Numerical methods for control and optimisation of fluid flows

Serena Costanzo


SORBONNE UNIVERSITÉ

École doctorale Sciences Mécaniques, Acoustique, Électronique et Robotique

THÈSE DE DOCTORAT
SPÉCIALITÉ MÉCANIQUE DES FLUIDES

Numerical methods for control and optimisation of fluid flows

---

Presentée par **Serena COSTANZO**

pour obtenir le grade de Docteur de Sorbonne Université

Sous la direction de Taraneh SAYADI, Chargée de Recherche
Pascal FREY, Professeur

à l'Institut Jean Le Rond ∂'Alembert, Sorbonne Université, CNRS, UMR 7190

Soutenue le 22/06/2022 devant le jury composé de:

Angelo IOLLO	Professeur	Rapporteur
Kunihiko TAIRA	Professeur	Rapporteur
Paola CINNELLA	Professeur	Examinatrice
Lutz LESSHAFFT	Directeur de Recherche	Examinateur
Taraneh SAYADI	Chargée de Recherche	Directrice de Thèse
Pascal FREY	Professeur	Co-directeur de Thèse
Anca BELME	Maître de conférences	Invitée

"Non chi comincia, ma quel che persevera"
Leonardo da Vinci

Abstract

Advances in computing power have made high-fidelity numerical simulations of complex flows possible. However, due to the high computational cost associated with CFD simulations of unsteady flows, the use of accurate high-fidelity simulations makes the application of state-of-the-art control strategies ever more challenging. In this thesis, we propose different solutions to reduce the computational cost of the optimization problem, allowing the use of control strategies on the basis of detailed simulations, or combined with data-driven low-fidelity simulations, leading to a multi-fidelity approach. Control of high-fidelity simulations is performed using adjoint equations, which provide the fastest path to solution. However the sequential nature of adjoint-based optimization methods leads to an increase in time to solution. To reduce this time, we propose an algorithm allowing the parallelization on the time domain of the direct-adjoint problem. To enable the use of low-fidelity simulations in the optimization problem, we investigate dimensional reduction via Petrov-Galerkin projection methods combined with orthogonal interpolation on the Grassmann manifold and its tangent space. Finally a dynamics identification strategy based on system identification and clustering is proposed to identify the dominant dynamics existing in a flow and their mutual interactions. Insight on the underlying dynamics in a flow could play an important role in the choice of the optimization strategy and the possibility to use reduced-order models. All the methods are validated on two dimensional incompressible fluid flows, and implemented in a numerical Navier-Stokes solver with immersed boundaries.

Keywords: computational fluid dynamics; optimisation algorithms; adjoint-based methods; reduced order models; system identification; time parallelization.

Résumé

L'utilisation de simulations numériques haute-fidélité d'écoulements complexes a été rendue possible par les récentes avancées en matière de puissance de calcul. Cependant, le coût élevé de calcul, associé aux simulations CFD d'écoulements instationnaires de haute fidélité, rend l'application de stratégies de contrôle de pointe de plus en plus difficile. Dans cette thèse, nous proposons différentes méthodes de réduction du coût de calcul des procédures d'optimisation permettant d'utiliser des stratégies de contrôle sur la base de simulations détaillées, ou combinées avec des simulations basse-fidélité pilotées par des données, conduisant à une approche multi-fidélité. Le contrôle des simulations haute-fidélité est effectué à l'aide d'équations adjointes qui permettent de déterminer le chemin le plus direct vers la solution. Cependant, la séquentialité des méthodes adjointes entraîne une augmentation du temps de résolution. Afin de réduire ce temps, nous proposons un algorithme de parallélisation temporelle du problème direct-adjoint. De plus, l'utilisation de simulations de basse-fidélité est rendue possible par la réduction dimensionnelle via des méthodes de projection de Petrov-Galerkin combinées à une interpolation sur le manifold de Grassmann et son espace tangent. Enfin, une stratégie d'identification des dynamiques basée sur la localisation et le regroupement de systèmes est proposée. La compréhension des dynamiques sous-jacentes aux écoulements peut faciliter le choix de stratégies d'optimisation et d'utilisation de modèles d'ordre réduit. Toutes ces méthodes ont été mises en place dans un solveur numérique Navier-Stokes bidimensionnel à frontières immergées.

Mots clés: computational fluid dynamics; algorithmes d'optimisation; méthode adjointes; modelès d'ordre réduit; identification des systèmes; parallélisation temporelle.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Control and optimization strategies . . . 3
  - 1.2.1 Gradient-based optimization . . . 3
  - 1.2.2 Derivative-free optimization . . . 8
  - 1.2.3 Multi-fidelity optimization . . . 10
  - 1.2.4 Data-driven model reduction and identification of inherent flow dynamics: . . . 13
- 1.3 Outline of the thesis . . . 14

2 Adjoint-based optimization **17**
- 2.1 Introduction . . . 17
- 2.2 Optimisation problems . . . 18
- 2.3 Adjoint-based algorithms . . . 19
  - 2.3.1 Continuous versus discrete adjoint equations . . . 22
- 2.4 Governing equations and numerical framework . . . 23
  - 2.4.1 Discrete adjoint Navier-Stokes equations . . . 25
- 2.5 Application to flow configurations . . . 25
  - 2.5.1 Drag reduction – steady actuation . . . 26
  - 2.5.2 Total pressure loss – unsteady actuation . . . 27
- 2.6 Conclusions . . . 34

3 Parallel in time algorithm of adjoint-based techniques **35**
- 3.1 Introduction . . . 35
- 3.2 State-of-the-art and background . . . 36
  - 3.2.1 ParaExp algorithm . . . 36
  - 3.2.2 Hybrid serial-direct-parallel-adjoint algorithm . . . 38
- 3.3 Exponential time integrators . . . 43
  - 3.3.1 Projection-based methods . . . 44
  - 3.3.2 Polynomial interpolation based methods . . . 45
  - 3.3.3 Projection on Navier-Stokes equations . . . 45
- 3.4 Results . . . 46
  - 3.4.1 Temporal Energy Growth – examining the exponential time integrator . . . 46
  - 3.4.2 Drag reduction – steady actuation . . . 48
  - 3.4.3 Total pressure loss – unsteady actuation . . . 51
- 3.5 Conclusions . . . 54

4 Model Reduction - projection based methods **55**
4.1 Introduction . . . 55
4.2 Projection-based model reduction . . . 57
4.3 Snapshot proper-orthogonal-decomposition . . . 58
4.4 Variable parameter ROM - Grassmann manifold . . . 61
4.5 Galerkin and Petrov-Galerkin projection . . . 66
4.6 Gauss-Newton with approximated tensors . . . 68
4.6.1 Least-squares Petrov-Galerkin projection . . . 68
4.6.2 Tensors approximation and hyper-reduction . . . 69
4.7 Results . . . 71
4.7.1 Reduced-order model - GNAT . . . 71
4.7.2 Variable Parameter Gauss-Newton with Approximated Tensors (GNAT) . . . 75
4.8 Conclusions . . . 81

5 Data-driven identification of inherent flow dynamics **83**
5.1 Introduction . . . 83
5.2 Equation identification - SINDy algorithm . . . 84
5.3 SINDy applied to the Navier-Stokes equation . . . 86
5.4 Clustering algorithms . . . 88
5.4.1 Networks . . . 89
5.4.2 Community detection - Leicht-Newman network . . . 91
5.4.3 Stochastic model of turbulent axisymmetric wakes behind a bluff body . . . 95
5.5 Combining regression and clustering procedures for dynamical identification . . . 100
5.6 Conclusions . . . 104

6 Conclusions and perspectives **107**

List of Figures

1.1 Effect of forcing at the nozzle inlet, for a jet in cross flow set-up, relevant for the injection process. (a) unforced jet, (b) jet forced with sine wave excitation, (c) jet forced with square wave excitation. Image reproduced from [109]. 3
1.2 Graphical representation of the checkpointing procedure. Image reproduced from [193]. 5
1.3 An overview over important contributions to time parallel methods. Image reproduced from [75]. 6
1.4 (a) convex function; (b) nonconvex function. For nonconvex functions, local minima make it challenging for gradient-based optimization. Image reproduced from [27]. 7
1.5 Von Kármán vortex street generated by the Rishiri Island in Hokkaido, Japan (top: photograph from NASA, 2001; STS-100); cylinder wake at a low Reynolds number (bottom). Image reproduced from [196]. 12

