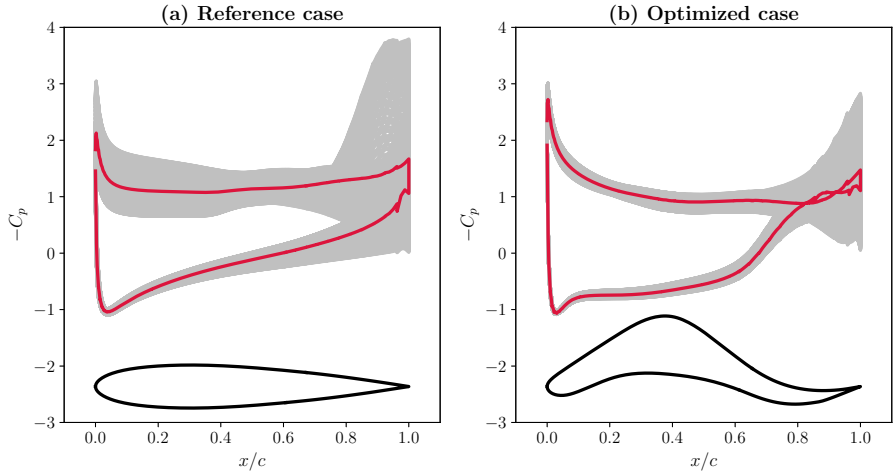


Figure 53: The geometry of the airfoil and the time-averaged pressure coefficient (red) along the airfoil surface. The shaded area also displays the variations of the pressure coefficient over time.

originates in the vicinity of the leading edge. However, in the optimized case, the time-averaged pressure force near the trailing edge is significantly reduced. The pressure curves in this region ($x/c = 0.8 \sim 1$) are similar, resulting in minimal lift and drag production in this region. Despite generating significantly less lift near the trailing edge, the airfoil is still able to satisfy the lift coefficient constraint. This is achieved via the shear layer produced at the leading edge of the airfoil. As shown in Figure 53 (b), sharp variations in the pressure coefficient around the airfoil's leading edge arise from an intense, but relatively stable, shear layer. Furthermore, the airfoil's suction surface is relatively flat and tangent to the flow, leading to the enhanced lift. In contrast, the sharp difference of the pressure coefficient at the leading edge of the reference airfoil is less intense than the optimized one (Figure 53 (a)). The pressure variations on the suction side near the nose of the reference airfoil fluctuate change with a large amplitude, indicating an unstable shear layer at the leading edge.

Flow Structures

Here we will explore the behaviour of coherent structures in the wakes of the reference and optimized designs. Figure 54 and Figure 55 show time histories of the lift and drag coefficients, and contours of vorticity at several instants during a shedding cycle. Green arrows show the direction of the jet formed at the interface of

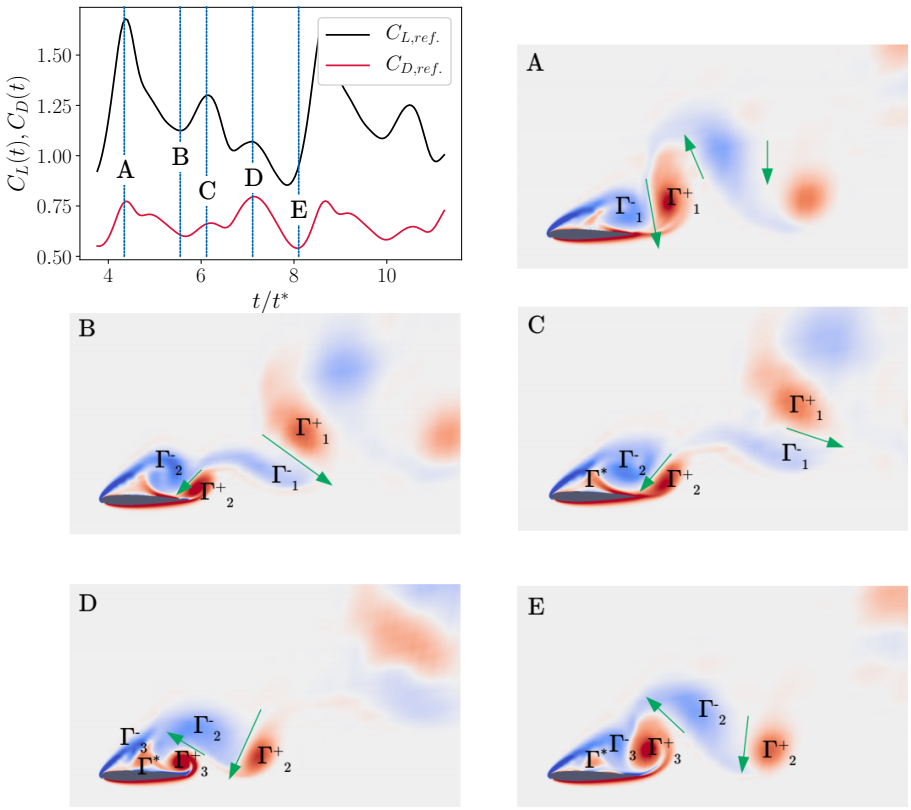


Figure 54: Flow structures around the reference airfoil.

two counter-rotating vortices. In frame "A", the first Trailing Edge Vortex (TEV), Γ_1^+, leaves the airfoil surface and travels downstream. Moreover, the first Leading Edge Vortex (LEV), Γ_1^-, increases in intensity and envelopes the suction surface of the airfoil. Hence, the pressure coefficient on the suction surface reaches a minima, yielding high lift. The shedding of LEV and TEV manifests as a von-Karman street in the wake of the airfoil. In frame "B", the aerodynamic forces decrease, since Γ_2^- is not strong enough to produce a notable low-pressure zone on the suction surface of the airfoil. Furthermore, Γ_2^+ does not allow Γ_2^- to grow. According to Kelvin's circulation theorem, after Γ_1^- separates, the second TEV, Γ_2^+, forms and gains energy by absorbing the kinetic energy coming from Γ_2^-, and the shear layer at the trailing edge of the airfoil. This energy transfer indicates that if two counter-rotating vortices are located beside each other, the low-energy vortex absorbs energy from the high-energy vortex through shear. In frame "C", the pair vortex, Γ^*, has developed on the suction surface. This counter-rotating vortex reduces the connection of Γ_2^- with the airfoil surface. It is observed that Γ^* in the reference case plays an important role in

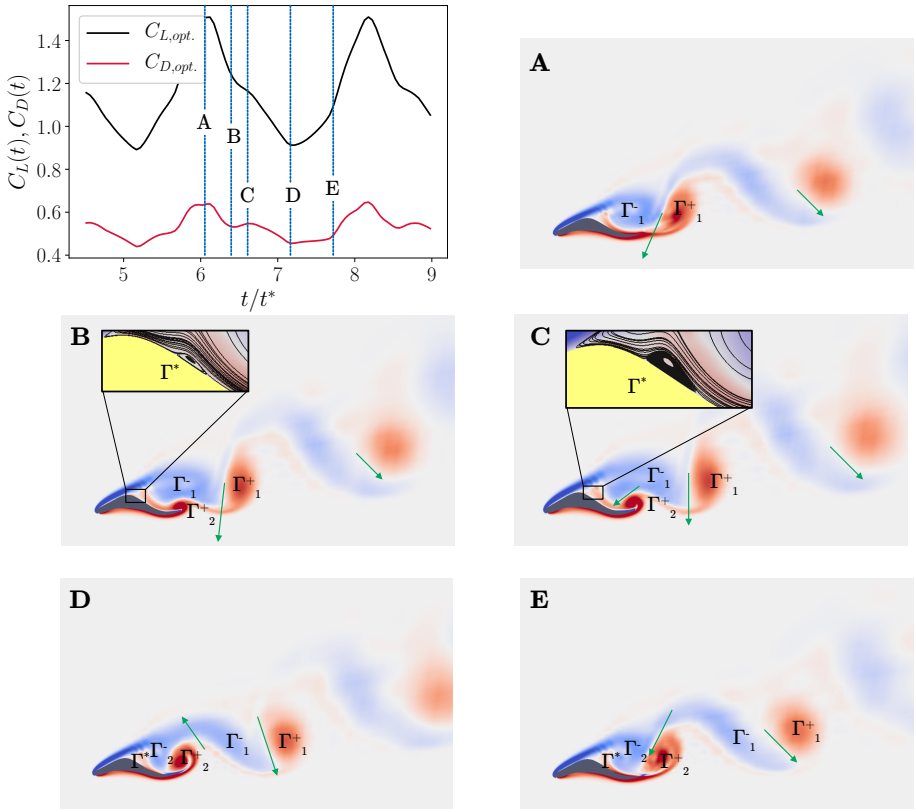
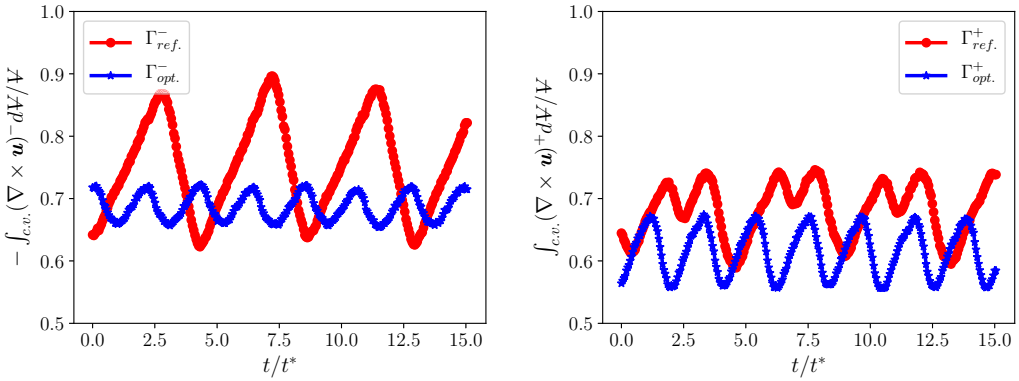


Figure 55: Flow structures around the optimized airfoil.

the quasi-periodic behaviour of vortical structures in the wake. Different intensities and growth rates for Γ^* leads to different vortex shedding patterns, which influences the LEV and TEV dynamics. This is evident in time variations of the lift and drag coefficients.

Figure 55 shows different instants during one shedding cycle for the optimized design. In frame "A", Γ_1^+ has detached from the airfoil, while Γ_1^- envelops the suction surface. When Γ_1^- reaches its maximum intensity, the lift is maximized. In frame "B", a shear layer forms a LSB, which initiates Γ^*. As Γ^* increases in intensity, it ejects Γ_1^- away from the surface with the help of Γ_2^+ (as shown in frame "C"). It is observed that LEV and TEV enter a harmonic vortex shedding pattern for the optimized design, where the aerodynamic loads repeat cyclically. The flow field becomes more complicated in the post-stall region, where at least two shedding frequencies appear. Hence, we observe that the optimized design has a simpler shedding pattern, with harmonic interaction between the LEV and TEV.

To quantify the strength of these vortices, a control surface is placed around



(a) Instantaneous circulation generated by LEV.

(b) Instantaneous circulation generated by TEV.

Figure 56: Time variation of the total strength of core vortices (i.e., LEV, and TEV) over $15t^*$ for both reference and optimized cases.

the airfoil over which an integral of $\nabla \times \boldsymbol{u}$ is computed, yielding circulation. The control surface boundaries were placed far enough away from the airfoil such that this circulation was not sensitive to them. Figure 56a shows the total circulation magnitude of the LEV over $15t^*$ for the reference case, $\Gamma^{-}_{ref.}$, and the optimized case, $\Gamma^{-}_{opt.}$. The strength of $\Gamma^{-}_{ref.}$ is notably higher than $\Gamma^{-}_{opt.}$. Furthermore, the oscillation of $\Gamma^{-}_{ref.}$ is greater, while $\Gamma^{-}_{opt.}$ has smaller oscillations with nearly constant peaks. The same observation is made from Figure 56b showing the strength of $\Gamma^{+}_{ref.}$ and $\Gamma^{+}_{opt.}$. This indicates that the optimized airfoil produces a weaker LEV. This is because the optimized airfoil guides the flow over the surface more smoothly, which leads to simpler interaction between the LEV and TEV structures in the wake.

Sensitivity Analysis

This section analyses the sensitivity of the state vector and the objective function with respect to the design parameters. According to Eq. (161), the total gradient of the objective function, $\overline{\mathcal{J}}$, can be written as

$$\frac{\overline{d\mathcal{J}}}{d\mathcal{S}} = \frac{2}{T}\int_0^T \beta \frac{dC_D}{d\mathcal{S}} C_D dt + \frac{2}{T}\int_0^T \frac{dC_L}{d\mathcal{S}}\left(C_L - \overline{C}_{L,target}\right)dt, \tag{163}$$

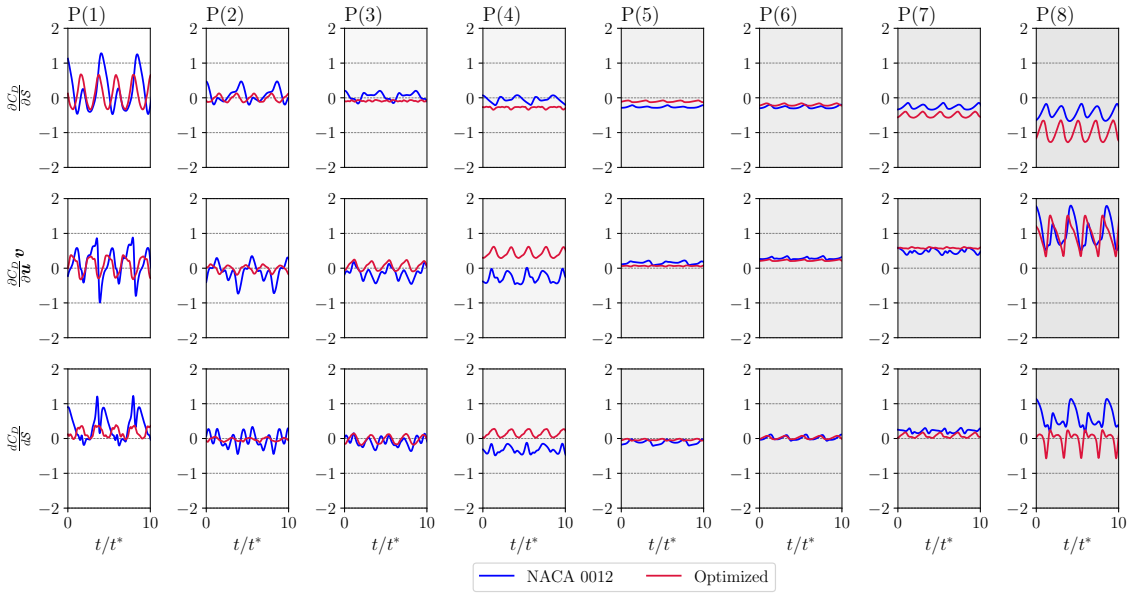


Figure 57: Instantaneous gradients of the drag coefficient (geometrical, state, and total sensitivities) with respect to the control points.

where

$$\begin{aligned} \frac{dC_L}{d\mathcal{S}} &= \frac{\partial C_L}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial C_L}{\partial \mathcal{S}}, \\ \frac{dC_D}{d\mathcal{S}} &= \frac{\partial C_D}{\partial \boldsymbol{u}}\boldsymbol{v} + \frac{\partial C_D}{\partial \mathcal{S}}, \end{aligned} \tag{164}$$

and $\boldsymbol{v}$ is approximated by $\boldsymbol{v} \approx \overline{\boldsymbol{v}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}}$. Since the design objective is drag minimization, we will isolate this term. The total sensitivity of the drag coefficient, $dC_D/d\mathcal{S}$, is dependent on the geometrical sensitivity, $\partial C_D/\partial \mathcal{S}$, and the state sensitivity, $(\partial C_D/\partial \mathcal{S})\boldsymbol{v}$. The geometrical sensitivity is only dependant on the shape of the airfoil, independent of the flow field. Also, the state sensitivity is only dependent on the flow field, independent of perturbations in the airfoil geometry. Therefore, these two sensitivities will be explored seperately.

Figure 57 displays the sensitivity of the drag coefficient with respect to control points (shown in Figure 46), where each column corresponds to a control point. The first, second, and third rows show the time variation of the geometrical sensitivity, state sensitivity, and total sensitivity. From Figure 53, the pressure coefficient has large fluctuations near the trailing edge, which suggests that the aerodynamic loads are highly sensitive here (P(1) and P(8)). This implies that the trailing edge has a significant influence on TEV formation, affecting the drag coefficient. Furthermore,

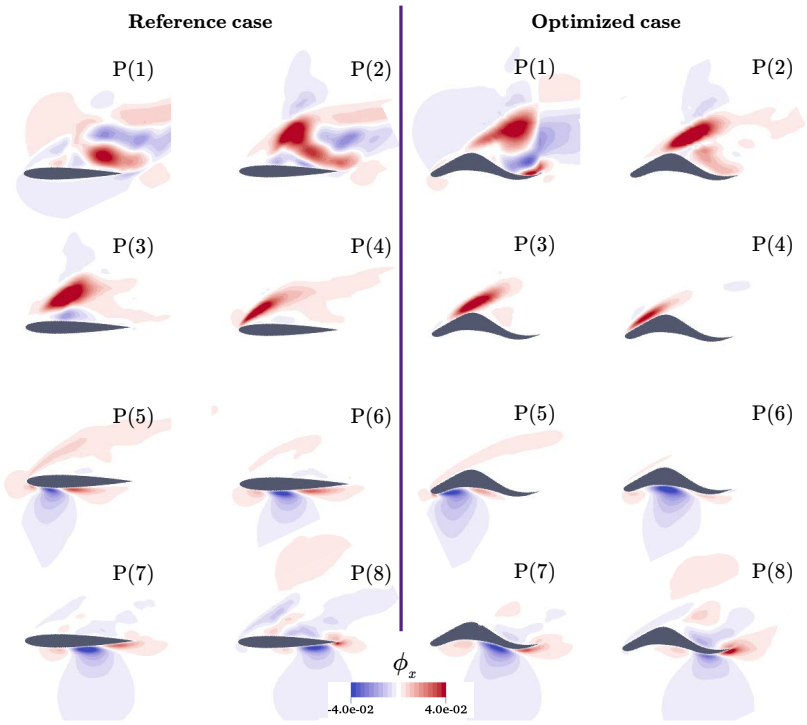


Figure 58: The primary modes of the solutions of the sensitivity for both reference and optimized cases.

the drag coefficient is sensitive to the suction surface of the airfoil (P(2) to P(4)), where sensitivities fluctuate with the flow. However, since the flow field on the pressure side of the airfoil is relatively steady, the sensitivities for P(5) and P(6) are also relatively steady. Moreover, the time variation of $dC_D/d\mathcal{S}$ reveals that the reference airfoil is more sensitive than the optimized design.