2.1 Schematics of the two different possible paths to derive the discrete adjoint problem. Orange arrows: differentiate-then-discretize (or continuous) approach; Blue arrows: discretize-then-differentiate (or discrete) approach. 22
2.2 Flow around a cylinder at $Re = 200$. (a) Vorticity contours and immersed boundary points. (b) Schematic of Lagrangian points (blue) versus the background Cartesian grid. 26
2.3 Drag coefficient C_d, with the red line signifying the location where actuation starts, $t = 41s$. 27
2.4 The resulting optimal actuation. (a) Profile of the boundary velocities; (b) Profile of the boundary velocities (red) projected on the cylinder surface as a reference (black). 28
2.5 The resulting cost functional. Drag coefficient C_d without the actuation (black) and with the optimal control (red). 28
2.6 Schematic of a rotor of an axial compressor, and the numerical domain in the red box with one blade (periodic boundary conditions on the horizontal axis simulate a series of blades). The upper boundary moves with a constant velocity and the dynamic roughness is added on its profile. 29
2.7 Evolution of the average pressure loss in time, comparison of the base case (black line), and the case with actuation (red line). 31
2.8 Vorticity profile and pressure distribution for $t = 0.35$: (a) uncontrolled vorticity, (b) controlled vorticity, (c) uncontrolled pressure, (d) controlled pressure. 32

2.9 Vorticity profile and pressure distribution for $t = 0.77$: (a) uncontrolled vorticity, (b) controlled vorticity, (c) uncontrolled pressure, (d) controlled pressure. 33

3.1 Overlapping time decomposition of an initial-value problem into four inhomogeneous problems with zero initial guess (solid red curves) and four homogeneous problems (dashed blue curves), the latter are exponentially propagated. The solution of the original problem is obtained by summation of all these curves [73]. 38

3.2 A schematic of a direct-adjoint loop with three processors, using a parallel-in-time procedure. Solid lines follow the forward evolution of the direct equation (top) and backward integration of the inhomogeneous adjoint equations on the three processors (bottom). Once the direct and inhomogeneous adjoint equations are solved, the direct solution is communicated (red lines) to each processor to initialize the homogeneous adjoint equation. This equation is then solved up to T_0 (dashed blue lines) by each processor. 40

3.3 A schematic of the algorithm highlighting the communication between processors. Red lines represent the communication operations. 42

3.4 Temporal energy growth: vorticity field in a lid driven cavity at $Re = 1000$. 47

3.5 Convergence history of time integrators: (black) Explicit Adam-Bashforth method; (red) Krylov-based exponential Euler method. . 48

3.6 Drag reduction: vorticity field of the flow around a cylinder at $Re = 200$. 49

3.7 Performance of the parallel-in-time algorithm (red line) compared to the serial counterpart (black line) reported for a single iteration of the optimization loop, using steady control. 50

3.8 Accuracy of the estimated gradient computed for a single optimization loop: —, Adams-Bashforth method; —, exponential integrator without interpolation; — Exponential integrator with interpolation. 50

3.9 Total pressure loss: vorticity profile and pressure distribution for $t = 0.35$. (a) uncontrolled vorticity, (b) uncontrolled pressure. 52

3.10 Performance of the parallel-in-time algorithm (red line) compared to the serial counterpart (black line) reported for a single iteration of the optimization loop, using unsteady control. 53

4.1 Schematic of snapshot-POD. Temporal snapshots are collected and then a singular value decomposition is employed to extract the POD basis $\mathbf{U}$ and the singular values $\mathbf{\Sigma}$. 58

4.2 Results of the SVD on a sample of snapshots. (a) an example of a vorticity snapshot, (b) singular values, (c) POD modes. 61

4.3 Interpolation of four subspaces using the Grassmann manifold $\mathcal{G}(n_w, n)$ and its tangent space $\mathcal{T}_{\lambda_0}$ at the point $\mathcal{W}_0$. The subspaces are mapped onto the tangent space to perform the interpolation, then the resulting point $\widetilde{\chi}$ is projected back onto the manifold using an exponential mapping. 63

4.4 Training set of modes used for the interpolation for different Reynolds numbers. Left column: lift coefficient, the time interval used for the extraction of the modes is highlighted in red; center: horizontal velocity first POD mode; right: vertical velocity first POD mode. . . 64

4.5 First POD modes for horizontal and vertical velocity field at $Re^* = 150$. (Left): data-based POD modes extracted from the SVD of the training FOM data; (center): POD modes obtained with the interpolation on the Grassmann manifold and its tangent space at $Re_0 = 140$, using $N_R = 6$; (right): error between the data-based and the interpolated modes. 65

4.6 A schematic of the GNAT method. Model 1 corresponds to the FOM simulation. Model 2 is the first dimensional reduction performed by a Petrov-Galerkin Least-Square problem. Model 3 is the online phase involving a tensors approximation on a subset of points selected with the use of a greedy algorithm. 71

4.7 Results of the SVD on a sample of incremental snapshots for the GNAT method. (a) an example of a vorticity snapshot; (b) singular values, highlighted in red are modes selected for the projection, preserving the 99% of the energy; (c) incremental POD modes. . . . 72

4.8 Greedy points selection. (a) Subset of points $\mathcal{I}$ used for the hyper-reduction; (b) selection of the triad points, (u, v, p), on a cartesian staggered grid. 73

4.9 Reproduction of the lift (a) and drag coefficients (b) for model 2 and model 3, using GNAT compared to the high-dimensional model (model 1). 74

4.10 GNAT results: Reproduction of the horizontal velocity for a random point in the domain. 74

4.11 Set of modes used for the interpolation for different Reynolds numbers. The snapshots are collected using an incremental strategy. (Left): lift coefficient, the time interval used for the extraction of the modes is highlighted in red; (center): horizontal velocity first POD mode; (right): vertical velocity first POD mode. 77

4.12 First incremental POD modes for horizontal and vertical velocity fields at $Re^* = 150$, with $N_R = 6$. (Left), data-based POD modes extracted by performing an Singular Value Decomposition (SVD) on the data; (center), POD modes obtained with the interpolation of the Grassmann manifold and its tangent space at $Re_0 = 140$; (right), error between the data-based and interpolated modes. 78

4.13 Interpolation of reduced-order models: reconstruction of lift coefficient C_l (left column) and drag coefficient C_d (right column) using different sets of operating points N_R given in table 4.3. 79

4.14 Relative error in time in ℓ_2 for the approximated flow using model 2 of the GNAT method for $Re^* = 150$ using the interpolated modes obtained from the training dataset in table 4.3 and the data-based modes. 80

5.1 Vorticity field for a flow around a cylinder, $Re = 200$, selection of three random points in the domain. 87

5.2 Identified active terms of the library of candidate functions for three random points. 88

5.3 Graphic representation of nodes and edges of a network. (a) nodes; (b) nodes connected by a simple edge; (c) nodes connected by a directed edge; (d) nodes connected by weighted edges; (e) node with a self-edge. 89

5.4 Example of an undirected unweighted acyclic graph. (a) graph representation; (b) corresponding symmetric adjacency matrix. 90

5.5 Example of an agglomerative hierarchical clustering scheme applied for four data points. The dendogram on the right shows how the process generates a summary of a hierarchical clustering. Image reproduced from [27] 91

5.6 Example of an directed weighted cyclic graph. (a) graph representation; (b) corresponding asymmetric adjacency matrix: the edges have different weights highlighted with different shades of red, the contribution of self-loops appears on the diagonal. 93

5.7 Repeated bisection following the Leicht-Newman network procedure (a) the network is initially divided in two communities (5.20) (bisection I); (b) the community on the left is in turn further divided in two communities using the generalized formulation of the modularity for multiple groups (5.21) (bisection II). 95

5.8 Experimental configuration for the axisymmetric wake used in [175]. (a) the bluff body is mounted from the wind tunnel ceiling. (b) Center of Pressure probability distribution. 97

5.9 Probability distribution for the evolution of the Center of Pressure obtained by the modified system (5.26). 98

5.10 Community identification. (a) Initialization of 200 clusters with the *k-means* algorithm; (b) Communities identified by the Leicht-Newman network. 99

5.11 Vorticity field for a flow around a cylinder, $Re = 200$. 101

5.12 Dynamics identification on a two dimensional flow around a cylinder at $Re = 200$. (a) the selected points are divided in three main communities sharing same dynamics; (b) rearranged adjacency matrix showing correlations between communities; (c) example of active coefficients identified for points belonging to different communities, in the figure the values are normalized for each point. 102
5.13 Two dimensional slice of a three dimensional boundary layer flow, $Re = 1 \times 10^5$, $Ma = 0.2$. 104

List of Tables

2.1 Value of the cost functional before and after the actuation process. . 27
2.2 Pressure loss minimization, listing the cost functional before and after the actuation. 30

4.1 Time-averaged ℓ_2-error (4.32) of the approximation for model 2 and model 3. 74
4.2 Wall time for each level of approximation of acGNAT. 75
4.3 Training set of operating points Re used in the interpolation procedure for different N_R. 76

5.1 Numerical values of the identified coefficients for points belonging to different communities. 103

List of Acronyms

AD Automatic Differentiation 23

ADE Algebraic Differential Equation 15

CFD Computational Fluid Dynamics 2

CoP centre-of-pressure 96

DA Data Assimilation 56

DEIM Discrete Empirical Interpolation Method 13

DMD Dynamic Mode Decomposition 11

EBK Exponential Block Krylov 7

EIM Empirical Interpolation Method 11

FOM Full-Order Model 11

GNAT Gauss-Newton with Approximated Tensors vi

LASSO Least Absolute Shrinkage and Selection Operator 85

LSPG Least-Squares Petrov-Galerkin 68

PCA Prinicipal-Component Analysis 58

PDE Partial Differential Equation 6

POD Proper Orthogonal Decomposition 11

RBF Radial Basis Function 10

ROM Reduced-Order Model 2

SINDy Sparse Identification of Nonlinear Dynamics 15

SPD symmetric positive definite 69

STRidge Sequential Threshold Ridge Regression 100

SVD Singular Value Decomposition ix

Chapter 1

Introduction

Contents

1.1 Motivation 1
1.2 Control and optimization strategies 3
1.2.1 Gradient-based optimization 3
1.2.2 Derivative-free optimization 8
1.2.3 Multi-fidelity optimization 10
1.2.4 Data-driven model reduction and identification of inherent flow dynamics: 13
1.3 Outline of the thesis 14

1.1 Motivation

Numerical simulations of multiphysics and multiscale phenomena in fluid mechanics have advanced remarkably over the past decades due to the growth in computing power and infrastructures. Complex physical processes involved in the engineering industry, such as complex turbulent flows, including interfaces, shocks, and flames can now be simulated and predicted with an astonishing degree of fidelity and accuracy. However, in the current era in which awareness of climate change imposes ever more stringent restrictions on pollutants/emission resulting in an increasing demand for cleaner energy while minimising the waste in resources, it is just as crucial to be able to extract relevant optimization and control strategies dedicated to improving the performance and efficiency of the underlying industrial processes [71].

Flows encountered in energy conversion systems are often turbulent and are governed by the interaction of different physical and chemical phenomena, affecting the amount of emitted pollutants (NO_x, SO_x, CO, soot etc. [57]). The reduction of emissions and waste products can be achieved by acting on the parameters governing the physics of the system, ensuring that the system operates at the desired optimal condition. Effective flow control strategies can be achieved by small control effort, as an example figure 1.1 shows the smoke visualization of the jet injected from a flush nozzle into crossflow [109], highlighting the differences between (a) the uncontrolled flow, (b) the response of the transverse jet to sinusoidal excitation and (c) the response to square wave excitation. One application of jets in crossflow is in dilution or primary air jet injection in gas turbine combustor, making this application relevant for NO_x control.

In the case of unsteady and complex flow scenarios of interest, the number of model parameters and the large dimensionality of the objective function make the optimization procedure very complex. Therefore, reduced-order models or approximated equations are commonly used to replace the high fidelity model. By simplifying the function evaluation one allows the application of conventional control strategies [111, 197]. This approach also reduces the cost of the optimization algorithm such that multiple queries are possible in a short time period. While effective, the limited fidelity of the approaches in use today results in a stagnation of the improvements. The following arguments will therefore justify a multi-fidelity approach in order to increase the efficiency of the optimization process while maintaining the low cost of the algorithm: (i) resorting to model reduction amounts to replacing the exact solution of the high-fidelity model by an approximation which results in added uncertainties in the value of the functional and in the resulting optimization algorithm, therefore augmenting the fidelity of the forward solution, when necessary, will allow the optimization result to be more reliable, (ii) during the past few decades the focus has been directed towards building and designing computational paradigms dedicated to solving complex flow regimes with measurable success and improved efficiency; it is then beneficial to the computational science community to propose a strategy to make use of these accurate simulations through the optimization process.

However, in order to achieve this goal two main challenges need to be addressed:

- In order to perform the optimization on the basis of high-fidelity simulations, the solver need to be coupled with sophisticated control strategies, often leading to prohibitive computational costs. Although advances in computing capability and software have made Computational Fluid Dynamics (CFD) a valuable tool in determining control strategies in a limited number of applications [101, 103, 105, 55, 141, 2], as of yet, the application of high-fidelity control strategies to ever more complex systems of equations remains vastly unexplored.

- In order to successfully perform a multi-fidelity optimization, the reduced order models need to be predictive and robust with changing operating conditions. Despite recent efforts, most state-of-the-art reduced-order models fall short when highly unsteady and turbulent flow cases are considered [117].

In this thesis we propose different solutions to enable the use of control and optimization strategies on the basis of detailed simulations provided by the CFD, by (i) providing solutions to speed up the optimisation procedure, and (ii) proposing predictive Reduced-Order Model (ROM)s applicable over a variable range of parameter regime.

Figure 1.1: Effect of forcing at the nozzle inlet, for a jet in cross flow set-up, relevant for the injection process. (a) unforced jet, (b) jet forced with sine wave excitation, (c) jet forced with square wave excitation. Image reproduced from [109].

1.2 Control and optimization strategies

In the context of fluid dynamics, control and optimization strategies aim to modify a flow in order to match a desired output. The variables otherwise solved in the forward integration are in the context of control and optimization problems referred to as the state variables, q. In addition to the state variables, a set of parameters, g, is also defined constituting the control or design variables. The cost functional, $\mathcal{J}$, refers to the function that is ultimately optimized. Finally, a set of constraints is described by the governing equations and side conditions that the state and control variables have to satisfy. Therefore, the resulting optimization problem can be formulated as:

$$\begin{aligned} \text{Optimize} \quad &: \quad \mathcal{J}(q,g) \\ \text{While} \quad &: \quad \frac{\partial q}{\partial t} = F(q,g,t) \quad \text{for} \quad (0 \leq t \leq T)\,, \end{aligned} \tag{1.1}$$

which is completed by defining the appropriate initial conditions. Functional F denotes an integro-differential operator.
Control strategies can be divided into (i) passive control [68, 76] and (ii) active control techniques [42, 210, 63], where the former are constant in time and the latter depend on a time varying component (actuator). Independent of the type of control, due to the nonlinearities and unsteadiness present in the equations governing the flow, optimisation algorithms need to be employed to extract the most effective control regime.

Two major types of optimization methods in use today are (i) gradient-based methods and (ii) derivative free methods. In the following we briefly describe these methods and highlight their shortcoming when applied to high-fidelity simulations.

1.2.1 Gradient-based optimization

Gradient-based optimization methods reach the optimum by incrementally moving the design point to improve the value of the objective function based on a local model. The design point is iteratively moved along a descending direction until a minimum is reached. The descending direction is approximated by a first-order

(gradient), or second-order (Hessian) Taylor series expansions. Optimization algorithms which follow this approach rely on the knowledge of the local form of the cost function and are hence called *descend direction methods*.