Figure 58 shows the primary mode of the sensitivity, which corresponds to the fluid momentum in x direction, $\tilde{\mathbf{\Phi}}_x$. For the reference case, P(1) has a notable effect on the TEV and drag reduction. Also, P(1) affects a wide range of secondary structures in the wake. For the optimized case, P(1) has less sensitivity to the TEV. On the other hand, the optimized airfoil has higher sensitivity to the LEV. In general, for both the reference and optimized cases, P(2) and P(3) control the shear layer at the leading edge, which directly changes the LEV and high-pressure differences at the suction surface of the airfoil. However, P(5) to P(7) indicate that the pressure surface of the airfoil does not significantly affect vortical structures in the wake of the airfoil. On the other hand, P(8) influences these vortical structures on the suction surface of the airfoil. The conclusion drawn from Figure 58 is that control points in the vicinity of

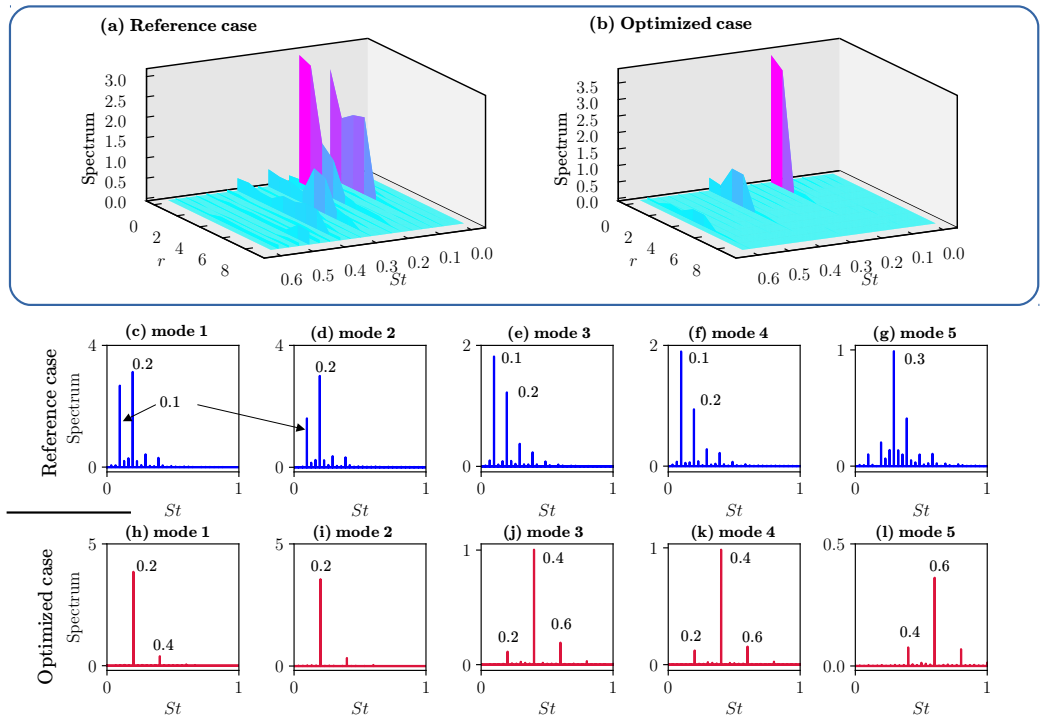


Figure 59: Variation of frequency for different modes in the reference and optimized cases.

the trailing edge have a significant effect on drag minimization. On the other hand, control points adjacent to the leading edge have a notable influence on lift.

Dynamics

In the previous sections, we explained how optimization changes the airfoil shape, and hence, flow structures in the wake. Then, we focused on the behaviour of the sensitivity solution to understand each control point's effect on the flow field. Here, we consider the optimization procedure from a different perspective in Hilbert space. We transform the optimization problem from $\mathscr{P}$ to $\mathscr{H}$, and then we identify the characteristics of the underlying dynamical system.

Figure 59 shows a Discrete Fourier Transform (DFT) of the generalized coordinates. Figure 59 (a) shows the Strouhal number distribution for ten subsequent modes, indexed by their rank r. The Strouhal number is defined as $St = f_s l_c / u_\infty$, where f_s, and l_c are the frequency and characteristic length, taken here to be the wake width, respectively. Dominant Strouhal numbers are evident in modes with

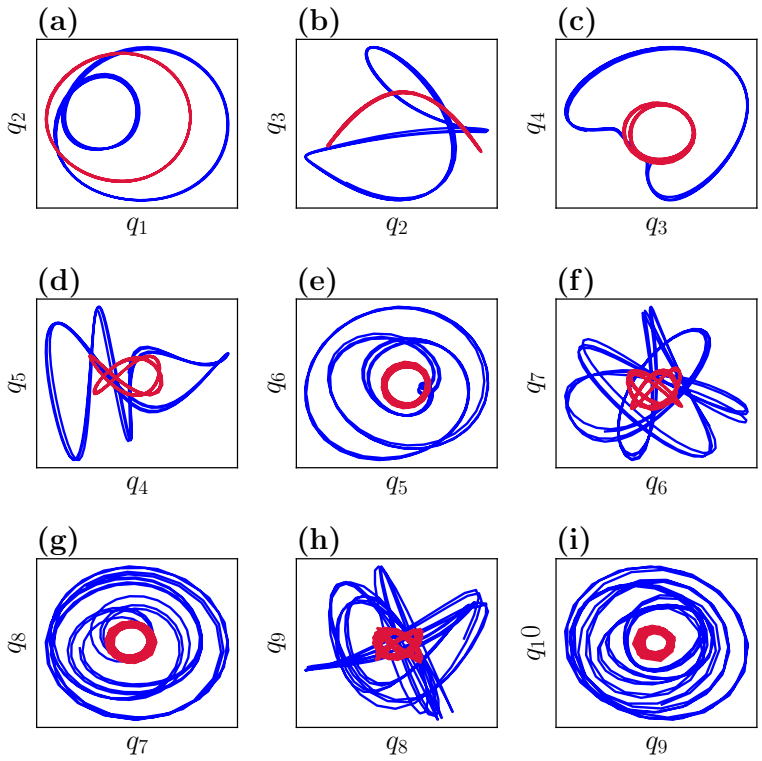


Figure 60: Correlations between modes: reference (blue) and optimized (red) cases.

high spectrum magnitudes. For the reference case, several dominant shedding modes are observed. However, the optimized case has only three dominant modes, as shown in Figure 59 (b). This suggests that the wake is significantly simpler in nature for the optimized case. Figures 59 (c) to (g) display the amplitude spectrum versus Strouhal number for the first five dominant modes for the reference case. Frequencies detected in the first four modes are $St_1 = 0.2$ and $St_2 = 0.1$. Figure 59 (h) and Figure 59 (i) also show that the dominant Strouhal number is $St_1 = 0.2$ for the optimized case. Yarusevich et al. [147] performed an extensive experiment on the wake of the reference airfoil, and the reported Strouhal number was $St \approx 0.2$ for an airfoil at $\alpha_{eff} = 10°$. Additionally, similar observations can be found in the results of Huang and Lin [63]. They considered wake structures of a NACA 0012 airfoil, and mentioned that the Strouhal number of the dominant vortex shedding for angles of attack higher than $\alpha_{eff} = 25°$ remains approximately constant with a value of $St \approx 0.2$.

Correlations can be defined as interactions between the generalized coordinates,

$q_i \in \boldsymbol{q} = [q_1, q_2, \ldots, q_r]^\top$, to understand the dynamical behaviour of the system. Figure 60 shows correlations for the first ten modes in the reference and optimized cases. These ten modes correspond to five "pair" modes in the dynamical system. The pair modes are two subsequent modes that show the same, or similar, dynamical behaviour with a certain offset. Figures 60 (a), (c), (e), (g), and (i) show correlations of "pair" modes. The remaining figures show correlations between "non-pair" modes. Each correlation loop, specifically in "pair" modes, belongs to a specific vortical structure. Figure 60 (a) shows that the correlation of the reference case has two loops, which indicates that two different dominant patterns in the vortex shedding occur one after another. However, the optimized case has only one loop indicating one dominant pattern occurs cyclically. High rank correlations also contain more loops, which indicates that each dominant pattern has different forms that repeat harmonically in different shedding cycles. It is worth pointing out that as the dynamical system experiences higher non-linearity, the number of loops in corresponding correlations increases. Interestingly, in the optimized case, high rank correlations have multiple similar loops, indicating linear behaviour of the vortical structures in the wake of the optimized airfoil.

Figure 61 and Figure 62 show correlations coloured by sensitivity magnitudes, $h_i \in \boldsymbol{h} = [h_1, h_2, \ldots, h_r]^\top$, for the reference and optimized cases, respectively. The maximum sensitivity belongs to the first rank, and as the rank increases, the sensitivity magnitude reduces. The primary weight of the sensitivity belongs to prominent vortical structures in the wake of the airfoil. As we discussed earlier, the sensitivity of the optimized case is less than that of the reference case, which is observed by comparing sensitivity magnitudes in Figure 61 and Figure 62. It is worth mentioning that the maximum sensitivity magnitude for each mode happens when the generalized coordinates in correlations move to another loop. This exchange occurs when a LEV or TEV sheds downstream. Therefore, based on these observations, the sensitivity of the generalized coordinates in Hilbert space is strongly related to the sensitivity of the state vector in physical space. Consequently, applying optimization to the dynamical system in Hilbert space is analogous to optimization in physical space.

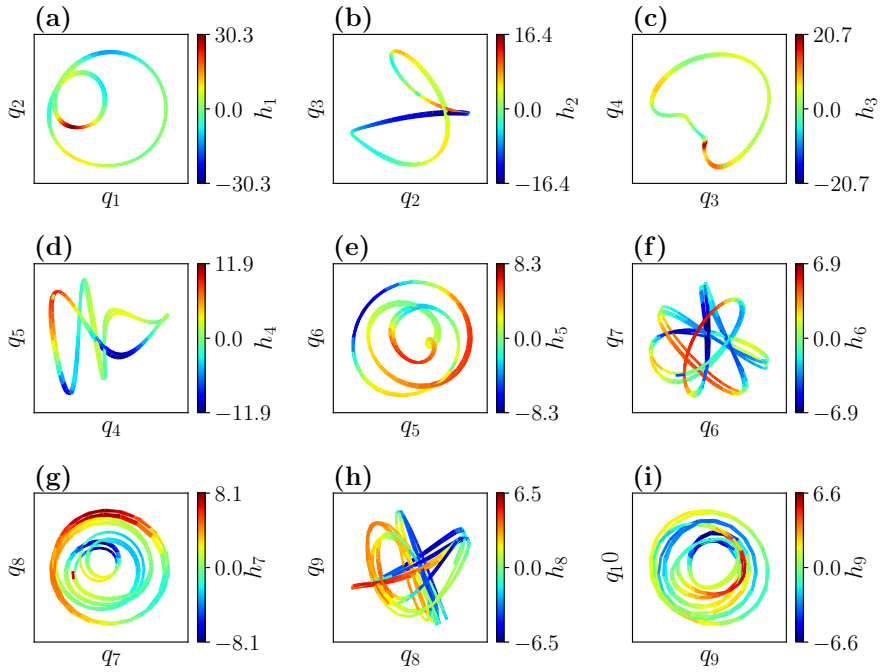


Figure 61: Correlations of modes coloured by sensitivity magnitude for the reference case.

10.2 Optimization of Flapping Wing

One of the challenging design problems in chaotic flow belongs to Fluid-Structure Interaction (FSI), where flow is induced continuously by a solid object and vice versa. For example, birds oscillate their wings, and produce complicated flow structures, which help generate more lift than a stationary wing. This type of FSI problem results in a dynamic stall phenomenon, an important matter in wind turbines, helicopters, aircraft take-off/landing, etc. Figure 63 shows one of the bio-inspired robots manufactured by FESTO [1]. This smart bird mimics the seagull's flying pattern, and produces thrust and lift by oscillating wings. From a mathematical standpoint, optimization of wings for this robot is quite challenging work due to the chaoticity of turbulent flow, and complexity of flow structures in the wake of this robot. Therefore, there is still no robust, and well-established framework to perform optimization in FSI problems. Therefore, we will focus on possible ways to resolve limitations in these complex problems.

[1] https://www.festo.com/group/en/cms/10238.htm

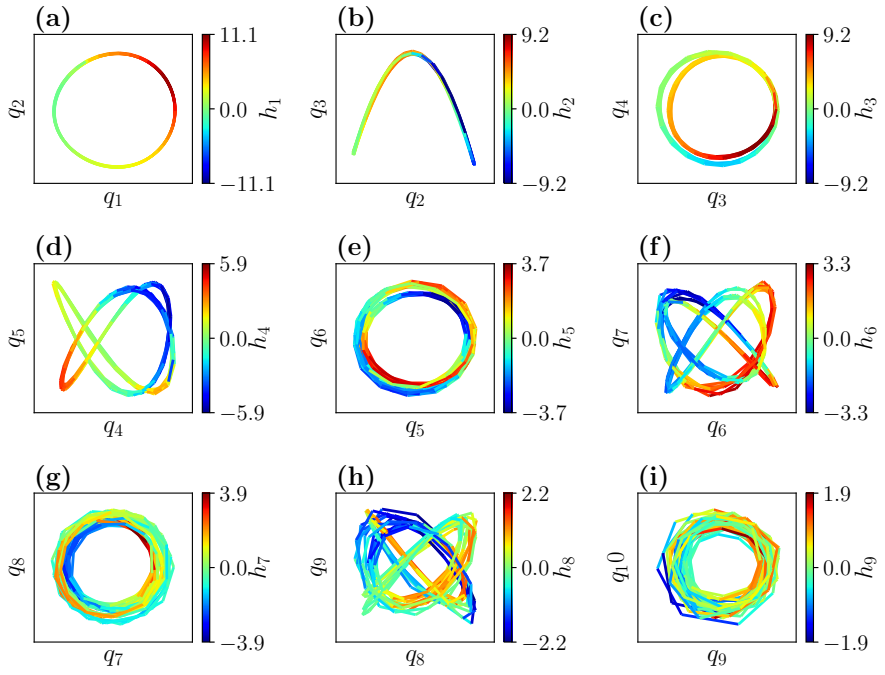


Figure 62: Correlations of modes coloured by sensitivity magnitude for the optimized case.

10.2.1 Kinematics & Aerodynamic Characteristics

In flapping wing problems, the reduced frequency, k_a, is a non-dimensional parameter, representing a relation between the oscillational velocity of the wing and free-stream velocity, u_∞. We can define this reduced frequency as $k_a = \pi f_a l_c / u_\infty$, where f_a is the flapping frequency, and l_c is the characteristic length. We also define the Reynolds number according to the chord length of the wing, c, as $Re = \rho_\infty c u_\infty / \mu$, where ρ_∞ and μ are free-stream fluid density and viscosity, respectively.

In the present work, the kinematics of a wing section, $\mathcal{X}(t, \mathcal{S}) = \{x \in \mathbb{R}^4 \mid x \subset \mathscr{D}\}$, is considered in cartesian coordinates. The translation and pitching functions of a wing can be defined as

$$\mathcal{X}(t, \mathcal{S}) = \begin{bmatrix} \mathrm{x}(t) \\ \mathrm{y}(t) \\ \mathrm{z}(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{y}_0 \sin(2\pi f_a t) \\ 0 \\ \theta_m + \theta_0 \sin(2\pi f_a t + \theta_s) \end{bmatrix} \tag{165}$$

where, y_0 is the plunging amplitude, normally represented in non-dimensional form,

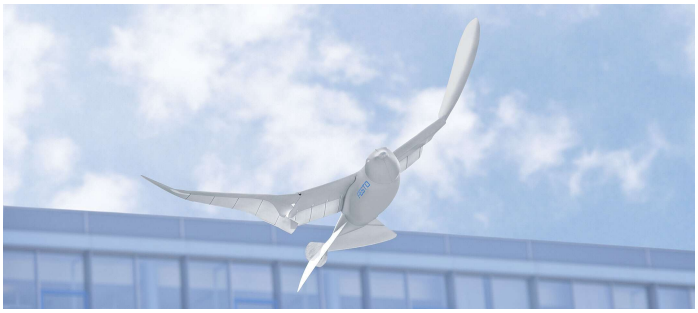
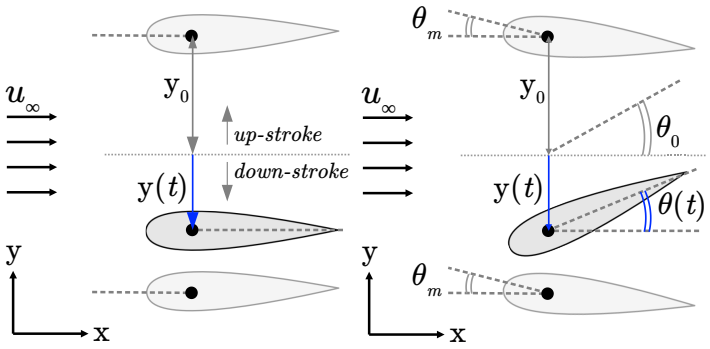


Figure 63: Smart bird manufactured by FESTO.



(a) Pure plunging (b) Plunging + pitching

Figure 64: Kinematics of a moving wing in cartesian coordinates.

y_0/c. Additionally, θ_m denotes the mean pitching angle, θ_0 is the pithing amplitude, and θ_s is the shift angle between the plunging and pitching motions. The pivot point/line is located at the center of pressure $x_{cp} = 0.25c$ to reduce the effect of aerodynamic loads on the pitching moment. Figure 64 shows a schematic of the flapping wing for two different types of motions. In Figure 64a, a pure plunging motion is perpendicular to the free-stream velocity. This motion only contains translation functions, including the up-stroke and down-stroke phases. In Figure 64b, the wing uses both translation and pitching functions. A combination of the up-stroke and down-stroke phases for one flapping cycle is given by $T^* = 1/f_a$.