Common methods in extracting the gradient are analytical or use finite differences. However, the subtleties associated with the numerical approach and the complexity of the governing equations, make extracting the analytical expression of the gradient impossible without a suitable (often complex) mathematical formulation. In the case of finite differences, the large parameter spaces make this approach prohibitively expensive for large problems, making the cost of the computation of the gradient proportional to the number of design variables, and can also be easily overwhelmed by numerical noise. A suitable alternative is the use of adjoint-based algorithms.

1.2.1.1 Adjoint-based methods

Adjoint-based methods allow the determination of the gradient at a cost comparable to a single function evaluation, regardless the number of design parameters. The adjoint equations are derived from the application of variational principle to an unconstrained optimization problem, to which the constraints (the equations governing the problem) are added to the cost function (Lagrangian) using the method of *Lagrangian multipliers*, allowing the extraction of the location where the contour-line of the cost functional is aligned with the contour-line of the constraint [112]. Adjoint-based algorithms are then used to transform a constrained problem into an unconstrained alternative, allowing the use of a vast range of optimization techniques, otherwise inapplicable. The gradient is then computed in the form of algebraic expressions based on the problem's Lagrangian multipliers or adjoint variables. Once derived, the adjoint equations are solved backward in time, providing gradient or sensitivity information about the optimization problem.

Originally arising as part of a design algorithm for fluid systems [162, 100, 103, 173], adjoint methods have been applied to aero- and thermo-acoustic applications [105, 120, 187]. These areas provide a suitable application for adjoint-based methods, which are inherently linear, since they too are dominated by linear dynamics. Recently, nonlinear problems have also been tackled, within the context of optimal control of separation on a realistic high-lift airfoil or wing, enhancement of mixing efficiency, minimal turbulence seeds and shape optimization to name a few [143, 186, 168, 69]. The same techniques have also been recently incorporated into reactive and multi-phase simulations [60, 25, 119, 23, 38, 87, 86, 67].

The formalism leading to the definition of the adjoint equation and the resulting optimality condition as well as its application on incompressible fluid flows is presented in chapter 2.

While very efficient and flexible, these algorithms still suffer from a great many challenges. On the algorithmic side, a key challenge is associated with the unsteadi-

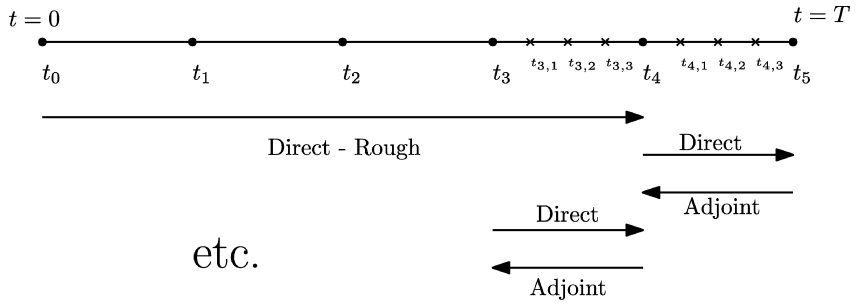


Figure 1.2: Graphical representation of the checkpointing procedure. Image reproduced from [193].

ness of the flow and the integration of the reverse problem. The evolution of the adjoint variable is governed by a linear, variable-coefficient dynamical system with a general structure similar to the forward problem, except that the adjoint is integrated backward in time. The variable-coefficient nature of the adjoint equations dictates that the solution of the forward integration is needed at each time step of the adjoint problem. This solution must be either stored in memory or recalculated from forward solutions at specifically chosen time instants, referred to as checkpoints. In large scale high-fidelity simulations, relevant in engineering applications, many time steps are usually required for each forward integration, leading to excessive memory requirements to store the solutions.

Checkpointing schemes in which only a small number of time steps is stored provide a remedy. In this approach, the solution is stored at carefully chosen checkpoints, and during the backward integration of the adjoint equations, the discarded intermediate solutions are then restored by starting anew the forward integration from the respective checkpoint. A schematic of the checkpointing procedure, showing the coarser checkpoints, is given in figure 1.2. Various checkpointing algorithms exist which aim to optimize the number of stored points in memory and the time required for the respective forward integration to access the intermediate solutions [201].

In unsteady cases, the use of checkpointing algorithms increases the computational costs nearly by a factor of three. In addition, the overall time to solution increases proportionally, since these operations (forward and backward integrations) are executed sequentially. This cost cannot be circumvented, since the adjoint equations need to be solved in order to gain access to derivative information. Therefore, each iteration of the optimization algorithm is at best twice the cost of a full CFD calculation (forward integration).

A way to speed up the optimization problem is parallelization. When dealing with complex unsteady problems, spatial parallelization has been widely adopted. However sometimes spatial parallelization is not feasible or it has reached its maximum efficiency, hence a time parallelization can be introduced in the resolution of the problem.

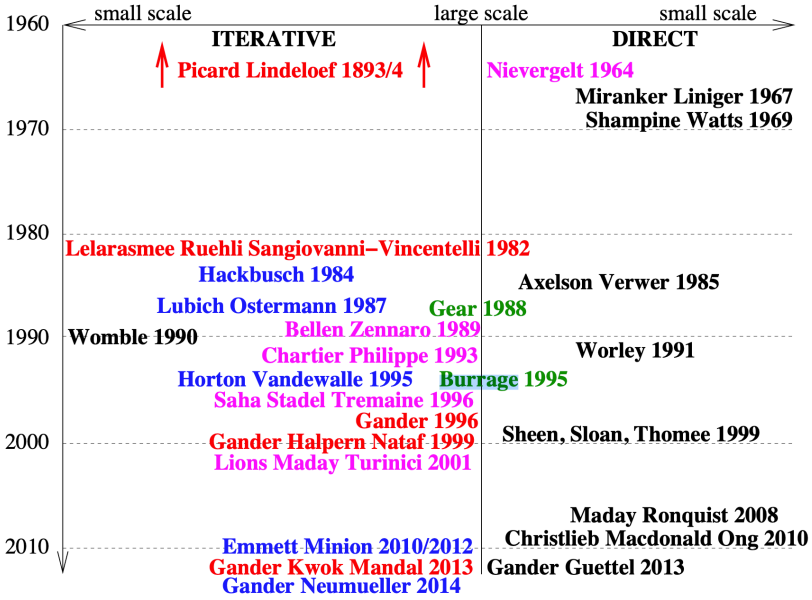


Figure 1.3: An overview over important contributions to time parallel methods. Image reproduced from [75].

1.2.1.2 Parallel-in-time algorithms

Time parallel integration has been an active area of research, which started with the pioneering work of Nievergelt (1965). The existing space-time parallel methods can be divided into four main categories: (i) methods based on multiple shooting, (ii) methods based on domain decomposition and waveform relaxation, (iii) methods based on multigrid, and (iv) direct time parallel methods [75]. An overview of time parallel methods is given in figure 1.3.

Direct time parallel methods are based on direct solvers in space-time and do not rely on iterations. In this thesis, we will consider the linear *ParaExp* algorithm proposed by Gander and Güttel [73]. This method is based on an overlapping time domain decomposition, where the time domain is decomposed in partitions and the linear problem is split into subproblems on overlapping intervals. In particular, the linear initial-value problem is separated in its homogeneous and inhomogeneous equations, proving that the method performs well when the existing inhomogeneity is hard to integrate, which is a common scenario in complex flows. The major gain obtained by the employment of this algorithm is given by the use of exponential integrators to solve the homogeneous equations resulting from the separation of the original problem. An extension of the linear *ParaExp* algorithm to nonlinear problems has been introduced by Gander [74] and its application on nonlinear Partial Differential Equation (PDE) has been proposed by Kooij [113].

The use and the adaptation of the *ParaExp* algorithm for optimization purposes based on the gradient information has been studied by Skene *et al.* [193], who

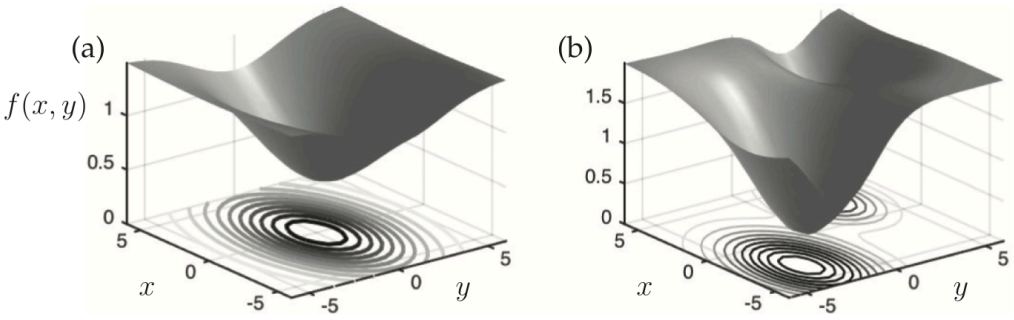


Figure 1.4: (a) convex function; (b) nonconvex function. For nonconvex functions, local minima make it challenging for gradient-based optimization. Image reproduced from [27].

proposed different strategies for the parallelization of the direct-adjoint loop. In their work, they proposed two algorithms to accelerate the computation of the gradient for a nonlinear forward problem. The first nonlinear algorithm proposed applies the nonlinear parallel algorithm proposed by Kooij to the forward problem and the linear *ParaExp* to the adjoint. The nonlinear forward equation has to be linearized through an iterative procedure in order to use the parallel integrator, and these iterations proved to reduce the performance of the algorithm.

The second algorithm adopts a hybrid approach, solving the nonlinear forward equation in serial and applying the linear *ParaExp* to the nonlinear adjoint equations. This algorithm does not need iterations and proved to reach the maximum speedup with a smaller number of processors and it will be employed in this thesis. However, the characteristics of the governing equations of interest to this work, and in particular, the divergence free constraint (incompressibility effect) as well as the nonlinearity and the unsteadiness of the flow, make direct application of the existing parallel-in-time algorithms not straightforward.

In chapter 3 we address these underlying issues, proposing an adaptation of the hybrid algorithm by Skene *et al.* to algebraic differential equations. The algebraic nature of the linear adjoint equations rising from the incompressible Navier-Stokes equation, and the divergence free constraint, makes the use of exponential time integrators (and therefore *ParaExp*) nontrivial. In our method we reformulate the homogeneous adjoint momentum equation following the Exponential Block Krylov (EBK) method used for the exponential integration of incompressible Navier-Stokes equation by Kooij [114]. The algorithm is then applied on a two-dimensional incompressible flow optimization problem, using both passive and active control.

Although gradient-based algorithms are very efficient in finding a local optimum, they suffer from few shortcomings which limit their application to unsteady complex

flow configurations:

- gradient-based optimization techniques based on adjoints tend to diverge when applied to complex configurations in presence of turbulence.
- Gradient-based algorithms show low performances when optimising an objective function with multiple local minima. As an example, figure 1.4 shows two functions to be minimized, the function on the left (1.4 (a)) is convex, i.e. it has one global minimum, while the function on the right (1.4 (b)) is nonconvex and includes multiple minima [27]. When applied to nonconvex functions, gradient-based algorithms could get stuck into local minima, and the convergence to a global optimum is not guaranteed.

One way to avoid the divergence of the adjoint equations in presence of turbulence, the gradient can be obtained using ROMs, solving the adjoint of only the coherent part of the flow included in the low-fidelity model. In this way, chaos is excluded and the adjoint equation is guaranteed to converge. Although the solution of the adjoint equation will then be an approximate gradient, it can be used to identify the direction towards the optimum. The resulting optimization algorithm falls under the category of multi-fidelity optimization procedures, that will be discussed later.

Alternatively, derivative-free optimization methods can be employed to ensure the convergence of the algorithm to the global optimum of nonconvex functions, even in the presence of turbulence.

1.2.2 Derivative-free optimization

Derivative free methods are specially useful for situations where multiple objectives are present, the optimization problem has many local minima, or access to gradient information is non-trivial or not feasible.

The procedure adopted by all derivative-free methods is based on the determination of the next points to evaluate in the design parameters space (search space). One way of helping the algorithm escape eventual local minima and ensuring the convergence of the optimization algorithm to the global minimum is the use of randomization. Randomization can be inferred at each optimization iteration, using random numbers during the search of the next candidate point, leading to *stochastic algorithms*, or it can be used to initialize and modify a collection of points in the design space defined as a *population* of *individuals* [112]. In this section we briefly explain these two main optimization strategies. For complete reviews of derivative-free optimization algorithms, the reader is referred to the recent works by Larson [116] and Rios [176].

1.2.2.1 Stochastic algorithms

Stochastic methods operate by iteratively moving a single design point in the feasible space of the objective function until a global minimum is reached, similarly to gra-

dient descend algorithms. In some cases, stochasticity is added to the conventional gradient-based descent algorithms (noisy descent) to increase the performance for large nonlinear optimisation problems, helping the algorithm traverse past saddle points [89]. In other algorithms, the randomization is directly used in the search of the new candidate point. Used this way, the stochasticity of the random search is controlled and slowly reduced advancing with the iterations, forcing the search to converge to a minimum. Starting the search with a high stochasticity allows the algorithm to escape local minima. Algorithms such as *simulated annealing*, *cross-entropy method*, *natural evolution strategies* and *Covariance matrix adaptation*, belong to the class of stochastic algorithms where probability distributions are used to sample the search space in order to move the design point towards the optimum. Each one of these algorithms employs a different strategy to sample or update these distribution functions [112]. In a series of lecture notes, Koumoutsakos and Müller [115] highlight some applications of stochastic algorithms to flow control, in particular the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [85] which has been applied to multi-objective optimization of automated profile design for compressor blades. The stochastic nature of these optimization algorithms also proves to be advantageous when considering robustness and optimality of the design in the presence of noise and uncertainties. However, due to the large number of function evaluations needed to converge, most of the optimization is performed for steady actuation or by the use of low-fidelity models.

1.2.2.2 Population methods

While stochastic methods use stochasticity to move one single design point toward the optimum of the objective function, population methods spread a collection of points (otherwise known as individuals) over the search space, in order to increase the chances that the samples are close to the best regions. Large number of individuals distributed in the design space and exchanging information about its local shape allows the algorithm to escape local optima [112]. Most population methods are stochastic in nature and they are usually easy to parallelise. *Genetic algorithms, particle swarm optimization*, etc. belong to this class of methods. An efficient class of generic algorithms based on the surrogate management framework [135] has been applied to optimise cardiovascular geometries. Hansen *et al.* [85] also demonstrate various examples of such algorithms applied to a variety of engineering problems ranging from aerodynamics and turbo-machinery to microtechnology, showing their suitability for optimization cases characterised by noise and multimodality in the absence of gradient information.

While an efficient class of generic algorithms based on the surrogate management framework [135, 136, 137] and artificial neural networks [161] have been used for optimization in fluid mechanics, mainly in the area of aerodynamic shape optimization, derivative-free optimization methods could require many function evaluations, for training purposes for example. When detailed simulations of complex turbulent phenomenon are concerned, each function evaluation commands

a full (potentially unsteady) CFD computation, making the global optimization procedure very expensive and impractical.

One way of speeding up the optimization process is to substitute the high-fidelity model with a low-fidelity counterpart when searching for the next promising design point during the optimization iterations.

1.2.3 Multi-fidelity optimization

While it is desirable to optimise a design using only the high-fidelity model, as discussed earlier, in cases where high-fidelity simulations prove to be too expensive, lower-fidelity models may provide valuable information that can accelerate the optimization process. The lower-fidelity models may be simplified physics models, or approximate models generated using methods such as response surfaces [84, 133, 189, 213], reduced-order models ROMs [6], or coarse discretisations [3].

The resulting optimization algorithm falls under the category of multi-fidelity optimization procedures. To be robust, multi-fidelity optimisation algorithms, applicable to unsteady multi-physics flow problems, must rely on accurate low-fidelity models. The more accurate the low-fidelity model, the less need for high-fidelity function calls. To increase the accuracy of the low-fidelity model, system identification can be used in order to discover and locate the dominant dynamics existing in a fluid. Additional information on the underlying physical mechanisms governing the flow can be used to define the optimization strategy (design parameters, sensor placement) and to build adapted low-fidelity models.