To investigate the aerodynamic performance of a flapping wing, aerodynamic loads are non-dimensionalized in the form of lift and thrust coefficients

$$\begin{aligned} C_L &= \frac{F_L}{0.5\rho_\infty l_c u_\infty^2}, \\ C_T &= \frac{F_T}{0.5\rho_\infty l_c u_\infty^2}, \end{aligned} \tag{166}$$

where F_L and F_T are lift and thrust, respectively. Note that thrust at very low

reduced frequencies turns into drag. For 2D cases, the characteristic length is $l_c = c$, and for 3D cases, we change it to $l_c = cAR$, where AR denotes the aspect ratio of the wing. Additionally, the power consumed by the flapping wing is approximated by the power coefficient

$$C_P \approx \frac{F_L \dot{y}(t)}{0.5\rho_\infty l_c u_\infty^3}, \tag{167}$$

where $\dot{y}(t)$ denotes translational velocity of the wing in y direction. Here, we neglect the power consumed by pitching moment, since the pivot point is located at the center of pressure. Finally, using Eq. (166) and Eq. (167), the propulsive efficiency is given by

$$E_p = \frac{\overline{C_T}}{\overline{C_P}}. \tag{168}$$

10.2.2 Shape Optimization of a 2D Flapping Wing

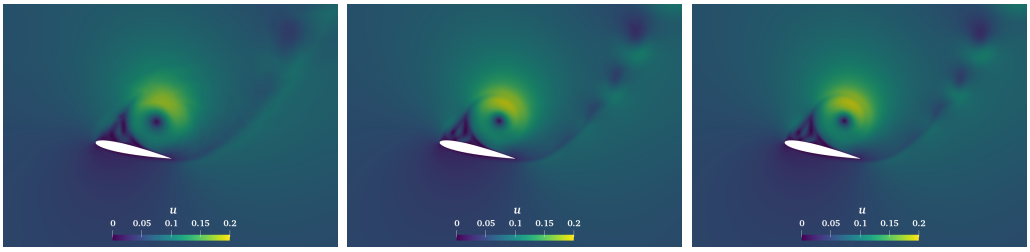
Flow Setup & Validation

An unstructured mesh is used to simulate flow past a 2D symmetric flapping wing. A NACA 0012 airfoil is placed $15c$ and $25c$ away from the upstream and downstream boundaries, respectively. Second-order BDF2 temporal scheme with $\Delta t = 0.01T^*$ is chosen to solve the discretized Navier-Stokes equations. In Table 10, the time-averaged aerodynamic results for different solution polynomial degrees, p_s, are compared with forces in Ref. [91]. In this problem, kinematic parameters are $y_0 = c$, $k_a = 1.41$, $T^* = 4.44c/u_\infty$, $\theta_m = 10^\circ$, $\theta_0 = 30^\circ$, and $\theta_s = 90^\circ$. Each simulation initially is run until $10T^*$, and then aerodynamic loads are time-averaged over the next $7T^*$. The present numerical simulations demonstrate that all polynomial solution degrees give appropriate results, close to those in Ref. [91]. Figure 65 compares

p_s	Order of accuracy	DOF	$\overline{C_L}$	$\overline{C_T}$	$C_{L,rms}$	$C_{D,rms}$
1	2^{rd}	9.89×10^4	1.7273	0.7178	2.8278	0.9459
2	3^{rd}	2.22×10^5	1.7629	0.7123	2.8405	0.9517
3	4^{th}	3.95×10^5	1.7409	0.7163	2.8332	0.9472
Ref. [91]	2^{th}	18.4×10^6	1.5507	0.7245	2.7743	0.9224

Table 10: Comparing the current results with those in Ref. [91].

velocity contours for different solution polynomial degrees. In Figure 65a, velocity contours for $p_s = 1$ indicate that large dissipation exists in wake of the wing, compared to those with $p_s = 2$. On the other hand, Figures 65b and 65c show identical



(a) $p_s = 1$ (b) $p_s = 2$ (c) $p_s = 3$

Figure 65: Comparing velocity contours for different solution polynomial degrees at the instant $t/T^* = 0.25$.

velocity contours with higher resolutions, indicating local dissipations in wake of the wing remain minimal for these polynomial solution degrees. Since simulation with $p_s = 2$ is computationally cheaper, compared to $p_s = 3$, it is decided to choose $p_s = 2$ for the rest of this study.

Optimization Setup

This section demonstrates the application of the proposed approach to unsteady aerodynamic optimization. We consider a 2D NACA 0012 flapping wing at $Re = 1000$, and $Re = 2400$, with a reduced frequency $k_a = 1.41$. The effect of geometrical constraints on the design is also considered. Four different test cases were selected, as shown in Table 11. The first two cases, "C1" and "C2", were performed at $Re = 1000$, and the last two cases, "C3" and "C4", at $Re = 2400$. Geometrical constraints in "C1" and "C2" only limit the minimum thickness. On the other hand, more geometrical constraints are added to "C3" and "C4" to control the maximum/minimum thickness of the wing at different sections alongside the chord line. Figure 66 displays

Case	Re	θ_m	θ_0	n_s	Number of constraints	Number of bounds
C1	1,000	10°	30°	11	24	28
C2	1,000	10°	10°	11	24	28
C3	2,400	10°	30°	11	30	28
C4	2,400	10°	10°	11	30	28

Table 11: Test cases for optimization of a flapping wing at $y_0 = c$, $k_a = 1.41$, and $\theta_s = 90°$.

the control points (CPs) assigned to define coordinates, and shape of a NACA 0012 airfoil, as the baseline design. In optimization, CP1 is frozen (i.e., stationary) for all

design cycles. Also, CP7, CP8, and CP9 are dependent on the coordinates of CP6 and CP10, which avoids low-quality mesh (i.e., highly skewed mesh, or sharp wall curvatures) during the optimization procedure. In addition to these control points, the pitching amplitude was added to design parameters. Therefore, the total number of design parameters in this optimization reaches $n_s = 11$.

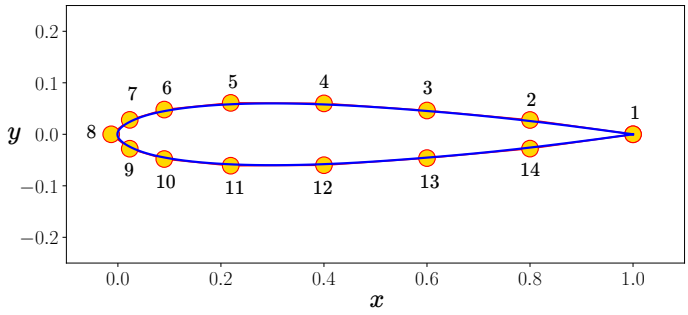
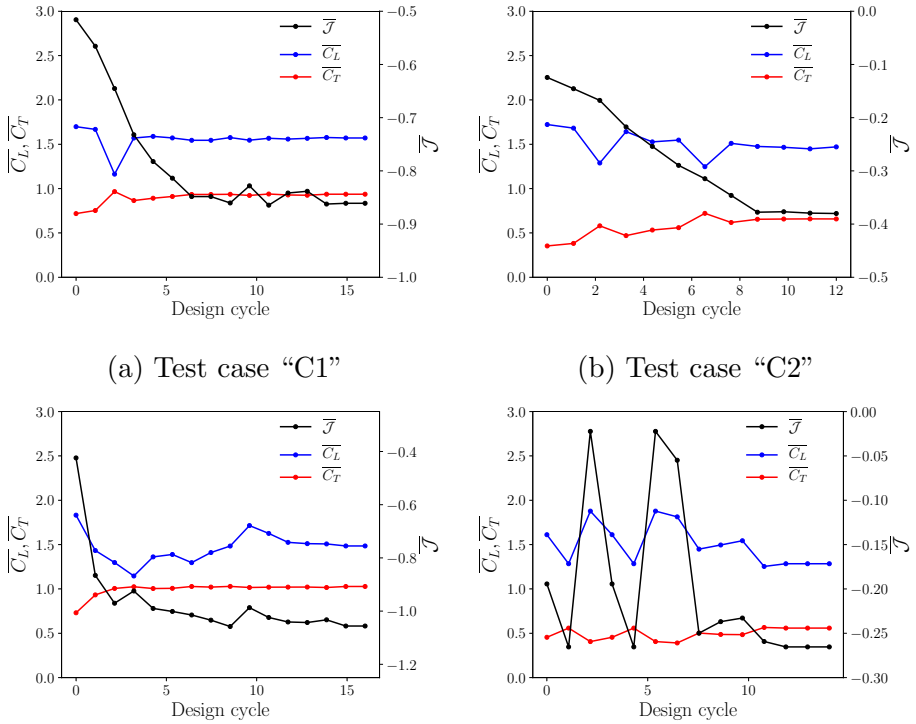


Figure 66: NACA 0012 coordinates defined by control points.

In general, the objective is to increase the time-averaged thrust coefficient. We also keep the time-averaged lift coefficient unchanged, since it balances with the gravitational force of an external body. Therefore, we define this optimization as

$$\begin{aligned}
&\underset{\tilde{\boldsymbol{q}}\in\mathbb{R}^r,\mathcal{S}\in\mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} = \frac{1}{T^*}\int_{2T^*}^{3T^*} -C_T^2 + \left(C_L - C_{L,target}\right)^2 dt, \\
&\text{subject to} \\
&\qquad \mathcal{X}(t,\mathcal{S}), \\
&\qquad \mathcal{R}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{X}, \mathcal{S}) = 0, \\
&\qquad \frac{d\mathcal{R}}{d\mathcal{S}}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{X}, \mathcal{S}) = 0, \\
&\qquad \mathbf{C}(\mathcal{X}, \mathcal{S}) \leq 0,
\end{aligned} \tag{169}$$

were the target lift coefficient is $C_{L,target} = 1.5$. This target lift guarantees additional improvement in the propulsive efficiency as $\overline{C_T}$ increases. In this optimization, the KKT system is solved using the BDF1 scheme. In the sampling step, the steady-state sensitivity function is solved over 10% of time intervals in $\Delta\mathbb{T}$ for each design parameter. Moreover, these design parameters were modified using Sequential Least-Squares Programming (SLSP) as an optimizer.



(a) Test case "C1"

(b) Test case "C2"

(c) Test case "C3"

(d) Test case "C4"

Figure 67: Progressive results of optimization for different test cases.

Optimization Performance

Figure 67 shows the results of optimization for each case. In general, the time-averaged objective function, $\overline{\mathcal{J}}$, decreases after each design cycle. However, we see that the objective function increases at some design cycles. This is because the design space is bounded, or limited, by geometrical constraints. Another reason is also attributed to fluid physics. Ref. [43] showed that flow past a 2D NACA 0012 airfoil at $R = 1000$ has the potential to be chaotic at high angles of attack in post-stall. Additionally, Ref. [45] considered chaoticity of fluid physics for the same airfoil, but at $Re = 2400$, and they confirmed that flow remains chaotic beyond stall. Here, we add more complexity to these problems by considering a moving wing in the flow field, yielding dominant chaoticity in the fluid dynamics. Therefore, fluctuations in the objective function are expected due to high uncertainties in these chaotic problems. In general, $\overline{C_T}$ increases progressively after each design cycle, which is the primary objective of this optimization. According to the constraint, given by the target lift coefficient, $\overline{C_L}$ changes significantly at the early stage of the design. Afterward, the

optimizer, with the help of sensitivities, try to compensate losses in $\overline{C_L}$, and lead it toward $C_{L,target}$. For all cases, except "C4", $\overline{C_L}$ is in an acceptable range. However, in "C4", the objective function undergoes large fluctuations due to limited routes in design space that is difficult to be found by the optimizer. It is believed that strict constraint functions associated with high uncertainties, are responsible for this deficiency in the design procedure. However, the optimizer provides the best possible design in the presence of these constraints.

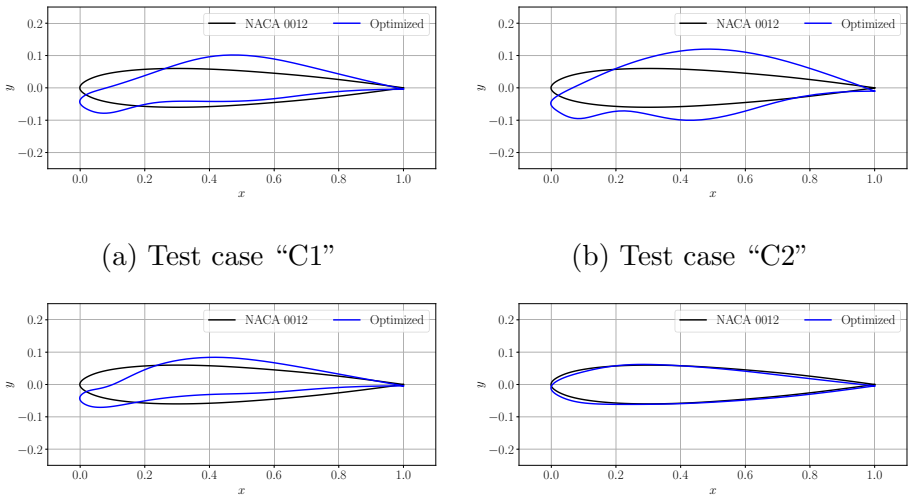
Overall results for all test cases are provided in Table 12. In "C1", 32.86% of improvement is achieved, leading to 34.4% of increment in the resulting propulsive efficiency. In "C3", similar performance is observed. In "C2", $\overline{C_T}$ is approximately doubled, yielding 92.55% and 79.42% of improvements in the thrust coefficient and propulsive efficiency, respectively. Although "C4" shows poor optimization performance than other test cases, 12.36% and 11.4% of increments in the thrust coefficient and propulsive efficiency are observed, respectively.

Case	Design (shape)	$\overline{C_L}$	$\overline{C_T}$	$\overline{C_P}$	E_p	$\Delta\overline{C_T}/\overline{C_T}$	$\Delta E_p/E_p$
C1	NACA 0012	1.695	0.7267	2.697	0.2694	-	-
	Optimized	1.605	0.9655	2.665	0.3621	32.86%	34.40%
C2	NACA 0012	1.733	0.3531	4.059	0.0870	-	-
	Optimized	1.666	0.6799	4.354	0.1561	92.55%	79.42%
C3	NACA 0012	1.765	0.7689	2.709	0.2837	-	-
	Optimized	1.531	1.0545	2.653	0.3973	37.14%	40.04%
C4	NACA 0012	1.624	0.4488	4.192	0.1070	-	-
	Optimized	1.542	0.5043	4.229	0.1192	12.36%	11.40%

Table 12: Comparing the propulsive performance of the flapping wing for the baseline and optimized designs.

Figure 68 shows the shape of both the baseline and optimized designs. A common agreement among all optimized designs is that the nose is deviated downward to produce a larger thrust coefficient. This modification decreases the effective angle of attack, and adds more camber to the wing. It also helps produce higher aerodynamic loads during the down-stroke phase. In "C1" and "C2" the optimized wings have more thickness approximately at x = $0.4c$. This is because there is no user-defined constraint, or bound, to limit the maximum thickness. Cambered wings create a larger Leading Edge Vortex (LEV) during both the up-stroke and down-stroke phases. The

LEV produced during the up-stroke phase yields a considerable reduction in the time-averaged lift coefficient, which is against the objective of this optimization. Therefore, the lower side of the wing becomes thicker to weaken this LEV during the up-stroke phase. In "C3" and "C4", geometrical constraints limit the maximum thickness of the wing and, hence, the optimizer performs differently according to these new constraints.



(a) Test case "C1"

(b) Test case "C2"

(c) Test case "C3"

(d) Test case "C4"

Figure 68: Comparing the shape of the baseline and optimized designs.

Flow Structures

Figures 69 and 70 show vorticity contours for different test cases. In Figure 69 (upper), the optimized wing delays stall during the down-stroke phase. This delay increases the aerodynamic loads by taking advantage of dynamic stall. At instant $t/T^* = 0.25$, primary LEV for the baseline design is maturely developed, while this LEV for the optimized wing is at the early stage of formation. The same situation is observed in Figure 69 (lower). At instant $t/T^* = 0.5$, a small set of vortices, shed from the trailing edge of the baseline wing, appears as a vortex sheet, which is often known as a reason for lift reduction during dynamic stall [49, 71]. However, the optimized wing breaks down this vortex sheet into a set of separated vortices, yielding larger lift. As shown in Figure 70 (upper), at instant $t/T^* = 0.25$, the LEV for the baseline design is fully developed. However, the optimized wing creates two LEVs simultaneously, one

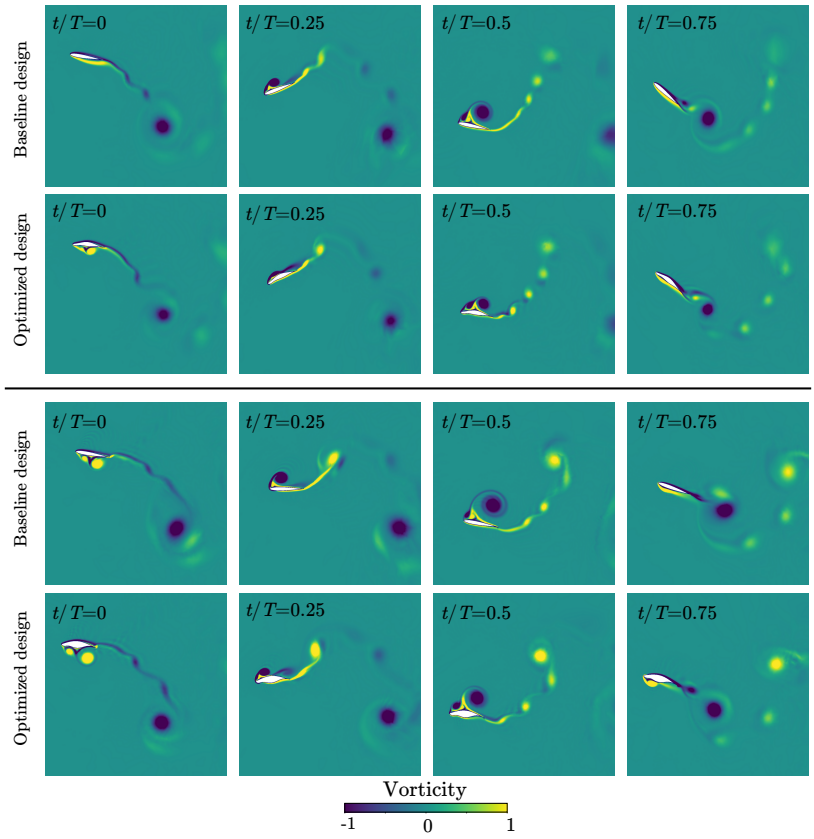


Figure 69: Vorticity contours at different instants: "C1" (upper), and "C2" (lower).

at the leading edge, and the other one at approximately x = 0.5c. These binary LEVs attach to the upper surface of the wing and, hence, delay stall, yielding higher lift production. At instant $t/T^* = 0.5$, the baseline wing is beyond stall, and the LEV is about to pinch off. However, at the same instant, the optimized wing produces larger lift due to delayed stall. Furthermore, this optimized design breaks down the vortex sheet into smaller separated vortices. Figure 70 (lower) presents the results of "C4". As it is shown, the optimized wing does not change the flow field notably. Only a few minor modifications in the optimized shape are responsible for higher lift production.