1.2.3.1 Surrogate models – low-fidelity representative of the objective function

One suitable alternative to reduce the cost of function evaluations is based on a response surrogate model, designed to be smooth, that is able to approximate the objective function in the design space. Performing the optimisation procedure on a surrogate surface limits the number of calls to the expensive high-fidelity model. One example of these optimisation methods relies on kriging models [104]. This model predicts the value of the design point using stochastic processes based on Gaussian random function. This model has proven to be flexible enough to handle nonlinear and multimodal functions. Alternatively, the surrogate model can be constructed using interpolation models based on polynomials [48, 166, 165] or Radial Basis Function (RBF) [22, 84, 170]. Studies have shown that, independent of the type of RBF, convergence can be achieved without further assumptions on the objective function. Although surrogate surfaces can be used in the context of multi-fidelity optimization to reduce the cost of the optimization procedure, they carry no information regarding the dynamics of the flow, but only the objective function and they will not be treated in this thesis.

1.2.3.2 Reduced-order models – low-fidelity representative of the high-fidelity simulation

Another class of low-fidelity models are physically based models that leverage information from the flow structures and governing equations. Although these models can not be used to replace the surrogate surface, they can be combined with high-fidelity simulations to reduce the cost of local function or gradient evaluation at the selected design point. Following this approach, when a new promising point is identified through an optimization algorithm (stochastic or following the descend direction), a ROM can be used for evaluating the function or the gradient.

Certain advantages can come from replacing the high-fidelity model with a ROM: (i) the cost of evaluating the function of the gradient is much lower than the high-fidelity model, accelerating the overall optimization algorithm; (ii) in cases where gradient information can not be directly extracted from the flow, such as in turbulent flows, the ROM is able to evaluate an approximate gradient; (iii) even in circumstances where the gradient can be extracted using the adjoint methodology, in practice, incorporating adjoint information inside an existing high-fidelity solver is a challenging task. In these cases an approximate gradient computed using ROMs reduces the cost of extracting the gradient information.

ROMs can be categorized into physical based models, based on the knowledge of the underlying governing equations which are projected on a lower dimensional subspace, and models that use input-output data or system identification [122, 123, 121].

One of the features of most turbulent flows, is the existence of coherent structures. Although, the flow itself is non-deterministic due to the presence of chaos, deterministic behaviour can be attributed to these coherent structures. As an example, figure 1.5 shows a photograph of the von Karman vortex shedding generated by the Rishiri Island, compared to the two dimensional flow around a cylinder at low Reynolds number, suggesting the existence of coherent spatial features that capture the dominant dynamics of the flow.

Physical based models are able to capture the principal characteristics of the high-fidelity Full-Order Model (FOM), potentially leading to high speedup. Projection based ROM are used to reduce the dimension of the system of linear or nonlinear equations governing the physical phenomenon, leveraging the data from the detailed simulation (or experimental data) using data-decomposition techniques. Data decomposition techniques have received considerable attention in the community due to their ability to provide physical explanations for dynamical flow processes by extracting the relevant energy-dominated or frequency-dominated modes to form the basis for the projection-based methods.

Some of the most used techniques to extract these modes have been summarized and applied in the reviews by Taira *et al.* [194, 196]. Among these techniques are Proper Orthogonal Decomposition (POD) [192, 11, 92], Dynamic Mode Decomposition (DMD) [185], and greedy reduced basis [130, 126] and Empirical Interpolation Method (EIM) [14, 45, 159] with its generalized version [128, 129, 21].

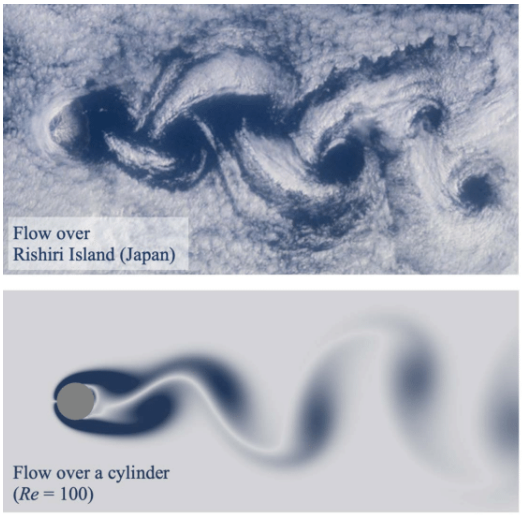


Figure 1.5: Von Kármán vortex street generated by the Rishiri Island in Hokkaido, Japan (top: photograph from NASA, 2001; STS-100); cylinder wake at a low Reynolds number (bottom). Image reproduced from [196].

Through these procedures, the most energetic modes are extracted, which then form a hierarchical basis for describing the flow.

POD, also known as Karhunen-Loève expansion and Prinicipal-Component Analysis (PCA), is a linear procedure to extract uncorrelated modes from a flow. The resulting features (modes) are then used to define a orthogonal basis that optimally approximates the high-dimensional data. Despite the common use of projection-based models, they can suffer from a few shortcomings. One important limitation of such models is the lack of predictability when highly nonlinear and especially turbulent configurations are considered. Secondly, the truncation of the expansion ignores the cumulative effect of truncated scales on the retained degrees of freedom, making the model-reduced system prone to detrimental long-term instabilities [172].

Different methods have been proposed to overcome these problems and improve the stability of the Galerkin projection method stabilizing the inner products [108, 13, 179] introducing numerical dissipation via closure models [13, 17, 96, 97, 190, 203], including a numerical model for the pressure [72, 92, 155], using eddy viscosity [9, 18, 163], performing nonlinear Galerkin projection [134, 188], and perform Petrov-Galerkin projection [64, 208, 41].

In their basic form, projection based ROMs scale with the dimensionality of the underlying high-fidelity simulation and are therefore efficient for primarily linear and steady problems. To further improve the computational efficiency and the dimension of the reduced models, empirical techniques were proposed for capturing the non-linearities and evaluate the non-linear terms at a subset of

points [14, 21, 41, 45, 128, 150, 149, 159, 129]. Among these, Carlberg *et al.* [41] proposed the GNAT method, where the error made by the truncation process is included when defining the coefficients of the remaining modes, and therefore improves the performance of the reduced model. This method is based on a fully discrete Petrov-Galerkin projection, proving an *a priori* stability. In addition, a second level of approximation is added to the model reduction to approximate the tensors involved in the Petrov-Galerkin projection using a gappy POD technique, further reducing the dimension of the online stage.

The low-fidelity models obtained with projection based ROM mentioned reduce the computational cost of the simulations by dimensionally reducing the high-fidelity model. However these methods lack of robustness to varying parameters. Therefore, when considering changes in the parameter space a new ROM must be rebuilt. This is particularly disadvantageous in control applications, where design parameters are modified at each optimization iteration or the function has to be evaluated at new operating points. To enhance adaptivity of projection based ROM, Amsallem *et al.* [5, 4] proposed an interpolation on a tangent space of the Grassmann manifold as an alternative method for adapting precomputed orthogonal bases to new physical or modeling parameters.

The GNAT dimensional reduction methodology and the interpolation on the Grassmann manifold and its tangent space are employed in this thesis (chapter 4) to construct an optimised POD basis, and its respective ROM for fixed and variable operating conditions.

1.2.4 Data-driven model reduction and identification of inherent flow dynamics:

Together with data-driven projection-based models, described earlier, system identification can be utilized to estimate the unknown parameters driving the underlying predetermined basis, leading to models that are particularly suited for control applications, since, by design, the dynamics taken into account are part of the input-output behavior of the system. Used in this context, the identification process is used to replace other methods which aim to add nonlinearities to an inherently linear basis such as Discrete Empirical Interpolation Method (DEIM) or GNAT. Following this approach [123] once the data-driven reduced basis is extracted and truncated, machine learning strategies are used to identify the nonlinearities through mode coupling.

Dynamical data obtained numerically or experimentally can, on the other hand, be utilized to infer the equations governing the existing dynamics, which has been shown to scale to high-dimensional systems [29]. The identification of the dynamics existing in a dynamical system provides physical insight and interpretability into the system's behaviour. Two important challenges in analysing a dynamical

complex system are the existing nonlinearities and the lack of known governing equations for many modern realistic systems. For example in industrial applications involving phenomena such as turbulence and combustion, usually described by high-dimensional systems, finding patterns to determine the dominant behaviour is not trivial, motivating the need for automated model discovery techniques.

System identification has been employed to discover the underlying equations governing a physical system or completing them with closure models. Methods for data-driven discovery of dynamical systems include equation-free modeling [110], artificial neural networks [81], nonlinear regression [200], empirical dynamical modeling [209], normal form identification [132], nonlinear Laplacian spectral analysis [77], modeling emergent behaviour [177], automated inference of dynamics [50], and more recently sparse regression [29, 43, 181, 180]. Their application has been extended to incorporate the effect of control, where a small deviation in the design parameters can produce fundamentally different system behaviour [66, 31, 107], increasing the robustness of these model to varying parameters.

As already mentioned, in many regimes, the dynamics of the flow, usually described by complex partial differential equations, are governed by only a few nonlinear terms, allowing the production of simpler lower-fidelity models capable of predicting the main features of said system. Identifying and locating the different dynamics present in complex physical systems and eventually understand how they affect the overall solution can increase the accuracy of the numerical model employed. Therefore, the identification process can alternatively be used directly to discover dominant dynamics in the flow as proposed by [35]. Following this approach, in chapter 5, we propose a strategy to identify and classify the dominant dynamics existing in a flow. The equation governing the flow (incompressible Navier-Stokes) is directly analysed on the discrete spatio-temporal domain. The active terms in the equation governing the dynamics, at discrete locations of the domain, are then identified using sparse regression techniques. Coupling the identification procedure with clustering algorithm, we are finally able to identify the dominant dynamics of the flow, spatially locate them and determine how they interact one another. Additional insight on the mechanism governing a system and eventually their interactions can have a significant role in the selection of the design parameters, the placement of the sensors, and the possibility to replace the high-fidelity simulation with ROMs, allowing a more detailed understanding of the effect of the control and then reducing the complexity of the problem.

1.3 Outline of the thesis

In this introduction chapter, the main challenges encountered when applying optimization algorithms to complex dynamical systems have been presented. The main objective of this thesis is to explore and propose different solutions to ease the use of control and optimization strategies, otherwise unfeasible due to the

high-computational cost associated with high-fidelity simulations.

Before introducing the methods used to accelerate the solution to the optimization problem, the formal derivation of the optimality system for gradient-based optimization methods, is given in chapter 2. In this chapter, the gradient is extracted using the adjoint equations. The adjoint equations are derived from the incompressible Navier-Stokes equations, for two-dimensional fluid flows applications. Following the discretized-then-differentiate approach, the discrete adjoint equations are extracted and implemented in a numerical solver with immersed boundaries capabilities. The numerical code is then validated by imposing both passive and active control for unsteady fluid systems.

In chapter 3 we introduce parallel-in-time techniques and extend the hybrid algorithm proposed by Skene *et al.* [193] to the incompressible Navier-Stokes equations. The original algorithm is modified to incorporate the divergence-free constraint into the momentum equation, therefore to reformulate the Algebraic Differential Equation (ADE) governing the inverse problem in the form of a PDE. This operation is required to allow the use of exponential integrators, that represent the main source of time gain in the algorithm. Our algorithm is first validated, studying the convergence of the exponential time integrator, its performance is then evaluated on passive and active control.

While these first two chapter are closely related to each other and focus on gradient-based optimization techniques, the rest of the thesis aim to propose methodologies that can be employed for both gradient-based and derivative-free optimizations, approaching multi-fidelity optimization.

Chapter 4 investigates modal decomposition and projection-based ROM. We use the GNAT method to build a reduced-order model based on spatial POD modes and add an adaptive capability by performing the interpolation on a tangent space to the Grassmann manifold.

Finally, in chapter 5 we propose a data-based strategy to identify dominant dynamics existing in a system. In this chapter, we introduce sparse regression for dynamical systems based on Sparse Identification of Nonlinear Dynamics (SINDy) algorithm. The clustering is done by using network science and graph partitioning.

Chapter 2

Adjoint-based optimization

Contents

2.1 Introduction 17
2.2 Optimisation problems 18
2.3 Adjoint-based algorithms 19
2.3.1 Continuous versus discrete adjoint equations 22
2.4 Governing equations and numerical framework 23
2.4.1 Discrete adjoint Navier-Stokes equations 25
2.5 Application to flow configurations 25
2.5.1 Drag reduction – steady actuation 26
2.5.2 Total pressure loss – unsteady actuation 27
2.6 Conclusions 34

2.1 Introduction

The use of adjoint-based techniques for flow control and optimization has been an important research area for many years. Originally arising as part of a design algorithm for fluid systems, with the work of Pironneau [162] and later Jameson and co-workers [100, 101, 103, 173], adjoint-based methods have been widely used in fluid mechanics, with applications in acoustic and thermo-acoustics [105, 120], dominated by linear dynamics, and more recently to nonlinear systems and reactive and multiphase flows [186, 168, 69, 60, 25, 119, 23, 38, 87, 86, 67].

In the context of fluid mechanics, the optimization problem aims to find the optimal value for a set of design parameters to manipulate the flow to reach a desired output. This is effectively a minimisation problem, most commonly nonlinear, including a set of constraints. In this chapter we first introduce the general constrained optimization problem and how to solve it using gradient-based techniques §2.2.

The most common ways in computing the gradient, such as analytical differentiation or finite-difference, are proved to be inefficient, or even numerically non-realizable, when applied to large problems and high-dimensional design space, due to the large number of function evaluations needed. To overcome this difficulty, we adopt adjoint-based techniques, which allow the determination of the gradient at a cost of a single function evaluation, regardless of the number of design parameters. In section §2.3 we formally derive the continuous adjoint equations by using a variational approach. In Section §2.4 the discrete adjoint methodology is described and

applied to the case of incompressible Navier-Stokes equations with immersed boundary forces (a two-dimensional, finite volume, python solver [52] used throughout this work). Finally the results obtained are presented in section §2.5.

2.2 Optimisation problems

The goal of any optimisation problem is the minimisation or maximisation of an objective function (cost functional) $\mathcal{J}(\boldsymbol{q}, \boldsymbol{g})$, where $\boldsymbol{q}$ are the state variables and $\boldsymbol{g}$ are the control or design variables, subject to a set of constraints $\mathbf{F}(\boldsymbol{q}, \boldsymbol{g}, t)$. Using this formalism, the generic constrained optimisation problem can be formulated as

$$\begin{aligned} \min_{\mathbf{g}} \quad & \mathcal{J}(\boldsymbol{q}, \boldsymbol{g}) \\ \text{s.t.} \quad & \mathbf{F}(\boldsymbol{q}, \boldsymbol{g}, t) = 0 \ \text{ for } \ 0 \leq t \leq T. \end{aligned} \tag{2.1}$$

The set of equations described above is completed by defining initial conditions and boundary conditions when appropriate.