Figure 71 shows the instantaneous lift and trust coefficients during one flapping cycle. An important preference in optimizing the flapping wing is that the aerodynamic loads, both the lift and thrust coefficients, should increase during the down-stroke phase, and the lift coefficient should not decrease notably during the up-stroke phase. In "C1", the lift coefficient for the baseline and optimized wings have the same trend during one flapping cycle, as shown in Figure 71a. However, the optimized wing

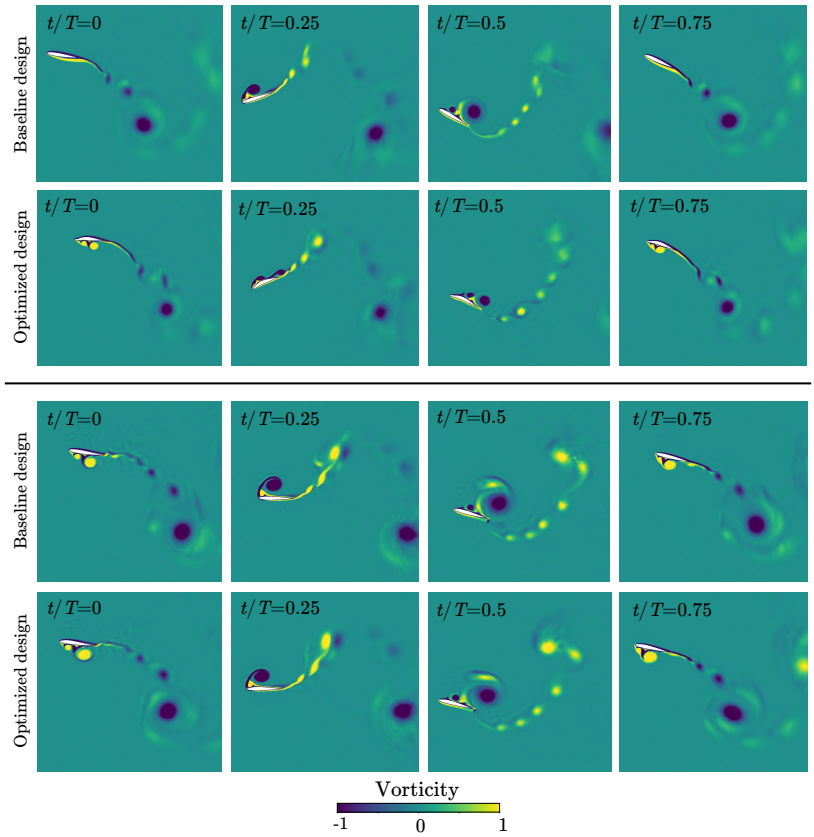
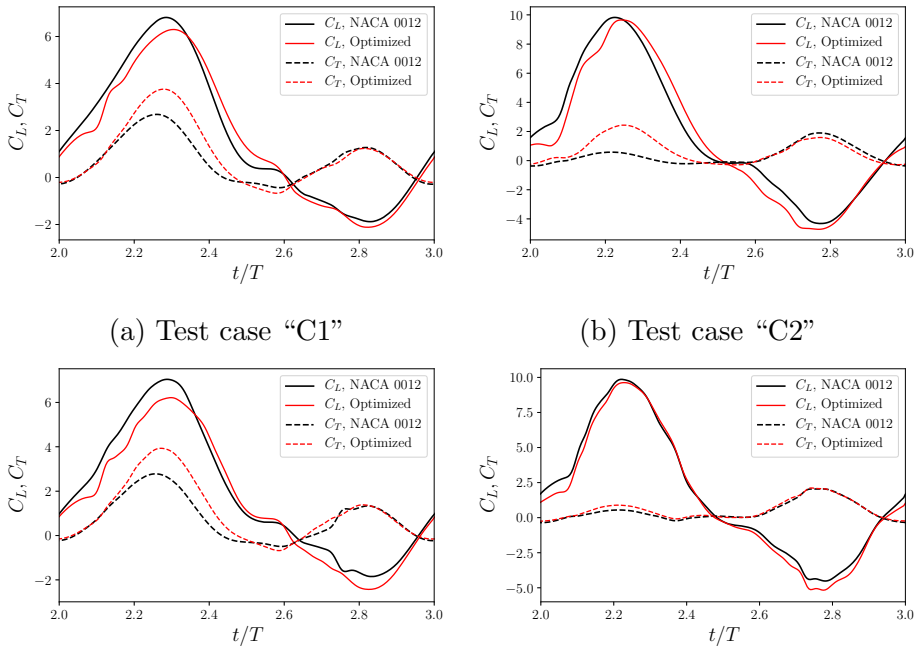


Figure 70: Vorticity contours at different instants: "C3" (upper), and "C4" (lower).

increases the thrust coefficient significantly during the down-stroke phase. In this case, we can see the impact of dynamic stall on the aerodynamic loads. The same observations exist in "C2", as shown in Figure 71b. It is worth pointing out that the baseline design has a smaller thrust coefficient during the down-stroke phase, while the optimized wing alleviates this deficiency. Figure 71c also dictates the same observations in Figure 71a. In "C4", the optimized wing could only improve the thrust coefficient slightly, as seen from Figure 71d.

According to these results, we showed that shape optimization could significantly improve the aerodynamic performance of a flapping wing. Although strict constraints governed the optimization procedure, the optimizer could search for better parameters in the design space, which is augmented by uncertainties. Additionally, strong nonlinearity in fluid physics makes these types of problems more challenging. In the end, we conclude that the proposed ROM-constrained optimization has a particular promise to tackle these kinds of complex and chaotic problems.



(a) Test case "C1"

(b) Test case "C2"

(c) Test case "C3"

(d) Test case "C4"

Figure 71: Instantaneous lift and thrust coefficients for both baseline and optimized designs.

10.2.3 Kinematics Optimization of a 3D Flapping Wing

Flow Setup & Validation

In this example, an unstructured mesh with hexahedral elements for a 3D symmetric wing is used. A NACA 0020 airfoil, with the aspect ratio of $AR = 0.45$, is placed $10c$ and $25c$ away from the upstream and downstream boundaries, respectively. The periodic boundary condition is also set for side-wall boundaries. The discretized Navier-Stokes equations are solved in time using a second-order BDF2 temporal scheme, with $\Delta t = \frac{1}{750}T^*$ as the time-step. For validation, we consider a pure plunging motion of the wing at $Re = 20,000$, $\mathrm{y}_0 = 0.5$, $k_a = 2$, and $\theta(t) = 0$. Table 13 shows the time-averaged thrust and power coefficients for the present simulations and those in Ref. [3]. These simulations were run until four flapping cycles, and then the time-averaged values are computed over the last three cycles. The results of the present simulations are provided with 99% of confidence intervals. As it is shown, the present results are in good agreement with those in Ref. [3]. Note that Ref. [3] used a laminar flow solver for a 2D NACA 0020 and, hence, some discrepancies were expected. Moreover,

the resulting values of the medium and fine meshes are close to each other, indicating the medium mesh can be used for the rest of this study.

Test	DOF	$\overline{C_T}$	$\overline{C_P}$
Coarse mesh	1.98×10^5	0.38 ± 0.06	0.85 ± 0.07
Medium mesh	5.9×10^5	0.42 ± 0.07	0.84 ± 0.09
Fine mesh	15.9×10^5	0.43 ± 0.06	0.84 ± 0.06
Ref. [3]	2.7×10^5	0.44	0.87

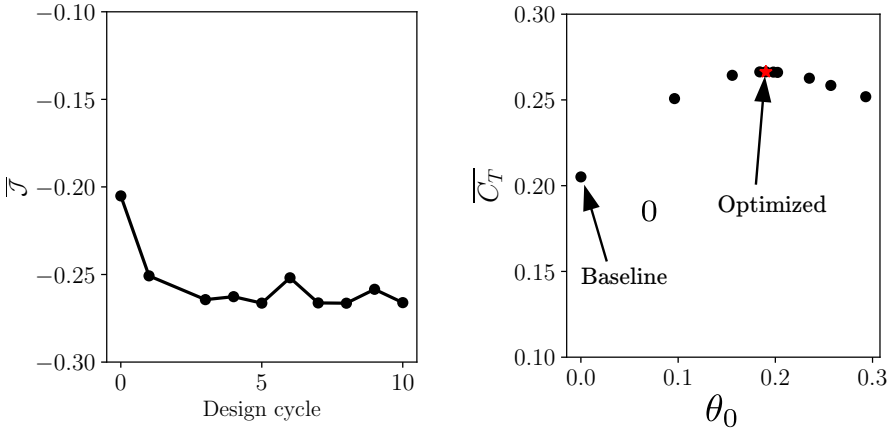
Table 13: Comparing the time-averaged thrust and power coefficients for the present simulations with those in Ref. [3].

Optimization Setup

Kinematics optimization of a 3D NACA 0020 wing at $Re = 2 \times 10^4$, $\mathrm{y}_0 = 0.5$, $k_a = 2$, $\theta_m = 0^\circ$, and $\theta_s = 90^\circ$ is considered. This optimization helps increase the time-averaged thrust coefficient of a pure plunging motion, $\theta_0 = 0^\circ$, by combining it with a pitching motion, $\theta_0 \neq 0^\circ$. Solving this problem is impossible by conventional PDE-constrained optimization due to the chaoticity of flow at high Reynolds numbers. Moreover, applying conventional LSS requires 1.5TB of memory for solving an extensive system of equations using parallel algorithms. Therefore, here we show the application of the proposed approach to optimization of this large-scale chaotic problem. In this case, we define this optimization as

$$
\begin{aligned}
&\underset{\tilde{\boldsymbol{q}} \in \mathbb{R}^r, \mathcal{S} \in \mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} = \frac{1}{T^*} \int_{2T^*}^{3T^*} -\big(C_T\big)^2 dt, \\
&\text{subject to} \\
&\qquad \mathcal{X}(t, \mathcal{S}), \\
&\qquad \mathcal{R}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{q}}, t, \mathcal{X}, \mathcal{S}) = 0, \\
&\qquad \frac{d\mathcal{R}}{d\mathcal{S}}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{q}}, t, \mathcal{X}, \mathcal{S}) = 0, \\
&\qquad \mathbf{C}(\mathcal{X}, \mathcal{S}) \leq 0,
\end{aligned}
\tag{170}
$$

where the third flapping cycle is considered for evaluating the aerodynamic performance. Therefore, we start optimizing the pitching angle with a set of bounds in design space. In the sampling step, 50% of time intervals in $\Delta\mathbb{T}$ is chosen to collect state vectors. Moreover, 5% and 25% of time intervals in the same time domain are selected to solve steady-state and unsteady sensitivity functions, respectively. Finally,



(a) Time-averaged objective function.

(b) Thrust coefficient versus pitching amplitude.

Figure 72: Optimization results for a flapping wing: The baseline design, which has only pure plunging motion, is optimized over ten design cycle, yeilding a combination of plunging and pitching motions with an optimum pitching angle.

the first eight dominant modes for each sensitivity dataset were kept to build the ROB. The same optimizer in the previous examples is used to perform optimization for ten design cycles.

Optimization Results

Figure 72 shows the optimization progress for ten design cycles. In Figure 72a, the time-averaged objective function decreases monotonically at the early stage, and then it starts fluctuating once the design parameter reaches around the optimum point. Since the sensitivity of the time-averaged objective function near this point is small, high uncertainty agitates this sensitivity value, yielding fluctuations in the objective function, or some poor approximation in design space. Figure 72b shows the time-averaged thrust coefficient versus the pitching angle, θ_0. The baseline design starts with pure plunging motion, $\theta_0 = 0°$, and finally, the optimized kinematics is obtained by $\theta_0 = 10.54°$. This optimized kinematics elevates the time-averaged thrust coefficient by 29.7%, which is noted by a red star at the peak. It is worth mentioning that sensitivity analysis using the present approach only requires 4MB of memory, which is achievable on a single processor. Figure 73 displays vorticity contours on a plane passing through the middle of the wing. It is shown that the optimized design

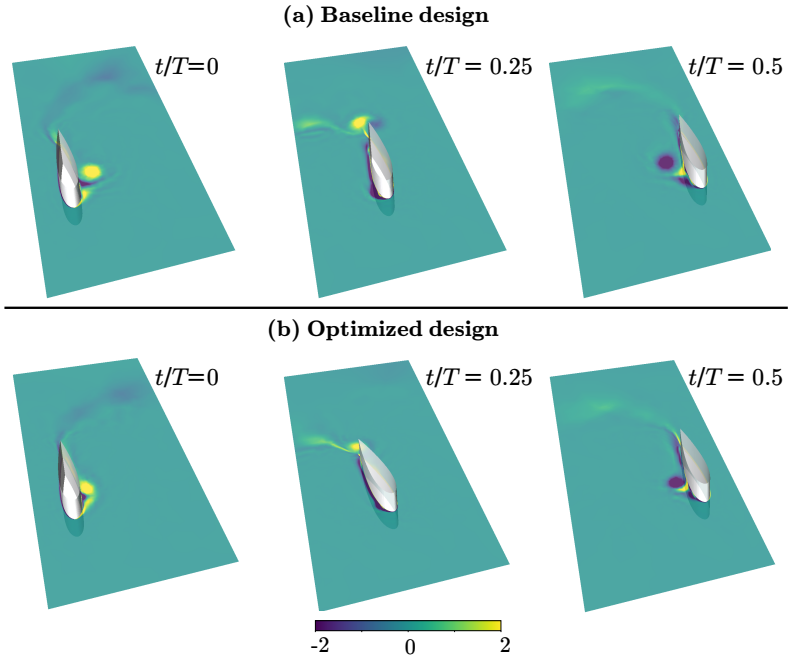


Figure 73: Vorticity contours for the baseline and optimized designs.

postpones stall, which leads to higher lift compared to the baseline design.

10.3 Remarks

In this study, we proposed a new approach for PDE-constrained optimization of nonlinear systems. The intuition of this study is that, instead of directly optimizing Full-Order Models (FOMs) in physical space, we transform the physical governing equations into an unphysical space, where the dynamics of the system evolve on manifolds. This allows us to optimize the evolutionary behaviour of dynamical systems in lower dimensions. Hence, in optimization problems, the optimizer is able to change the dynamical behaviour of the system, such that its effect in physical space minimizes the objective function. To this end, we developed a closure model in the form of a Reduced-Order Model (ROM). This closure model was explicitly derived from the FOM using the Least-Squares Petrov-Galerkin (LSPG) approach, and leverages the minimum-residual property over a discrete temporal domain. Using this ROM, we omitted low energy unstable modes, while maintaining the accuracy of the ROM. Additionally, sampling techniques were considered for building the trial and test basis functions, which shapes the manifolds in Hilbert space. This procedure is referred

to as a physics-constrained data-driven approach. The main feature of developing a ROM using this approach is that it provides derivatives of the FOM (i.e., Jacobian) in lower dimensions. Furthermore, sequential least-squares minimizations were used to solve this closure model, leading to a robust framework, specialized for unsteady optimization.

Shape optimization of a NACA 0012 airfoil in the presence of flow separation at $Re = 1000$ was then considered. It was shown that the proposed framework results in a significant improvement in the aerodynamic performance of the airfoil, with a drag reduction of approximately 20% observed at $\alpha_{eff} = 25°$. Unlike the reference case with the non-linear dynamical system, the results illustrate that the dynamical system of the optimized case exhibit a linear-like dynamical behaviour. The present approach was also applied to shape optimization of a 2D moving wing section at $Re = 2,400$ and reduced frequency $k = 1.41$, where the non-linear interaction of this wing with fluid introduces chaotic flow structures. It was shown that the proposed approach can significantly improve the thrust force and propulsive efficiency by 37% and 40%, respectively. Additionally, kinematic optimization of a 3D NACA 0020 wing at $Re = 2 \times 10^4$, $\mathrm{y}_0 = 0.5$, $k_a = 2$, and $\theta_s = 90°$ was considered. The optimized wing increasees the thurst force by about 29.6%. Moreover, sensitivity analysis using the present approach only requires 4MB of memory, while solving the same problem using the conventional LSS needs 1.5TB memory requirements.

Therefore, we expect that implementing the proposed optimization approach can be applied to large-scale non-linear problems, such as turbulent flows, and this approach significantly reduces computational cost and data storage requirements. Future work will focus on applying this approach to larger and more complicated turbulent flows. In the end, we conclude that the proposed approach can be applied to unsteady optimization of larger, chaotic, and more complicated fluid dynamics problems. Besides the performance, and robustness of the present approach, it reduces memory requirements significantly, making a particular promise for future engineering design and optimization. Although the examples considered in this study are simple canonical problems, our intention was to used them to demonstrate the feasibility of the present approach to optimization. We will advance the proposed approach to consider it for larger problems with more complicated flows in future work.

Chapter 11

Conclusions

11.1 Final Notes

Conventional sensitivity analysis often fails to compute time-averaged sensitivities of chaotic dynamical systems over long time periods. This issue arises from the fact that chaotic dynamical systems have at least one unstable mode with a positive Lyapunov exponent. Therefore, any small perturbation in the direction of this unstable mode grows exponentially in time, depending on the magnitude of the corresponding Lyapunov exponent. From a mathematical standpoint, the *shadowing lemma*, in the form of the LSS approach, can be used to compute accurate time-averaged sensitivity solutions of chaotic systems. However, this approach is computationally expensive for large-scale fluid mechanic problems (i.e., $10^4 \sim 10^5$ times more expensive than the primal PDE solvers). Although several efforts have been made to reduce the computational costs of LSS, these methods still suffer from high computational cost and memory requirements for high-fidelity simulations, such as LES and DNS.

In this thesis, we proposed a novel approach to compute sensitivity functions with significantly reduced computational cost. Unlike other ROM approaches, we explicitly derived a closure model, in the form of a ROM, from the high-dimensional governing equations. Then, we fully discretized those PDEs, and transformed them from physical space into an unphysical space (i.e., Hilbert space), such that lower-dimensional models are represented as OΔEs. This transformation approach applies a significant physical constraint to the developed closure model.

The underlying sets of equations were solved via a novel architecture. It is worth

mentioning that the underlying equations leverage the minimum-residual property using the Petrov-Galerkin approach, such as LSPG described in [29]. Consequently, we implemented the proposed approach within an in-house scientific software package called the OPtimization Toolkit for Highly NOn-linear Systems (OPTHiNOS), which is suitable for multicore architectures.

Additionally, the application of the gradient-free Mesh Adaptive Direct Search (MADS) method, as an optimizer, was investigated for the aerodynamic optimization of turbulent flows. To achieve a proper time-averaged objective function, a method was proposed to compute the sensitivity of the objective function with respect to the simulation time. Using this algorithm, MADS launches optimization tasks on CPU clusters. Then, each task executes corresponding CFD simulations on GPU clusters to reduce computational costs. Dynamic Polynomial Approximation (DPA) was also proposed to control the accuracy of each simulation at each optimization step, significantly reducing computational costs.

We found it essential to develop methods for gradient-free and gradient-based optimization problems, since one of them may have priority over another one. For example, when the number of design parameters is not very large, and the optimum point for design is demanded, MADS associated with DPA can be a good choice. However, when the number of design variables increases, gradient-based approaches are preferable. Finally, the developed approaches can be improved by further investigations, and we provide few comments as suggestions for future work in the next section.

11.2 Future Work

Different avenues for future work are as follows:

1. Dimensionality reduction was applied using Proper Orthogonal Decompositions (POD). These POD modes are usually taken into account for *projection-based* reduced-order modelling because of their orthogonality properties. In Galerkin projections, this orthogonality plays a primary role. These modes are also mathematically interpretable, but not physically. In other words, POD modes may not contain the dynamical features of a physical phenomenon. Therefore, one may investigate using other dimensionality techniques, such as Dynamic Mode

Decomposition (DMD), Koopman Mode Decomposition (KMD), or Spectral Proper Orthogonal Decomposition (SPOD), to develop more accurate models.

2. In this study, we developed the weak form of the FOM using the Petrov-Galerkin approach. Dimensionality reduction of this FOM is constrained by LSPG, which helps the model satisfy the minimal-residual property at all time intervals. The main disadvantage of the LSPG approach is that only implicit time integrators can be used. However, other approaches, such as the Adjoint Petrov-Galerkin method, can be applied to the weak form of a FOM with both implicit and explicit time integrators. Therefore, the effectiveness of new dimensionality reduction approaches should be explored.

3. Data collection and reconstruction for large-scale CFD problems remain computationally expensive. Applying dimensionality reduction to all Jacobian matrices over a finite time domain needs significant computational effort. Applying novel deep learning algorithms to approximate these Jacobians in lower-dimensional forms might be an efficient approach. The probability of sparsity in low-dimensional Jacobian matrices should be investigated using conventional, or new, black-box machine learning algorithms.

4. In this study, we proposed different algorithms used to shape manifolds. Other possible training strategies should be taken into account. Practical strategies should reduce the computational cost, and improve the accuracy of closure models in the optimization procedure.

5. To compute the sensitivity of a chaotic dynamical system, we need to solve the KKT system over a finite number of time intervals. Multiple Shooting LSS is another approach [19] to reduce the number of time intervals over which Jacobian matrices are computed. Applying this approach may reduce computational costs for large-scale problems. Therefore, as an alternative LSS solver, this method can be applied to large-scale LES and DNS problems.

6. The binary Dynamic Polynomial Approximation (DPA) method in the MADS was shown to be an efficient strategy for gradient-free optimization to control the accuracy of CFD simulation. This approach works like a controller, which

increases the resolution of the CFD solution once the updated design parameters are close to the optimal region. More advanced DPA strategies should be considered.

7. MADS can be combined with surrogate models to improve its prediction in the optimization process. This will help reduce the number of function evaluations in the design space and accelerate optimization performance. Additionally, surrogate models can find approximated derivatives and build a hybrid optimization method, leveraging gradient-free and gradient-based approaches.

8. In this project, we tried to work on problems with low Reynolds numbers to reduce computational costs and the complexity in design. Applying the sensitivity analysis and optimization to aerodynamic problems at high Reynolds numbers will be one of the future works that we plan to consider. This will help extend the application of the proposed approaches to real-world engineering problems.

Appendix A

Lyapunov Solver

Algorithm 1: Lyapunov solver for computing the first n_l^{th} leading LEs.

Input: $\boldsymbol{u}_0, m_k, T_s$, and n_l.
Output: Γ_i and $\mathfrak{L}_i$ for $i = 1, 2, ..., n_l$.
Build an empty matrix $\mathbf{S}_l \in \mathbb{R}^{n_l \times m_k}$.
Build a random matrix $\mathcal{U} \in \mathbb{R}^{n_u \times n_l}$.
$T^{(0)} \leftarrow 0$,
$\boldsymbol{u}(t^{(0)}) \leftarrow \boldsymbol{u}_0$,
$\mathbf{QR} \leftarrow \mathcal{U}$.
for $j = 0$ ***to*** m_k **do**
  $T^{(j+1)} \leftarrow T^{(j)} + T_s$,
  for $i = 1$ ***to*** n_l **do**
    for $t^{(n)} = T^{(j)}$ ***to*** $T^{(j+1)}$ **do**
      $\boldsymbol{v}_i(t^{(n-1)}) \leftarrow \mathbf{Q}[:, i]$,
      $\boldsymbol{u}(t^{(n)}) \leftarrow \underset{\mathbf{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\Delta \boldsymbol{u}(t^{(n)})}{\Delta t} + \nabla . \boldsymbol{f}(\boldsymbol{u}(t^{(n)}), t^{(n)}, \mathcal{S}) \right\|_2^2, \quad t^{(n)} \in \Delta\mathbb{T}$,
      Compute $\frac{\partial \nabla . \boldsymbol{f}(t^{(n)})}{\partial \boldsymbol{u}(t^{(n)})}$,
      $\boldsymbol{v}_i(t^{(n)}) \leftarrow \underset{\mathbf{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\Delta \boldsymbol{v}_i(t^{(n)})}{\Delta t} + \frac{\partial \nabla . \boldsymbol{f}(t^{(n)})}{\partial \boldsymbol{u}(t^{(n)})} \boldsymbol{v}_i(t^{(n)}) \right\|_2^2, \quad t^{(n)} \in \Delta\mathbb{T}$,
    end
    $\mathcal{U}[:, i] \leftarrow \boldsymbol{v}_i(T^{(j+1)})$.
  end
  $\mathbf{QR} \leftarrow \mathcal{U}$,
  for $i = 1$ ***to*** n_l **do**
    $\mathbf{S}_l[i, j] \leftarrow \frac{1}{T_s} \log |\mathbf{R}[i, i]|$.
  end
  $\left[\Gamma_1, \Gamma_2, \ldots, \Gamma_{n_l}\right] \leftarrow \mathbf{Q}$, and $\left[\mathfrak{L}_1, \mathfrak{L}_2, \ldots, \mathfrak{L}_{n_l}\right]^\top \leftarrow \frac{1}{m_k} \sum_{j=0}^{m_k} \mathbf{S}_l[:, j]$.
end

Appendix B

Developing Continuous Form of the Forward Tangent Equation

The continuous form of a closure model is derived from a FOM, as a set of high-dimensional ODEs. Those in the high-dimensional governing equations obtain all low-dimensional terms in the closure model. Afterward, er apply the LSS approach to obtain the KKT system in low-dimensional space. Finally, we apply discretization to this KKT system. The residual of the FOM is

$$\boldsymbol{r}(\boldsymbol{u}, t, \mathcal{S}) = \frac{\partial \boldsymbol{u}}{\partial t} + \nabla.\boldsymbol{f}(\boldsymbol{u}, t, \mathcal{S}), \quad \boldsymbol{u}(t = 0, \mathcal{S}) = \boldsymbol{u}_0, \tag{171}$$

where $\boldsymbol{r}$ and $\nabla.\boldsymbol{f} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$. Similarly, the residual of the closure model is given by

$$\mathcal{R}(\tilde{\boldsymbol{q}}, t, \mathcal{S}) = \frac{\partial \tilde{\boldsymbol{q}}}{\partial t} + \mathcal{F}(\tilde{\boldsymbol{q}}, t, \mathcal{S}), \quad \tilde{\boldsymbol{q}}(t = 0) = \tilde{\boldsymbol{q}}_0, \tag{172}$$

where $\mathcal{R}$ and $\mathcal{F} : \mathbb{R}^r \to \mathbb{R}^r$. In the ideal case, the residual for these FOM and ROM is zero. However, approximating the vector of the state in the FOM via dimensionality reduction, $\tilde{\boldsymbol{u}} \approx \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}$, will impose some non-zero value for residual. The Pertov-Galerkin projection minimizes the residual of this FOM in high-dimensional space, $\boldsymbol{r} \in \mathbb{R}^{n_u}$, by projecting this residual into the that of the ROM in low-dimensional space, $\mathcal{R} \in \mathbb{R}^r$, via the trial basis function. Therefore, we can write the Petrov-Galerkin projection as

$$\mathcal{R}(\tilde{\boldsymbol{q}}, t, \mathcal{S}) = \tilde{\boldsymbol{\Psi}}^\top \boldsymbol{r}(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}). \tag{173}$$

Substituting Eq. (171) into Eq. (173), yields

$$\mathcal{R} = \tilde{\boldsymbol{\Psi}}^\top \left(\frac{\partial(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}})}{\partial t} + \nabla.\boldsymbol{f}(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S}) \right), \quad \tilde{\boldsymbol{q}}(t = 0) = \tilde{\boldsymbol{q}}_0. \tag{174}$$

We can obtain the derivative of $\mathcal{R}$ with respect to $\mathcal{S}$ as

$$\begin{aligned}\frac{d\mathcal{R}}{d\mathcal{S}} = \frac{\partial\tilde{\boldsymbol{\Psi}}^\top}{\partial\mathcal{S}}\left(\frac{\partial(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}})}{\partial t} + \nabla.\boldsymbol{f}(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S})\right) + \tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t} + \tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{h}}}{\partial t}\right.\\ \left. + \frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\tilde{\boldsymbol{q}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\mathcal{S}}\right), \quad [\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{h}}](t=0) = [\tilde{\boldsymbol{q}}_0, \tilde{\boldsymbol{h}}_0],\end{aligned} \tag{175}$$

and using the test basis function, $\tilde{\boldsymbol{\Psi}} = \mathbf{A}^\top\mathbf{A}\left(\zeta_0\mathbf{I} + \Delta t\beta_0\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\right)\tilde{\boldsymbol{\Phi}} = \frac{\partial\boldsymbol{r}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}$, we obtain

$$\begin{aligned}\frac{d\mathcal{R}}{d\mathcal{S}} = \left(\frac{\partial\tilde{\boldsymbol{\Phi}}^\top}{\partial\mathcal{S}}\frac{\partial\boldsymbol{r}}{\partial\boldsymbol{u}}^\top + \tilde{\boldsymbol{\Phi}}^\top\frac{\partial}{\partial\mathcal{S}}\frac{\partial\boldsymbol{r}}{\partial\boldsymbol{u}}^\top\right)\left(\tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t} + \nabla.\boldsymbol{f}(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S})\right)\\ + \tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t} + \tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{h}}}{\partial t} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\tilde{\boldsymbol{q}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\mathcal{S}}\right), \quad [\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{h}}](t=0) = [\tilde{\boldsymbol{q}}_0, \tilde{\boldsymbol{h}}_0].\end{aligned} \tag{176}$$

Multiplying the right-hand side of Eq. (176) by $\left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}$

$$\begin{aligned}\frac{d\mathcal{R}}{d\mathcal{S}} = \left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}\left(\frac{\partial\tilde{\boldsymbol{\Phi}}^\top}{\partial\mathcal{S}}\frac{\partial\boldsymbol{r}}{\partial\boldsymbol{u}}^\top + \tilde{\boldsymbol{\Phi}}^\top\frac{\partial}{\partial\mathcal{S}}\frac{\partial\boldsymbol{r}}{\partial\boldsymbol{u}}^\top\right)\left(\tilde{\boldsymbol{\Phi}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t} + \nabla.\boldsymbol{f}(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}, t, \mathcal{S})\right)\\ + \frac{d\tilde{\boldsymbol{h}}}{dt} + \left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\mathcal{S}}\right)\\ + \left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\frac{\partial\tilde{\boldsymbol{q}}}{\partial t} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\frac{\partial\tilde{\boldsymbol{\Phi}}}{\partial\mathcal{S}}\tilde{\boldsymbol{q}}\right), \quad [\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{h}}](t=0) = [\tilde{\boldsymbol{q}}_0, \tilde{\boldsymbol{h}}_0].\end{aligned} \tag{177}$$

If we assume $\partial\tilde{\boldsymbol{\Phi}}/\partial\mathcal{S} = 0$, we then rewrite Eq. (177) as

$$\frac{d\mathcal{R}}{d\mathcal{S}} \approx \frac{\partial\tilde{\boldsymbol{h}}}{\partial t} + \left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\mathcal{S}}\right), \quad \tilde{\boldsymbol{h}}(t=0) = \tilde{\boldsymbol{h}}_0. \tag{178}$$

According to Eq. (44), where the LSS approach is applied to the forward sensitivity function, we can apply the same concept to Eq. (178)

$$\frac{d\mathcal{R}}{d\mathcal{S}} \approx \frac{\partial\tilde{\boldsymbol{h}}}{\partial t} + \left(\tilde{\boldsymbol{\Psi}}^\top\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^\top\left(\frac{\partial\nabla.\boldsymbol{f}}{\partial\boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial\nabla.\boldsymbol{f}}{\partial\mathcal{S}} + \eta\nabla.\boldsymbol{f}\right), \quad \tilde{\boldsymbol{h}}(t=0) = \tilde{\boldsymbol{h}}_0. \tag{179}$$

Therefore, the LSS problem to compute $\tilde{\boldsymbol{h}}$ can be represented as follows

$$\min_{\tilde{\boldsymbol{h}},\mathcal{E}} \frac{1}{2T}\int_0^\top \left(\tilde{\boldsymbol{h}}^\top\mathbf{K}\tilde{\boldsymbol{h}} + \alpha_{lss}^2\mathcal{E}^2\right)dt \quad \text{s.t.} \quad \lim_{\Delta\mathcal{S}\to\varepsilon}\frac{d\mathcal{R}}{d\mathcal{S}} = 0. \tag{180}$$

Using Lagrange multiplier, three systems of equations for this LSS problem is given by

$$\begin{aligned}
&\frac{\partial \tilde{\boldsymbol{h}}}{\partial t} + \left(\tilde{\boldsymbol{\Psi}}^{\top}\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^{\top}\left(\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{h}} + \frac{\partial \nabla.\boldsymbol{f}}{\partial \mathcal{S}} + \eta\nabla.\boldsymbol{f}\right),\\
&-\frac{\partial \lambda}{\partial t} + \tilde{\boldsymbol{\Phi}}^{\top}\frac{\partial \nabla.\boldsymbol{f}}{\partial \boldsymbol{u}}^{\top}\tilde{\boldsymbol{\Psi}}\left(\tilde{\boldsymbol{\Phi}}^{\top}\tilde{\boldsymbol{\Psi}}\right)^{-1}\lambda + \mathbf{K}\tilde{\boldsymbol{h}} = 0, \quad \lambda(0) = \lambda(T) = 0,\\
&\alpha_{lss}^{2}\mathcal{E} + \lambda^{\top}\left(\tilde{\boldsymbol{\Psi}}^{\top}\tilde{\boldsymbol{\Phi}}\right)^{-1}\tilde{\boldsymbol{\Psi}}^{\top}\nabla.\boldsymbol{f} = 0.
\end{aligned} \tag{181}$$

Appendix C

Singular Value Decomposition

Algorithm 2: POD function.

Input: $\mathbf{X}$, $\beta = \mathcal{O}(10^{-3})$, and $tol = \mathcal{O}(10^{-3})$.
Output: $\tilde{\mathbf{\Phi}}$, r, $\mathcal{Q}(\mathcal{S}_i)$, $\quad i = 1, 2, ..., n_d$.
Apply SVD such that $\mathbf{X} = \mathtt{USV}^\top$.
for $j = 1 \quad$ ***to*** $\quad n_u$ **do**
  $Error \leftarrow 0$,
  for $i = 1 \quad$ ***to*** $\quad n_d$ **do**
    $\mathbf{U}(\mathcal{S}_i) \leftarrow \mathbf{X}[:, (i-1) \times m_u : i \times m_u]$,
    $\mathbf{U}(\mathcal{S}_i)\left[\boldsymbol{u}^{(0)} - \overline{\boldsymbol{u}}, \boldsymbol{u}^{(1)} - \overline{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(m_u)} - \overline{\boldsymbol{u}}\right]$,
    $\boldsymbol{z} \leftarrow \mathtt{V}[: j, :]^\top$,
    $Error \leftarrow Error + \beta j + \left\| \sum_{n=0}^{m_u} \boldsymbol{u}^{(n)} - \overline{\boldsymbol{u}} - \mathtt{U}[:, : j]\mathtt{S}[: j, : j]\boldsymbol{z}^{(n)} \right\|_2^2$.
  end
  if $Error \leq tol$ **then**
    $r \leftarrow j$,
    $\tilde{\mathbf{\Phi}} \leftarrow \mathtt{U}[:, : r]$,
    for $i = 1 \quad$ ***to*** $\quad n_d$ **do**
      $\mathcal{Q}(\mathcal{S}_i) \leftarrow \mathtt{S}[: r, : r]\mathtt{V}[: r, (i-1) \times m_u : i \times m_u]^\top$,
    end
    Break (exit from the loop).
  end
end

Appendix D

Data Collection From Steady-State Functions

Algorithm 3: Data collection from steady sensitivity function.

Input: $\mathbf{U}(\mathcal{S}_1)$, $\mathbb{N}$, and $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_d}\}$.
Output: $\mathbf{V}_{ss}(\mathcal{S}_i)$.
foreach $\mathcal{S}_i$ ***in*** $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_d}\}$ **do**
  Build empty matrix: $\mathbf{V}_{ss}(\mathcal{S}_i) \in \mathbb{R}^{n_u \times m_p}$,
  $c \leftarrow 0$,
  for p ***in*** $\mathbb{N}$ **do**
    Compute $\frac{\partial \nabla_{\cdot} \boldsymbol{f}^{(p)}}{\partial \boldsymbol{u}^{(p)}}$, and $\frac{\partial \nabla_{\cdot} \boldsymbol{f}^{(p)}}{\partial \mathcal{S}_i}$ using $\mathbf{U}(\mathcal{S}_1)[:, p]$,
    $\mathbf{V}_{ss}(\mathcal{S}_i)[:, c] \leftarrow \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \nabla_{\cdot} \boldsymbol{f}^{(p)}}{\partial \boldsymbol{u}^{(p)}} \boldsymbol{z} + \frac{\partial \nabla_{\cdot} \boldsymbol{f}^{(p)}}{\partial \mathcal{S}_i} \right\|_2^2$,
    $c \leftarrow c + 1$.
  end
end

Appendix E

Data Collection From Unsteady Functions

First, we set up an algorithm for computing the initial condition using Algorithm 4. This step is important, since we need to reduce those errors in the initial condition to approximate the unsteady sensitivity solution over each T_s.

Algorithm 4: Initial condition approximation for unsteady data sampling using BDF scheme.

Input: $\mathbf{U}(\mathcal{S}_1)$, $\mathcal{S}_1$, k, and p (given by Algorithm 5).
Output: $\boldsymbol{v}^{(p-k)}$, and $\boldsymbol{v}^{(p-k+1)}$.
Approximate I.C.:
Compute $\frac{\partial \nabla . \boldsymbol{f}^{(p-k)}}{\partial \boldsymbol{u}^{(p-k)}}$, and $\frac{\partial \nabla . \boldsymbol{f}^{(p-k)}}{\partial \mathcal{S}_1}$,
$\boldsymbol{v}^{(p-k)} \leftarrow \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \nabla . \boldsymbol{f}^{(p-k)}}{\partial \boldsymbol{u}^{(p-k)}} \boldsymbol{z} + \frac{\partial \nabla . \boldsymbol{f}^{(p-k)}}{\partial \mathcal{S}_1} \right\|_2^2$.
if $k > 1$ **then**
  Compute $\frac{\partial \nabla . \boldsymbol{f}^{(p-k+1)}}{\partial \boldsymbol{u}^{(p-k+1)}}$, and $\frac{\partial \nabla . \boldsymbol{f}^{(p-k+1)}}{\partial \mathcal{S}_1}$,
  $\boldsymbol{v}^{(p-k+1)} \leftarrow \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \left(\mathcal{I} + \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(p-k+1)}}{\partial \boldsymbol{u}^{(p-k+1)}} \right) \boldsymbol{z} - \boldsymbol{v}^{(p-k)} - \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(p-k+1)}}{\partial \mathcal{S}_1} \right\|_2^2$.
  Note: a similar approach should be used for higher k-steps (i.e., BDF3).
end

Afterward, the main procedure is obtained via Algorithm 5 to approximate the shape of the manifolds in Hilbert space. Note that this algorithm only approximates those manifolds in Hilbert space, and it does not contain any practical magnitude for the sensitivities, since these results are normalized via QR factorization at every couple of iterations.

Algorithm 5: Data collection from unsteady sensitivity function using BDF scheme.

Input: $\mathbf{U}(\mathcal{S}_1)$, $m_k, n_u, \acute{m}_p, \Delta t, T_s, \mathcal{S}_1$, and $\mathbb{N}_{\acute{p}} = \{\xi_{\acute{p}}^{(1)}, \xi_{\acute{p}}^{(2)}, \ldots, \xi_{\acute{p}}^{(\acute{m}_p)}\}$.
Output: $\mathbf{V}_{us}$.
Build an empty matrix $\mathbf{V}_{us} \in \mathbb{R}^{n_u \times \acute{m}_p}$,
Devide $\mathbb{N}_{\acute{p}}$ into n_p parts: $\mathbb{N}_{\acute{p}} = \{\mathbb{N}_{\acute{p}}^1, \ldots, \mathbb{N}_{\acute{p}}^{n_p}\}$, where $n_p \ll \acute{m}_p$,
$c \leftarrow 0$,
$m_s \leftarrow \Delta t T_s$.
foreach $\mathbb{N}_{\acute{p}}^i$ ***in*** $\mathbb{N}_{\acute{p}}$ **do**
  for p ***in*** $\mathbb{N}_{\acute{p}}^i$ **do**
    Approximate initial condition using Algorithm 4
    Set $\acute{m}^{(0)} \leftarrow p$,
    for $l = 0$ ***to*** m_k **do**
      $\acute{m}^{(l+1)} \leftarrow \acute{m}^{(l)} + m_s$,
      for $n = \acute{m}^{(l)}$ ***to*** $\acute{m}^{(l+1)}$ **do**
        Compute $\frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$, and $\frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \mathcal{S}_1}$,
        $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \leftarrow \zeta_0 \mathcal{I} + \beta_0 \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$, $\quad \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n-j)}} \leftarrow \zeta_j$, $\quad \frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}_1} \leftarrow \beta_0 \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \mathcal{S}_1}$,
        $\boldsymbol{v}^{(n)} \leftarrow \underset{\boldsymbol{z} \in \mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \boldsymbol{z} + \sum_{j=1}^{k} \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n-j)}} \boldsymbol{v}^{(n-j)} + \frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}_1} \right\|_2^2$.
      end
      $\mathbf{V}_{us}[:, c] \leftarrow \boldsymbol{v}^{(\acute{m}^{(l+1)})}$,
      $c \leftarrow c + 1$,
      for $j = 0$ ***to*** k **do**
        $\mathbf{QR} \leftarrow \boldsymbol{v}^{(\acute{m}^{(l+1)}-k)}$,
        $\boldsymbol{v}^{(\acute{m}^{(l+1)}-k)} \leftarrow \mathbf{Q}$.
      end
    end
  end
end

Appendix F

Trial Basis Function & ROB

This algorithm shows the way we build the trial basis functions. In the beginning, those ROB are computed using the POD function, and then the results are arranged.

Algorithm 6: ROB and trial basis function.

Input: $\mathbf{U}$, $\mathbf{V}_{ss}$, $\mathbf{V}_{us}$, m_u, m_p, $\acute{m}_p$, r, and $\{\mathcal{S}_i, \mathcal{S}_2, \ldots, \mathcal{S}_{n_s}\}$.
Output: $\tilde{\mathbf{\Phi}}$.
foreach $\mathcal{S}_i$ ***in*** $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_d}\}$ **do**
  Build empty matrices: $\mathbf{U}(\mathcal{S}_i) \in \mathbb{R}^{n_u \times m_u}$, $\mathbf{V}_{ss}(\mathcal{S}_i) \in \mathbb{R}^{n_u \times m_p}$, and $\mathbf{V}_{us}(\mathcal{S}_i) \in \mathbb{R}^{n_u \times \acute{m}_p}$.
  Compute $\overline{\boldsymbol{u}} \leftarrow \frac{1}{m_u} \sum_{j=0}^{m_u} \boldsymbol{u}^{(j)}$, $\overline{\boldsymbol{v}_{ss}} \leftarrow \frac{1}{m_p} \sum_{j=0}^{m_p} \boldsymbol{v}_{ss}^{(j)}$, and $\overline{\boldsymbol{v}_{us}} \leftarrow \frac{1}{\acute{m}_p} \sum_{j=0}^{\acute{m}_p} \boldsymbol{v}_{us}^{(j)}$.
  for $j = 0$ ***to*** m_u **do**
    $\mathbf{U}(\mathcal{S}_i)[:, j] \leftarrow \boldsymbol{u}^{(j)} - \overline{\boldsymbol{u}}$,
  end
  for $j = 0$ ***to*** m_p **do**
    $\mathbf{V}_{ss}(\mathcal{S}_i)[:, j] \leftarrow \boldsymbol{v}_{ss}^{(j)} - \overline{\boldsymbol{v}_{ss}}$,
  end
end
for $j = 0$ ***to*** $\acute{m}_p$ **do**
  $\mathbf{V}_{us}(\mathcal{S}_1)[:, j] \leftarrow \boldsymbol{v}_{us}^{(j)} - \overline{\boldsymbol{v}_{us}}$,
end
$\mathbf{X}_{state} \leftarrow \Big[\mathbf{U}(\mathcal{S}_1), \mathbf{U}(\mathcal{S}_2), \ldots, \mathbf{U}(\mathcal{S}_{n_d})\Big]$,
$\mathbf{X}_{sens} \leftarrow \Big[\mathbf{V}_{ss}(\mathcal{S}_1), \mathbf{V}_{ss}(\mathcal{S}_2), \ldots, \mathbf{V}_{ss}(\mathcal{S}_{n_d})\Big]$,
$\acute{\mathbf{X}}_{sens} \leftarrow \Big[\mathbf{V}_{us}(\mathcal{S}_1)\Big]$,
$\tilde{\mathbf{\Phi}}_u \leftarrow \mathbf{POD}(\mathbf{X}_{state})$ using Algorithm 2,
$\tilde{\mathbf{\Phi}}_v \leftarrow \mathbf{POD}(\mathbf{X}_{sens})$ using Algorithm 2,
$\acute{\tilde{\mathbf{\Phi}}}_v \leftarrow \mathbf{POD}(\acute{\mathbf{X}}_{sens})$ using Algorithm 2,
$\tilde{\mathbf{\Phi}} \leftarrow \Big[\tilde{\mathbf{\Phi}}_u, \tilde{\mathbf{\Phi}}_v, \acute{\tilde{\mathbf{\Phi}}}_v\Big]$.

Appendix G

Sensitivity Analysis

Algorithm 7: Time-averaged sensitivity of the objective function.

Input: $\mathbf{U}(\mathcal{S}_1)$, and $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_d}\}$.
Output: $\overline{\frac{\Delta \mathcal{J}}{\Delta \mathcal{S}_1}} = [\overline{\frac{\Delta \mathcal{J}}{\Delta s_1}}, \ldots, \overline{\frac{\Delta \mathcal{J}}{\Delta s_{n_s}}}]^\top$.
Build $\mathbf{V}_{ss}(\mathcal{S}_i)$ using Algorithm 3 for $i = 1, 2, ..., n_d$,
Build $\mathbf{V}_{us}(\mathcal{S}_1)$ using Algorithm 5,
Build $\tilde{\mathbf{\Phi}}$ using Algorithm 6,
Solve ROM using Algorithm **??** for all $\tilde{\boldsymbol{q}}^{(n)}$, where $n = 0, 1, \ldots, m_m$.
for $n = 0$ ***to*** m_u **do**
  $\boldsymbol{u}^{(n)} \leftarrow \overline{\boldsymbol{u}} + \tilde{\mathbf{\Phi}} \tilde{\boldsymbol{q}}^{(n)}$,
  Compute $\frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$, and $\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$,
  $\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \leftarrow \zeta_0 \mathcal{I} + \beta_0 \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}$,
  $\tilde{\mathbf{\Psi}}^{(n)} \leftarrow \frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}} \tilde{\mathbf{\Phi}}$,
  $\frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n)}} \leftarrow \tilde{\mathbf{\Psi}}^{(n)\top} \tilde{\mathbf{\Psi}}^{(n)}$,
  for j ***in*** k **do**
    $\frac{\partial \mathcal{R}^{(n)}}{\partial \tilde{\boldsymbol{q}}^{(n-j)}} \leftarrow \zeta_{n-j} \tilde{\mathbf{\Psi}}^{(n)\top} \tilde{\mathbf{\Phi}}$,
  end
  for $l = 0$ ***in*** n_s **do**
    Compute $\frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial s_l}$, and $\frac{\partial \mathcal{J}^{(n)}}{\partial s_l}$,
    $\frac{\partial \boldsymbol{r}^{(n)}}{\partial s_l} \leftarrow \beta_0 \Delta t \frac{\partial \nabla . \boldsymbol{f}^{(n)}}{\partial s_l}$,
    $\frac{\partial \mathcal{R}^{(n)}}{\partial s_l} \leftarrow \tilde{\mathbf{\Psi}}^{(n)\top} \frac{\partial \boldsymbol{r}^{(n)}}{\partial s_l}$.
  end
end
Build $\mathcal{B}$ and $\mathcal{C}$ in KKT system.
for s ***in*** $\mathcal{S}_1$ **do**
  Build $\mathcal{D}$,
  Solve Eq. (53) for all $\lambda^{(n)}$, where $n = 0, 1, \ldots, m_m$,
  for $n = 0$ ***to*** m_u **do**
    Compute $\boldsymbol{h}^{(n)}$ using the second set of ODE in Eq. (136),
  end
  Compute $\overline{\frac{\Delta \mathcal{J}}{\Delta s}}$ using Eq. (138).
end

Appendix H

Sensitivity Analysis Using the Backward Sensitivity Function

H.1 Derivations

As mentioned in Section 6.2.2, in order to derive the backward sensitivity function, we need to get rid of the sensitivity solutions from the forward sensitivity function using the *adjoint method*. Recall Eq. (109), the high-dimensional backward sensitivity function is

$$-\frac{\partial \boldsymbol{w}}{\partial t}+\Big[\frac{\partial \nabla_{\cdot}\boldsymbol{f}}{\partial \boldsymbol{u}}\Big]^{\top}\boldsymbol{w}+\frac{1}{T}\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}=0,\quad \boldsymbol{w}(T)=0, \tag{182}$$

and with regard to Eq. (4), we discretize Eq. (182) using BDF scheme

$$\left(\zeta_0\mathcal{I}+\beta_0\Delta t\left[\frac{\partial \nabla_{\cdot}\boldsymbol{f}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\right)\boldsymbol{w}^{(n)}+\sum_{j=1}^{k}\zeta_j\boldsymbol{w}^{(n+j)}+\frac{1}{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}}=0,\quad \boldsymbol{w}^{(m_u)}=0, \tag{183}$$

where $\boldsymbol{w}^{(m_u)}=0$ is the initial condition for the backward propagation of Eq. (183) in discrete time. Since the present study relies on the least-squares minimization of the residual over $\Delta\mathbb{T}$, we rewrite Eq. (183) in the form of the residual as follows

$$\lim_{\delta\mathcal{S}\to 0}\hat{\boldsymbol{r}}^{(n)}:=\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\boldsymbol{w}^{(n)}+\sum_{j=1}^{k}\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n+j)}}\right]^{\top}\boldsymbol{w}^{(n+j)}+\frac{1}{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}}=0,\quad \boldsymbol{w}^{(m_u)}=0, \tag{184}$$

where $\hat{\boldsymbol{r}}^{(n)}\in\mathbb{R}^{n_u}$ is the residual of the backward sensitivity function. Let us assume that we build $\tilde{\boldsymbol{\Phi}}$ using a dataset that includes the *adjoint* solutions. Therefore, $\tilde{\boldsymbol{\Phi}}$ can

project the *adjoint* solution from $\mathscr{P}$ to $\mathscr{H}$, and vice versa. With this assumption, we can define $\boldsymbol{w}^{(n)} \approx \overline{\boldsymbol{w}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n)}$, and substitute it in Eq. (184)

$$\begin{aligned}\lim_{\delta\mathcal{S}\to 0} \hat{\boldsymbol{r}}^{(n)} := \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top} \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n)} + \sum_{j=1}^{k} \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n+j)}}\right]^{\top} \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n+j)} \\ + \Big(\sum_{j=1}^{k} \zeta_j \mathcal{I} + \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\Big)\overline{\boldsymbol{w}} + \frac{1}{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}} = 0, \quad \boldsymbol{w}^{(m_u)} = 0,\end{aligned} \tag{185}$$

where $\tilde{\boldsymbol{a}} \in \mathbb{R}^r$ is the generalized coordinates of the *adjoint* solution. From a mathematical perspective, we can define $\tilde{\boldsymbol{a}}$ alternatively as $\mathbb{F} : \tilde{\boldsymbol{a}} \times \mathbb{T} \to \tilde{\boldsymbol{a}}$, where $\mathbb{F}$ is the exact representation of the FOM in $\mathscr{H}$, and $\tilde{\boldsymbol{a}}$ denotes the evolutional dynamics of the backward sensitivity function in $\mathscr{H}$. Note that the third term in Eq. (185) relates the sensitivity of the residual to source term, similar to procedure used in Eq. (133). We use the Petrov-Galerking projection with the test basis function, $\hat{\tilde{\boldsymbol{\Psi}}} = \left[\frac{\partial \hat{\boldsymbol{r}}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\tilde{\boldsymbol{\Phi}}$, to project Eq. (185) onto $\mathscr{H}$

$$\begin{aligned}\lim_{\delta\mathcal{S}\to 0} \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\hat{\boldsymbol{r}}^{(n)} := \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top} \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n)} + \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\sum_{j=1}^{k} \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n+j)}}\right]^{\top} \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n+j)} \\ + \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\Big(\sum_{j=1}^{k} \zeta_j \mathcal{I} + \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\Big)\overline{\boldsymbol{w}} + \frac{1}{m_u}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}} = 0, \quad \tilde{\boldsymbol{a}}^{(m_u)} = 0,\end{aligned} \tag{186}$$

and we then rearrange it as follows

$$\begin{aligned}\lim_{\delta\mathcal{S}\to 0} \hat{\mathcal{R}}^{(n)} := \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)}\tilde{\boldsymbol{a}}^{(n)} + \sum_{j=1}^{k} \zeta_j \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n+j)} \\ + \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\Big(\sum_{j=1}^{k} \zeta_j \mathcal{I} + \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^{\top}\Big)\overline{\boldsymbol{w}} + \frac{1}{m_u}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}} = 0, \quad \tilde{\boldsymbol{a}}^{(m_u)} = 0,\end{aligned} \tag{187}$$

where $\hat{\mathcal{R}} \in \mathbb{R}^r$ is the residual of the ROM developed for the backward sensitivity function. Subsequently, Eq. (187) is converted into the LSS problem, in the form of

the Lagrange function

$$\begin{aligned}\mathcal{L} = \frac{1}{m_u}\sum_{n=0}^{m_u}\Bigg[\tilde{\boldsymbol{a}}^{(n)\top}\mathbf{K}\tilde{\boldsymbol{a}}^{(n)} + \left(\alpha_{lss}\mathcal{E}^{(n)}\right)^2\Bigg] + \sum_{n=0}^{m_u}\hat{\lambda}^{(n)\top}\Bigg[\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)}\tilde{\boldsymbol{a}}^{(n)} \\ + \sum_{j=1}^{k}\zeta_j\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n+j)} + \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\Big(\sum_{j=1}^{k}\zeta_j\mathcal{I} + \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^\top\Big)\overline{\boldsymbol{w}} \\ + \frac{1}{m_u}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}} - \mathcal{E}^{(n)}\sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n+j)}\Bigg],\end{aligned} \tag{188}$$

where $\hat{\lambda} \in \mathbb{R}^r$ is the Lagrange multiplier. The solution of this Lagrange function is obtained by applying the KKT condition to Eq. (188)

$$\begin{aligned}&\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)}\tilde{\boldsymbol{a}}^{(n)} + \sum_{j=1}^{k}\zeta_j\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n+j)} + \hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\Big(\sum_{j=1}^{k}\zeta_j\mathcal{I} + \left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \boldsymbol{u}^{(n)}}\right]^\top\Big)\overline{\boldsymbol{w}} \\ &\qquad\qquad + \frac{1}{m_u}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\frac{\partial \mathcal{J}^{(n)}}{\partial \boldsymbol{u}^{(n)}} = 0, \\ &\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\hat{\tilde{\boldsymbol{\Psi}}}^{(n)}\hat{\lambda}^{(n)} + \sum_{j=1}^{k}\zeta_j\hat{\tilde{\boldsymbol{\Psi}}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\hat{\lambda}^{(n-j)} + \mathbf{K}^\top\tilde{\boldsymbol{a}}^{(n)} = 0, \quad \hat{\lambda}^0 = \hat{\lambda}^{m_u} = 0, \\ &\alpha_{lss}^2\mathcal{E}^{(n)} - m_u\hat{\lambda}^{(n)\top}\sum_{j=0}^{k}\frac{\zeta_j}{\beta_0}\tilde{\boldsymbol{q}}^{(n+j)} = 0,\end{aligned} \tag{189}$$

where these three OΔEs are the backward sensitivity function in lower-dimensional space. In the end, similar to Eq. (160), the time-averaged sensitivity of the objective function with respect to $\mathcal{S}$ is approximated by

$$\frac{\overline{\Delta\mathcal{J}}}{\Delta\mathcal{S}} \approx \frac{1}{m_u}\sum_{n=0}^{m_u}\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial \mathcal{S}}\right]^\top(\overline{\boldsymbol{w}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{a}}^{(n)}) + \frac{1}{m_u}\sum_{n=0}^{m_u}\mathcal{E}^{(n)}\Big(\mathcal{J}^{(n)} - \overline{\mathcal{J}}\Big) + \frac{1}{m_u}\sum_{n=0}^{m_u}\frac{\partial \mathcal{J}^{(n)}}{\partial \mathcal{S}}. \tag{190}$$

It is worth mentioning that this lower-dimensional KKT solution of the backward sensitivity function in Eq. (189) is not obtained by Eq. (136). This method allows us to employ Eq. (51) for sensitivity analysis without any further modification. Although [18] developed the *adjoint* formulation of the LSS problem, there is no need to use it in this study. Otherwise, we derive a new formulation for the *adjoint* LSS problem, and develop another model for this lower-dimensional KKT solution.

H.2 Data Collection

The same procedure can be considered if the sensitivity analysis is based on the backward sensitivity approach (i.e., *adjoint method*). The steady-state backward sensitivity function can be solved for $\boldsymbol{w} \in \mathbb{R}^{n_u}$ at different time intervals, included in $\mathbb{N}_p(n) = \{\xi_p^{(1)}, \xi_p^{(2)}, \ldots, \xi_p^{(m_p)}\}$. Moreover, instead of solving the backward sensitivity function for different set of design parameter, we solve it for a list of the objective functions, $\mathcal{J}_i \in \{\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{n_w}\}$, where n_w denotes the number of the objective functions. Consequently, the matrix of the adjoint solutions can be represented as

$$\mathbf{W}_{ss}(\mathcal{J}_i) = \begin{bmatrix} | & | & & | \\ \Delta \boldsymbol{w}_{ss}^{(\mathbb{N}_p(1))}(\mathcal{J}_i) & \Delta \boldsymbol{w}_{ss}^{(\mathbb{N}_p(2))}(\mathcal{J}_i) & \ldots & \Delta \boldsymbol{w}_{ss}^{(\mathbb{N}_p(m_p))}(\mathcal{J}_i) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n_u \times m_p}, \quad (191)$$

where $\Delta \boldsymbol{w}_{ss}^{(n)} = \boldsymbol{w}_{ss}^{(n)} - \overline{\boldsymbol{w}_{ss}}$, and $\overline{\boldsymbol{w}_{ss}} \in \mathbb{R}^{n_u}$ shows a reference vector for the adjoint dataset, and it satisfies $\frac{\partial \overline{\boldsymbol{w}_{ss}}}{\partial t} = 0$. Therefore, the adjoint dataset is generated by

$$\mathbf{X}_{adj} = \bigoplus_{i=1}^{n_w} \mathbf{W}_{ss}(\mathcal{J}_i), \quad (192)$$

and subsequently, the ROB for the adjoint solution can be obtained by

$$\tilde{\boldsymbol{\Phi}}_w = \mathbf{POD}(\mathbf{X}_{adj}) \quad \text{and} \quad \boldsymbol{w}_{ss}(\mathcal{J}_i) = \tilde{\boldsymbol{\Phi}}_w \mathcal{Q}_w(\mathcal{J}_i), \quad i = 1, 2, ..., n_w, \quad (193)$$

where $\tilde{\boldsymbol{\Phi}}_w \in \mathbb{R}^{n_u \times r_{adj}}$. Additionally, we can define the generalized coordinates for the adjoint solution as

$$\mathcal{Q}_w = \begin{bmatrix} | & | & & | \\ \tilde{\boldsymbol{a}}^{(\mathbb{N}_p(1))} & \tilde{\boldsymbol{a}}^{(\mathbb{N}_p(2))} & \ldots & \tilde{\boldsymbol{a}}^{(\mathbb{N}_p(m_p))} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{r_{adj} \times m_p}. \quad (194)$$

The unsteady backward sensitivity function is solved, and the corresponding data are collected using the similar approach proposed in Algorithm 5. Therefore, we collect the snapshots of the unsteady adjoint solution as

$$\acute{\mathbf{W}}_{us}(\mathcal{J}_i) = \begin{bmatrix} | & | & & | \\ \Delta \boldsymbol{w}_{us}^{(\mathbb{N}_{\acute{p}}(1))}(\mathcal{J}_i) & \Delta \boldsymbol{w}_{us}^{(\mathbb{N}_{\acute{p}}(2))}(\mathcal{J}_i) & \ldots & \Delta \boldsymbol{w}_{us}^{(\mathbb{N}_{\acute{p}}(\acute{m}_p))}(\mathcal{J}_i) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n_u \times \acute{m}_p}, \quad (195)$$

where $\Delta \boldsymbol{w}_{us}^{(n)} = \boldsymbol{w}_{us}^{(n)} - \overline{\boldsymbol{w}_{us}}$, and $\acute{m}_p \in \mathbb{R}_+$ is the total number of samples. Additionally, $\boldsymbol{w}_{us} \in \mathbb{R}^{n_u}$ and $\overline{\boldsymbol{w}_{us}} \in \mathbb{R}^{n_u}$ are discrete unsteady adjoint solution, and its corresponding reference vector, respectively. Also, $\acute{\mathbf{W}}_{sens} \in \mathbb{R}^{n_u \times \acute{m}_p}$ represents the dataset of the unsteady adjoint solution in discrete form. We build this dataset using

$$\acute{\mathbf{X}}_{adj} = \bigoplus_{i=1}^{n_w} \mathbf{W}_{us}(\mathcal{J}_i). \tag{196}$$

The ROB is built by

$$\acute{\tilde{\mathbf{\Phi}}}_w = \mathbf{POD}(\acute{\mathbf{X}}_{adj}) \quad \text{and} \quad \mathbf{W}_{us}(\mathcal{J}_i) = \acute{\tilde{\mathbf{\Phi}}}_w \acute{\mathcal{Q}}_w(\mathcal{J}_i), \quad i = 1, 2, ..., n_w. \tag{197}$$

In this case, the generalized coordinates for Eq. (197) are

$$\acute{\mathcal{Q}}_w = \begin{bmatrix} | & | & & | \\ \acute{\tilde{\boldsymbol{a}}}^{(\mathbb{N}_{\acute{p}}(1))} & \acute{\tilde{\boldsymbol{a}}}^{(\mathbb{N}_{\acute{p}}(2))} & ... & \acute{\tilde{\boldsymbol{a}}}^{(\mathbb{N}_{\acute{p}}(\acute{m}_p))} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{\acute{r}_{adj} \times \acute{m}_p}, \quad \mathbb{N}_{\acute{p}}(n) = \{\xi_{\acute{p}}^{(1)}, \xi_{\acute{p}}^{(2)}, \ldots, \xi_{\acute{p}}^{(\acute{m}_p)}\}, \tag{198}$$

where $\acute{r}_{adj}$ is the rank of $\acute{\tilde{\mathbf{\Phi}}}_w$. Finally, the trail basis functions is defnied as a linear combination of the ROB

$$\tilde{\mathbf{\Phi}} = \tilde{\mathbf{\Phi}}_u \oplus \tilde{\mathbf{\Phi}}_w \oplus \acute{\tilde{\mathbf{\Phi}}}_w, \quad \text{and} \quad \tilde{\mathbf{\Phi}} \in \mathbb{R}^{n_u \times r} \tag{199}$$

where $r = r_{state} + r_{adj} + \acute{r}_{adj}$.

H.3 Discrete Backward Minimization Problem

For optimization problems with the backward sensitivity function, we discretize all set of equations similar to Section 8.2, but the minimization problems are different. In the first category, three minimization problems are found by

$$r_{state} \in \underset{z \in \mathbb{R}_+ \leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=0}^{m_u} \boldsymbol{u}^{(n)} - \overline{\boldsymbol{u}} - \tilde{\mathbf{\Phi}}_u \tilde{\boldsymbol{q}}^{(n)} \right\|_2^2 \right), \tag{200}$$

$$r_{sens} \in \underset{z \in \mathbb{R}_+ \leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=1}^{m_p} \boldsymbol{w}_{ss}^{(\mathbb{N}_p(n))} - \overline{\boldsymbol{w}_{ss}} - \tilde{\mathbf{\Phi}}_w \tilde{\boldsymbol{a}}^{(\mathbb{N}_p(n))} \right\|_2^2 \right), \tag{201}$$

$$\acute{r}_{sens} \in \underset{z \in \mathbb{R}_+ \leq n_u}{\operatorname{argmin}} \left(\beta z + \left\| \sum_{n=1}^{\acute{m}_p} \boldsymbol{w}_{us}^{(\mathbb{N}_{\acute{p}}(n))} - \overline{\boldsymbol{w}_{us}} - \acute{\tilde{\mathbf{\Phi}}}_w \acute{\tilde{\boldsymbol{a}}}^{(\mathbb{N}_{\acute{p}}(n))} \right\|_2^2 \right), \tag{202}$$

where we minimize ℓ^2-norm of the errors in the adjoint solution at selected time intervals. Moreover, the accumulation of these errors in the training dataset can be minimized according to

$$\boldsymbol{w}_{ss}^{(\mathbb{N}_p(n))} = \underset{\boldsymbol{z}\in\mathbb{R}^r}{\operatorname{argmin}} \left\| \left[\frac{\partial \boldsymbol{r}^{(\mathbb{N}_p(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_p(n))}}\right]^\top + \frac{\partial \mathcal{J}_i^{(\mathbb{N}_p(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_p(n))}} \right\|_2^2, \tag{203}$$

$$\boldsymbol{w}_{us}^{(\mathbb{N}_{\acute{p}}(n))} \in \underset{z\,\in\,\mathbb{R}^{n_u}}{\operatorname{argmin}} \left\| \left[\frac{\partial \boldsymbol{r}^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_{\acute{p}}(n))}}\right]^\top \boldsymbol{w}^{(\mathbb{N}_{\acute{p}}(n))} + \sum_{j=1}^{k} \left[\frac{\partial \boldsymbol{r}^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_{\acute{p}}(n)+j)}}\right]^\top \boldsymbol{w}^{(\mathbb{N}_{\acute{p}}(n)+j)} + \frac{1}{m_u}\frac{\partial \mathcal{J}_i^{(\mathbb{N}_{\acute{p}}(n))}}{\partial \boldsymbol{u}^{(\mathbb{N}_{\acute{p}}(n))}} \right\|_2^2, \tag{204}$$

$$\tilde{\boldsymbol{a}}_c, \boldsymbol{\Theta}_w \in \underset{\substack{z\,\in\,\mathbb{R}^{n_u},\\ \mathbf{Y}\,\in\,\mathbb{R}^{r\times r}}}{\operatorname{argmin}} \sum_{n'''=1}^{m_q} \left\| \boldsymbol{w}_{ss}^{(\mathbb{N}(n'''))} - \tilde{\boldsymbol{\Phi}}\mathbf{Y}\boldsymbol{z}^{(\mathbb{N}(n'''))} \right\|_2^2. \tag{205}$$

Finally, we can define the ROM-constrained optimization as

$$\begin{aligned}
&\underset{\boldsymbol{q}\in\mathbb{R}^r, \mathcal{S}\in\mathscr{D}}{\text{minimize}} \quad \overline{\mathcal{J}} \approx \frac{1}{m_u}\sum_{i=1}^{n_w}\sum_{n=0}^{m_u} \mathcal{J}_i^{(n)}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}),\\
&\text{subject to}\\
&\tilde{\boldsymbol{\Psi}}^{(n)\top}\tilde{\boldsymbol{\Phi}}\sum_{j=0}^{k}\zeta_j\tilde{\boldsymbol{q}}^{(n-j)} + \Delta t^{(n)}\beta_0\tilde{\boldsymbol{\Psi}}^{(n)\top}\nabla.\boldsymbol{f}^{(n)}\left(\overline{\boldsymbol{u}} + \boldsymbol{\Phi}\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}\right) = 0,\\
&\text{Apply Eq. (189)},\\
&\text{Apply Eq. (200), Eq. (201), Eq. (202)},\\
&\text{Apply Eq. (203), Eq. (204), Eq. (205)},\\
&\mathbf{C}(\overline{\boldsymbol{u}} + \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{q}}^{(n)}, t^{(n)}, \mathcal{S}) \le 0.
\end{aligned} \tag{206}$$

Despite deriving the backward sensitivity function from the forward sensitivity function, we obtained the backward sensitivity function directly from the high-dimensional forward sensitivity function, and then explicitly developed the ROM. This mathematical trick leads to avoid any further error propagation due to the dimensionality reduction throughout new derivations. Additionally, we can apply the same LSS, derived for the forward sensitivity analysis, to this backward sensitivity function. Consequently, the time-averaged sensitivities can be found by

$$\begin{aligned}
\frac{\overline{\Delta\mathcal{J}_i}}{\Delta\mathcal{S}} \approx \frac{1}{m_u}\sum_{n=0}^{m_u}\left[\frac{\partial \boldsymbol{r}^{(n)}}{\partial\mathcal{S}}\right]^\top(\overline{\boldsymbol{w}} + \tilde{\boldsymbol{\Phi}}\boldsymbol{\Theta}_w\tilde{\boldsymbol{a}}_c^{(n)}) + \frac{1}{m_u}\sum_{n=0}^{m_u}\mathcal{E}^{(n)}\left(\mathcal{J}_i^{(n)} - \overline{\mathcal{J}_i}\right)\\
+ \frac{1}{m_u}\sum_{n=0}^{m_u}\frac{\partial \mathcal{J}_i^{(n)}}{\partial\mathcal{S}}.
\end{aligned} \tag{207}$$

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA journal*, 46(7):1803–1813, 2008.

[3] MA Ashraf, J Young, and JCS Lai. Reynolds number, thickness and camber effects on flapping airfoil propulsion. *Journal of Fluids and structures*, 27(2):145–160, 2011.

[4] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[5] Charles Audet and Warren Hare. Derivative-free and blackbox optimization. 2017.

[6] Charles Audet, Gilles Savard, and Walid Zghal. A mesh adaptive direct search

algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3):545–556, 2010.

[7] S Bahrami, C Tribes, C Devals, TC Vu, and F Guibault. Multi-fidelity shape optimization of hydraulic turbine runner blades using a multi-objective mesh adaptive direct search algorithm. *Applied mathematical modelling*, 40(2):1650–1668, 2016.

[8] S Bahrami, C Tribes, S von Fellenberg, TC Vu, and F Guibault. Multi-fidelity design optimization of francis turbine runner blades. In *IOP Conference Series: Earth and Environmental Science*, volume 22, page 012029. IOP Publishing, 2014.

[9] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020.

[10] Andrea D Beck, Thomas Bolemann, David Flad, Hannes Frank, Gregor J Gassner, Florian Hindenlang, and Claus-Dieter Munz. High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. *International Journal for Numerical Methods in Fluids*, 76(8):522–548, 2014.

[11] Giancarlo Benettin, Luigi Galgani, Antonio Giorgilli, and Jean-Marie Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 1: Theory. *Meccanica*, 15(1):9–20, 1980.

[12] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[13] Michel Bergmann, C-H Bruneau, and Angelo Iollo. Enablers for robust POD models. *Journal of Computational Physics*, 228(2):516–538, 2009.

[14] Michel Bergmann and Laurent Cordier. Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models. *Journal of Computational Physics*, 227(16):7813–7840, 2008.

[15] Michel Bergmann, Laurent Cordier, and Jean-Pierre Brancher. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. *Physics of fluids*, 17(9):097101, 2005.

[16] Thomas R Bewley, Parviz Moin, and Roger Temam. DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447(2):179–225, 2001.

[17] Patrick J Blonigan, Pablo Fernandez, Scott M Murman, Qiqi Wang, Georgios Rigas, and Luca Magri. Toward a chaotic adjoint for LES. *arXiv preprint arXiv:1702.06809*, 2017.

[18] Patrick J Blonigan, Steven A Gomez, and Qiqi Wang. Least squares shadowing for sensitivity analysis of turbulent fluid flows. In *52nd Aerospace Sciences Meeting*, page 1426, 2014.

[19] Patrick J Blonigan and Qiqi Wang. Multiple shooting shadowing for sensitivity analysis of chaotic dynamical systems. *Journal of Computational Physics*, 354:447–475, 2018.

[20] Patrick J Blonigan, Qiqi Wang, Eric J Nielsen, and Boris Diskin. Least-squares shadowing sensitivity analysis of chaotic flow around a two-dimensional airfoil. *AIAA Journal*, 56(2):658–672, 2018.

[21] Patrick Joseph Blonigan. *Least Squares Shadowing for sensitivity analysis of large chaotic systems and fluid flows*. PhD thesis, Massachusetts Institute of Technology, 2016.

[22] C Bogani, MG Gasparo, and A Papini. Generalized pattern search methods for a class of nonsmooth optimization problems with structure. *Journal of Computational and Applied Mathematics*, 229(1):283–293, 2009.

[23] Ido Bright, Guang Lin, and J Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(12):127102, 2013.

[24] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[25] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[26] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288, 2008.

[27] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications. *AIAA journal*, 46(10):2520–2529, 2008.

[28] JS Cagnone, Brian C Vermeire, and S Nadarajah. A p-adaptive LCP formulation for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 233:324–338, 2013.

[29] Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.

[30] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[31] Kevin Carlberg, Youngsoo Choi, and Syuzanna Sargsyan. Conservative model reduction for finite-volume models. *Journal of Computational Physics*, 371:280–314, 2018.

[32] Kevin Carlberg and Charbel Farhat. A compact proper orthogonal decomposition basis for optimization-oriented reduced-order models. In *12th*

AIAA/ISSMO multidisciplinary analysis and optimization conference, page 5964, 2008.

[33] Kevin Carlberg and Charbel Farhat. A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. *International Journal for Numerical Methods in Engineering*, 86(3):381–402, 2011.

[34] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[35] Gerald Carrier, D Destarac, Antoine Dumont, Michael Meheut, Itham Salah El Din, Jacques Peter, Saloua Ben Khelil, J Brezillon, and M Pestana. Gradient-based aerodynamic optimization with the elsA software. In *52nd Aerospace Sciences Meeting*, page 0568, 2014.

[36] Oscar Castillo, Patricia Melin, and Witold Pedrycz. *Soft computing for hybrid intelligent systems*, volume 154. Springer, 2008.

[37] Tony F Chan and Michael K Ng. Galerkin projection methods for solving multiple linear systems. *SIAM Journal on Scientific Computing*, 21(3):836–850, 1999.

[38] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817, 2000.

[39] Ian D Coope and Christopher John Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, 11(4):859–869, 2001.

[40] Lisandro D Dalcin, Rodrigo R Paz, Pablo A Kler, and Alejandro Cosimo. Parallel distributed computing using Python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

[41] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. MPI for Python. *Journal of Parallel and Distributed Computing*, 65(9):1108 – 1115, 2005.

[42] Giovanni Di Ilio, Daniele Chiappini, Stefano Ubertini, Gino Bella, and Sauro Succi. Fluid flow around NACA 0012 airfoil at low-Reynolds numbers with hybrid lattice Boltzmann method. *Computers & Fluids*, 166:200–208, 2018.

[43] D Durante, E Rossi, and A Colagrossi. Bifurcations and chaos transition of the flow over an airfoil at low Reynolds number varying the angle of attack. *Communications in Nonlinear Science and Numerical Simulation*, 89:105285, 2020.

[44] BI Epureanu. A parametric analysis of reduced order models of viscous flows in turbomachinery. *Journal of fluids and structures*, 17(7):971–982, 2003.

[45] Pablo Fernandez and Qiqi Wang. Lyapunov spectrum of the separated flow around the NACA 0012 airfoil and its dependence on numerical discretization. *Journal of Computational Physics*, 350:453–469, 2017.

[46] Franco Flandoli and Bohdan Maslowski. Ergodicity of the 2-D Navier-Stokes equation under random perturbations. *Communications in mathematical physics*, 172(1):119–141, 1995.

[47] Haotian Gao and Mingjun Wei. Domain decomposition in pod-galerkin projection for flows with moving boundary. In *54th AIAA Aerospace Sciences Meeting*, page 1102, 2016.

[48] NR Gauger and J Brezillon. The continuous adjoint approach in aerodynamic shape optimization. In *MEGAFLOW-Numerical Flow Simulation for Aircraft Design*, pages 181–193. Springer, 2005.

[49] Kobra Gharali and David A Johnson. Dynamic stall simulation of a pitching airfoil under unsteady freestream velocity. *Journal of Fluids and Structures*, 42:228–244, 2013.

[50] Francesco Ginelli, Pietro Poggi, Alessio Turchi, Hugues Chaté, Roberto Livi, and Antonio Politi. Characterizing dynamics with covariant Lyapunov vectors. *Physical review letters*, 99(13):130601, 2007.

[51] David E Goldberg. *Genetic Algorithms*. Pearson Education India, 2006.

[52] Muralikrishnan Gopalakrishnan Meena, Kunihiko Taira, and Keisuke Asai. Airfoil-wake modification with gurney flap at low reynolds number. *AIAA Journal*, 56(4):1348–1359, 2017.

[53] Crina Grosan, Ajith Abraham, and Monica Nicoara. Search optimization using hybrid particle sub-swarms and evolutionary algorithms. *International Journal of Simulation Systems, Science & Technology*, 6(10):60–79, 2005.

[54] Max D Gunzburger. *Perspectives in flow control and optimization*, volume 5. Siam, 2003.

[55] Martin Hairer and Jonathan C Mattingly. Ergodicity of the 2D Navier-Stokes equations with degenerate stochastic forcing. *Annals of Mathematics*, pages 993–1032, 2006.

[56] Rania Hassan, Babak Cohanim, Olivier De Weck, and Gerhard Venter. A comparison of particle swarm optimization and the Genetic Algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, page 1897, 2005.

[57] Alexander Hay, Jeffrey T Borggaard, and Dominique Pelletier. Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition. *Journal of Fluid Mechanics*, 629:41–72, 2009.

[58] Ping He, Charles A Mader, Joaquim RRA Martins, and Kevin J Maki. DAfoam: An open-source adjoint framework for multidisciplinary design optimization with openFOAM. *AIAA Journal*, 58(3):1304–1319, 2020.

[59] Vicente Hernandez, Jose E Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):351–362, 2005.

[60] Michael Hinze and Ulrich Matthes. Model order reduction for networks of ode and PDE systems. In *IFIP Conference on System Modeling and Optimization*, pages 92–101. Springer, 2011.

[61] Cheng Huang, Karthik Duraisamy, and Charles Merkle. Challenges in reduced order modeling of reacting flows. In *2018 Joint Propulsion Conference*, page 4675, 2018.

[62] Cheng Huang, Karthik Duraisamy, and Charles Merkle. Investigations and improvement of robustness of reduced-order models of reacting flow. In *AIAA Scitech 2019 Forum*, page 2012, 2019.

[63] Rong F Huang and Chih L Lin. Vortex shedding and shear-layer instability of wing at low-Reynolds numbers. *AIAA journal*, 33(8):1398–1403, 1995.

[64] DBP Huynh and AT Patera. Reduced basis approximation and a posteriori error estimation for stress intensity factors. *International Journal for Numerical Methods in Engineering*, 72(10):1219–1259, 2007.

[65] Hung T Huynh. A flux reconstruction approach to high-order schemes including discontinuous galerkin methods. In *18th AIAA Computational Fluid Dynamics Conference*, page 4079, 2007.

[66] Osamu Inoue and Nozomu Hatakeyama. Sound generation by a two-dimensional circular cylinder in a uniform flow. *Journal of Fluid Mechanics*, 471:285, 2002.

[67] Antony Jameson. Aerodynamic design via control theory. *Journal of scientific computing*, 3(3):233–260, 1988.

[68] Antony Jameson and Sangho Kim. Reduction of the adjoint gradient formula for aerodynamic shape optimization problems. *AIAA journal*, 41(11):2114–2129, 2003.

[69] J-N Juang and Hideto Suzuki. An eigensystem realization algorithm in frequency domain for modal parameter identification. 1988.

[70] Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5):620–627, 1985.

[71] Hamid Reza Karbasian and Kyung Chun Kim. Numerical investigations on flow structure and behavior of vortices in the dynamic stall of an oscillating pitching hydrofoil. *Ocean Engineering*, 127:200–211, 2016.

[72] Anatole Katok and Boris Hasselblatt. *Introduction to the modern theory of dynamical systems*, volume 54. Cambridge university press, 1997.

[73] Gaetan Kerschen, Jean-claude Golinval, Alexander F Vakakis, and Lawrence A Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics*, 41(1-3):147–169, 2005.

[74] Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

[75] L Kourentis and E Konstantinidis. Uncovering large-scale coherent structures in natural and forced turbulent wakes by combining PIV, pod, and ftle. *Experiments in fluids*, 52(3):749–763, 2012.

[76] Karl Kunisch and Stefan Volkwein. Proper orthogonal decomposition for optimality systems. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 42(1):1–23, 2008.

[77] Dilek Funda Kurtulus. On the unsteady behavior of the flow around NACA 0012 airfoil with steady external conditions at Re= 1000. *International journal of micro air vehicles*, 7(3):301–326, 2015.

[78] Toni Lassila and Gianluigi Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1583–1592, 2010.

[79] François-Xavier Le Dimet and Olivier Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus A: Dynamic Meteorology and Oceanography*, 38(2):97–110, 1986.

[80] Julia Ling and J Templeton. Evaluation of machine learning algorithms for prediction of regions of high Reynolds Averaged Navier Stokes uncertainty. *Physics of Fluids*, 27(8):085103, 2015.

[81] Yan Liu, Kailun Li, Jiazhong Zhang, Hang Wang, and Liguang Liu. Numerical bifurcation analysis of static stall of airfoil and dynamic stall under unsteady perturbation. *Communications in Nonlinear Science and Numerical Simulation*, 17(8):3427–3434, 2012.

[82] Hugo FS Lui and William R Wolf. Construction of reduced order models for fluid flows using deep feedforward neural networks. *arXiv preprint arXiv:1903.05206*, 2019.

[83] JiaQi Luo, JunTao Xiong, and Feng Liu. Aerodynamic design optimization by using a continuous adjoint method. *Science China Physics, Mechanics & Astronomy*, 57(7):1363–1375, 2014.

[84] Zhoujie Lyu, Gaetan KW Kenway, and Joaquim RRA Martins. Aerodynamic shape optimization investigations of the common research model wing benchmark. *AIAA journal*, 53(4):968–985, 2015.

[85] Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670, 2012.

[86] Alison L Marsden, Meng Wang, JE Dennis, and Parviz Moin. Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation. *Journal of Fluid Mechanics*, 572:13–36, 2007.

[87] Joaquim RRA 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, 2005.

[88] R Moarref, MR Jovanović, JA Tropp, AS Sharma, and BJ McKeon. A low-order decomposition of turbulent channel flow via resolvent analysis and convex optimization. *Physics of Fluids*, 26(5):051701, 2014.

[89] M. Mohebujjaman, L.G. Rebholz, and T. Iliescu. Physically constrained data-driven correction for reduced-order modeling of fluid flows. *International Journal for Numerical Methods in Fluids*, 89(3):103–122, Oct 2018.

[90] Bruce Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE transactions on automatic control*, 26(1):17–32, 1981.

[91] M. Moriche, O. Flores, and M. García-Villalba. On the aerodynamic forces on heaving and pitching airfoils at low Reynolds number. *Journal of Fluid Mechanics*, 828:395–423, 2017.

[92] C Mullis and RA Roberts. Synthesis of minimum roundoff noise fixed point digital filters. *IEEE Transactions on Circuits and Systems*, 23(9):551–562, 1976.

[93] Takaaki Murata, Kai Fukami, and Koji Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882, 2020.

[94] Siva Nadarajah and Antony Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, page 667, 2000.

[95] Aditya G Nair, Chi-An Yeh, Eurika Kaiser, Bernd R Noack, Steven L Brunton, and Kunihiko Taira. Cluster-based feedback control of turbulent post-stall separated flows. *Journal of Fluid Mechanics*, 875:345–375, 2019.

[96] Habib N Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual review of fluid mechanics*, 41:35–52, 2009.

[97] Daniel A Nelson and Gustaaf B Jacobs. High-order visualization of three-dimensional lagrangian coherent structures with DG-FTLE. *Computers & Fluids*, 139:197–215, 2016.

[98] James C Newman III, Arthur C Taylor III, Richard W Barnwell, Perry A Newman, and Gene J-W Hou. Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *Journal of Aircraft*, 36(1):87–96, 1999.

[99] Angxiu Ni and Qiqi Wang. Sensitivity analysis on chaotic dynamical systems by non-intrusive least squares shadowing (NILSS). *Journal of Computational Physics*, 347:56–77, 2017.

[100] Angxiu Ni, Qiqi Wang, Pablo Fernandez, and Chaitanya Talnikar. Sensitivity analysis on chaotic dynamical systems by finite difference non-intrusive least squares shadowing (FD-NILSS). *Journal of Computational Physics*, 394:615–631, 2019.

[101] S Andrew Ning and Ilan Kroo. Multidisciplinary considerations in the design of wings and wing tip devices. *Journal of aircraft*, 47(2):534–543, 2010.

[102] Bernd R Noack, Konstantin Afanasiev, MAREK MORZYŃSKI, Gilead Tadmor, and Frank Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.

[103] Bernd R Noack, Marek Morzynski, and Gilead Tadmor. *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media, 2011.

[104] IB Oliveira and AT Patera. Reduced-basis techniques for rapid reliable optimization of systems described by affinely parametrized coercive elliptic partial differential equations. *Optimization and Engineering*, 8(1):43–65, 2007.

[105] Carlos E Orozco and ON Ghattas. Massively parallel aerodynamic shape optimization. *Computing Systems in Engineering*, 3(1-4):311–320, 1992.

[106] Valery Iustinovich Oseledets. A multiplicative ergodic theorem: Characteristic Lyapunov exponents of dynamical systems. *Trudy Moskovskogo Matematicheskogo Obshchestva*, 19:179–210, 1968.

[107] Mohagna J Pandya and Oktay Baysal. Gradient-based aerodynamic shape optimization using alternating direction implicit method. *Journal of aircraft*, 34(3):346–352, 1997.

[108] Eric J. Parish, Christopher Wentland, and Karthik Duraisamy. The adjoint Petrov-Galerkin method for non-linear model reduction, 2018.

[109] Eric J Parish, Christopher R Wentland, and Karthik Duraisamy. The Adjoint Petrov–Galerkin method for non-linear model reduction. *Computer Methods in Applied Mechanics and Engineering*, 365:112991, 2020.

[110] Sina Piramoon and Mohammad A Ayoubi. An eigensystem realization algorithm for modal parameter identification of a vertical-shaft high-speed centrifugal machine. In *ASME International Mechanical Engineering Congress and Exposition*, volume 84546, page V07AT07A032. American Society of Mechanical Engineers, 2020.

[111] Olivier Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1):97–110, 1974.

[112] Daniel J Poole, Christian B Allen, and Thomas Rendall. Comparison of local and global constrained aerodynamic shape optimization. In *32nd AIAA Applied Aerodynamics Conference*, page 3223, 2014.

[113] C Praveen and R Duvigneau. Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design. *Computer Methods in Applied Mechanics and Engineering*, 198(9-12):1087–1096, 2009.

[114] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[115] María-Luisa Rapún and José M Vega. Reduced order models based on local pod plus galerkin projection. *Journal of Computational Physics*, 229(8):3046–3063, 2010.

[116] SS Ravindran. Reduced-order adaptive controllers for fluid flows using POD. *Journal of scientific computing*, 15(4):457–478, 2000.

[117] Marco E Rosti, Mohammad Omidyeganeh, and Alfredo Pinelli. Direct numerical simulation of the flow around an aerofoil in ramp-up motion. *Physics of Fluids*, 28(2):025106, 2016.

[118] Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[119] Clarence W Rowley and Scott TM Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[120] Clarence W Rowley, Igor Mezi, Shervin Bagheri, Philipp Schlatter, DANS HENNINGSON, et al. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641(1):115–127, 2009.

[121] Gianluigi Rozza and Andrea Manzoni. Model order reduction by geometrical parametrization for shape optimization in computational fluid dynamics. In

Proceedings of the ECCOMAS CFD, V European Conference on Computational Fluid Dynamics, pages 1–20, 2010.

[122] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[123] Lothar M Schmitt. Theory of Genetic Algorithms II: Models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *Theoretical Computer Science*, 310(1-3):181–231, 2004.

[124] Michael S Selig. *Summary of low speed airfoil data*. SOARTECH publications, 1995.

[125] Yayun Shi, Charles A Mader, Sicheng He, Gustavo LO Halila, and Joaquim RRA Martins. Natural laminar-flow airfoil optimization design using a discrete adjoint approach. *AIAA Journal*, 58(11):4702–4722, 2020.

[126] SN Sivanandam and SN Deepa. Genetic Algorithms. In *Introduction to Genetic Algorithms*, pages 15–37. Springer, 2008.

[127] Shaun N Skinner and Hossein Zare-Behtash. State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62:933–962, 2018.

[128] Jeffrey Slotnick, Abdollah Khodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie, and Dimitri Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. 2014.

[129] Jaroslaw Sobieszczanski-Sobieski and Raphael T Haftka. Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

[130] Gregg M Streuber and David W Zingg. Evaluating the risk of local optima in aerodynamic shape optimization. *AIAA Journal*, pages 1–13, 2020.

[131] M Sudharsan, Steven L Brunton, and James J Riley. Lagrangian coherent structures and inertial particle dynamics. *Physical Review E*, 93(3):033108, 2016.

[132] Kunihiko Taira, Steven L Brunton, Scott TM Dawson, Clarence W Rowley, Tim Colonius, Beverley J McKeon, Oliver T Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S Ukeiley. Modal analysis of fluid flows: An overview. *AIAA Journal*, 55(12):4013–4041, 2017.

[133] Kunihiko Taira, Maziar S Hemati, Steven L Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott TM Dawson, and Chi-An Yeh. Modal analysis of fluid flows: Applications and outlook. *AIAA journal*, 58(3):998–1022, 2020.

[134] Alejandra Uranga, Per-Olof Persson, Mark Drela, and Jaime Peraire. Implicit large eddy simulation of transitional flows over airfoils and wings. In *19th AIAA Computational Fluid Dynamics*, page 4131. 2009.

[135] Brian C Vermeire and Siavash Hedayati Nasab. Accelerated implicit-explicit Runge-Kutta schemes for locally stiff systems. *Journal of Computational Physics*, page 110022, 2020.

[136] Brian C Vermeire, Freddie D Witherden, and Peter E Vincent. On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools. *Journal of Computational Physics*, 334:497–521, 2017.

[137] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[138] Sean Wakayama and Ilan Kroo. Subsonic wing planform design using multidisciplinary optimization. *Journal of Aircraft*, 32(4):746–753, 1995.

[139] 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.

[140] Yuan Yuan Wang, Bin Qian Zhang, and Ying Chun Chen. Robust airfoil optimization based on improved particle swarm optimization method. *Applied Mathematics and Mechanics*, 32(10):1245, 2011.

[141] Freddie D Witherden, Antony M Farrington, and Peter E Vincent. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014.

[142] Tobias F Wunderlich. Multidisciplinary wing optimization of commercial aircraft with consideration of static aeroelasticity. *CEAS Aeronautical Journal*, 6(3):407–427, 2015.

[143] Zhaoke Xu and Jian Xia. Aerodynamic optimization based on continuous adjoint method for a flexible wing. *International Journal of Aerospace Engineering*, 2016, 2016.

[144] Nail Yamaleev, Boris Diskin, and Eric Nielsen. Adjoint-based methodology for time-dependent optimization. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 5857, 2008.

[145] Wataru Yamazaki, Kisa Matsushima, and Kazuhiro Nakahashi. Aerodynamic design optimization using the drag-decomposition method. *AIAA journal*, 46(5):1096–1106, 2008.

[146] Weiguang Yang. *Surgical design for the Fontan procedure using computational fluid dynamics and derivative-free optimization*. PhD thesis, UC San Diego, 2012.

[147] Serhiy Yarusevych, Pierre E Sullivan, and John G Kawall. On vortex shedding from an airfoil in low-Reynolds-number flows. *Journal of Fluid Mechanics*, 632:245, 2009.

[148] Matthew J Zahr and Charbel Farhat. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, 2015.

[149] Matthew J Zahr and P-O Persson. An adjoint method for a high-order discretization of deforming domain conservation laws for optimization of flow problems. *Journal of Computational Physics*, 326:516–543, 2016.

[150] Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on Particle Swarm Optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.

[151] David W Zingg, Marian Nemec, and Thomas H Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics*, 17(1-2):103–126, 2008.