One class of optimisation algorithms used to optimise the above problem, rely on gradient (first-order) or Hessian (second-order) information in order to identify a descent direction in the objective function phase-space to progressively move towards the optimum. These classes of algorithms are commonly known as “descent direction methods". Gradient-based methods (as a sub-category of such algorithms) are therefore iterative and rely on the knowledge of the gradient of the cost function $\mathcal{J}$ with respect to the control parameters $\boldsymbol{g}$. In these methods the design point is modified at each iteration taking a step in the descending direction, approximated by the local value of the gradient (compute the Hessian can be too costly), until a minimum is reached. Analytical derivation of the local gradient of the cost functional in complex systems, due to the numerical artifacts such as the meshing process and discretisation, and the complexity of the governing equation is usually impossible, hence different numerical techniques should be used to approximate the gradient [49, 83].
A straightforward approach to compute the gradient is to use finite differences to approximate the gradient numerically, incrementing one control variable g_k at a time, with $k = 1, 2, \ldots, K$, K being the number of parameters,

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}g_k} \approx \frac{\mathcal{J}\left(\boldsymbol{q}\left(\boldsymbol{g} + \Delta g_k \mathbf{e}_k\right), \boldsymbol{g} + \Delta g_k \mathbf{e}_k\right) - \mathcal{J}\left(\boldsymbol{q}\left(\boldsymbol{g}\right), \boldsymbol{g}\right)}{\Delta g_k}, \tag{2.2}$$

where $\boldsymbol{q}\left(\boldsymbol{g}\right)$ is obtained by solving the state equation $\mathbf{F}(\boldsymbol{q}, \boldsymbol{g}, t) = 0$. The computation of the gradient using this strategy requires $K+1$ solutions of the state equation which is, in general, nonlinear and computationally expensive.
Another option is to differentiate equations (2.1) with respect to the control variables $\boldsymbol{g}$ using the chain-rule to get the total derivative of the cost function

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{g}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{g}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{g}}, \tag{2.3}$$

where $d\boldsymbol{q}/d\boldsymbol{g}$ is the *sensitivity* and its computation using finite differences requires again $K+1$ solutions of the state equations.
Alternatively, the state equation in (2.1) can be differentiate with respect to $\boldsymbol{g}$ to obtain the following sensitivity equation

$$\frac{\partial \mathbf{F}}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{g}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{g}} = 0. \tag{2.4}$$

Using the sensitivity equation (2.4) to compute the gradient $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{g}$ in equation (2.3) requires the resolution of a linear equation for each parameter g_k.

All the methods proposed above make the computation of the gradient proportional to the number of control variables in $\boldsymbol{g}$, making the use of these approaches very expensive or even unfeasible for large problems and high-dimensional design spaces. An efficient alternative for the computation of sensitivities is the construction and use of adjoint equations.

2.3 Adjoint-based algorithms

Adjoint equations are derived from the application of variational principle to an unconstrained optimization problem, for a detailed classification of various differential equations and their respective adjoint equations see Cao *et al.*[37]. In the context of this thesis, we consider general ADE, where the incompressible formulation of the Navier-Stokes equations imposes the equality constraints, the state variables are the velocity and pressure field. The constrained optimization problem for such algebraic differential equations, with its initial conditions, can be written as

$$\min_{\mathbf{g}} \qquad \mathcal{J}(\boldsymbol{q},\boldsymbol{g}), \quad \text{where} \quad \mathcal{J}(\boldsymbol{q},\boldsymbol{g}) \equiv \int_0^T J(\boldsymbol{q},\boldsymbol{g},t)\,\mathrm{d}t, \tag{2.5.1}$$

$$\text{s.t.} \qquad \mathbf{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{g},t) = 0 \ \text{ for } \ 0 \le t \le T, \ \text{ and } \ \mathbf{G}(\boldsymbol{q}(0),\boldsymbol{g}) = 0. \tag{2.5.2}$$

Fréchet-differentiating equations (2.5.1) and (2.5.2) with respect to $\boldsymbol{g}$, we arrive at

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{g}} = \int_0^T \left(\frac{\partial J}{\partial \boldsymbol{q}}\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial J}{\partial \boldsymbol{g}}\right)\mathrm{d}t, \tag{2.6.1}$$

where the sensitivity $\mathrm{d}\boldsymbol{q}/\mathrm{d}\boldsymbol{g}$ is determined by

$$\frac{\partial \mathbf{F}}{\partial \boldsymbol{q}}\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}}\frac{\mathrm{d}\dot{\boldsymbol{q}}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{g}} = 0 \text{ for } 0 \le t \le T, \tag{2.6.2}$$

and

$$\frac{\partial \mathbf{G}}{\partial \boldsymbol{q}(0)}\frac{\mathrm{d}\boldsymbol{q}(0)}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{G}}{\partial \boldsymbol{g}} = 0. \tag{2.6.3}$$

For the application of adjoint-based algorithms, the constrained problem has to be transformed into an unconstrained alternative. This transformation can be

achieved using the method of *Lagrangian multipliers*, where an auxiliary function known as the *Lagrangian*, or *augmented cost function*, is defined as

$$\mathcal{L}(\boldsymbol{q}, \boldsymbol{g}, \boldsymbol{\lambda}, \boldsymbol{\lambda}_0) = \mathcal{J}(\boldsymbol{q}, \boldsymbol{g}) - \Big(\langle \boldsymbol{\lambda}^\dagger, \mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{g}, t) \rangle - \langle \boldsymbol{\lambda}_0^\dagger, \mathbf{G}(\boldsymbol{q}(0), \boldsymbol{g}) \rangle \Big) \tag{2.7}$$

here $(\cdot)^\dagger$ denote the conjugate transpose and $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\lambda}_0$ are the so-called Lagrange multipliers or adjoint variables. The inner products are defined as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \int_0^T [\mathbf{a}, \mathbf{b}] \mathrm{d}t, \tag{2.8}$$

$$[\mathbf{a}, \mathbf{b}] = \begin{cases} \int_\Omega \mathbf{a}^H \mathbf{b} \, \mathrm{d}V & \text{Continuous case,} \\ \mathbf{a}^H \mathbf{b} & \text{Discrete case.} \end{cases} \tag{2.9}$$

Note that if the constraints in equation (2.5.2) hold, the value of the Lagrangian and the cost functional coincide, as well as their respective gradients, for an arbitrary choice of $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\lambda}_0$. Setting the variation of the Lagrangian with respect to the independent variables $\boldsymbol{q}(t)$, $\boldsymbol{g}$ and $\boldsymbol{\lambda}(t)$ leads to the *optimality system* consisting of

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\lambda}} &= 0 \qquad \text{state equation (constraints);} \\ \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{q}} &= 0 \qquad \text{adjoint equation;} \\ \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{g}} &= 0 \qquad \text{optimality condition.} \end{aligned}$$

Substituting the definition of the Lagrangian (2.7) into this system and computing the derivatives we get to

- the state equation:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\lambda}} &= \mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{g}, t) = 0 \;\text{ for }\; 0 \le t \le T, \\ \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\lambda}_0} &= \mathbf{G}(\boldsymbol{q}(0), \boldsymbol{g}) = 0; \end{aligned} \tag{2.10}$$

- the adjoint equation and its initial condition:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{q}} = \frac{\partial J}{\partial \boldsymbol{q}} \boldsymbol{\lambda}^\dagger \frac{\partial \mathbf{F}}{\partial \boldsymbol{q}} + \frac{\partial}{\partial t} \left(\boldsymbol{\lambda}^\dagger \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}} \right) = 0 \;\text{ for }\; 0 \le t \le T, \\ \boldsymbol{\lambda}^\dagger(T) \left. \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}} \right|_T = 0, \\ \boldsymbol{\lambda}^\dagger(0) \left. \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}} \right|_0 - \boldsymbol{\lambda}_0^\dagger \frac{\partial \mathbf{G}}{\partial \boldsymbol{q}(0)} = 0; \end{aligned} \tag{2.11}$$

- and finally to $\partial \mathcal{L} / \partial \boldsymbol{g}$, hence the gradient:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{g}} = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{g}} - \int_0^T \boldsymbol{\lambda}^\dagger \left(\frac{\partial \mathbf{F}}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}} \frac{\mathrm{d}\dot{\boldsymbol{q}}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{F}}{\partial \boldsymbol{g}} \right) \mathrm{d}t \\ - \boldsymbol{\lambda}_0^\dagger \left(\frac{\partial \mathbf{G}}{\partial \boldsymbol{q}(0)} \frac{\mathrm{d}\boldsymbol{q}(0)}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{G}}{\partial \boldsymbol{g}} \right). \end{aligned} \tag{2.12}$$

Substituting equation (2.6.1) into equation (2.12) and integrating by parts the term $\boldsymbol{\lambda}^\dagger(\partial\mathbf{F}/\partial\dot{\boldsymbol{q}})(\mathrm{d}\dot{\boldsymbol{q}}/\mathrm{d}\boldsymbol{g})$, after some manipulations we arrive at

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{g}} = \int_0^T \left[\left(\frac{\partial J}{\partial \boldsymbol{q}} - \boldsymbol{\lambda}^\dagger\frac{\partial \mathbf{F}}{\partial \boldsymbol{q}} + \frac{\partial}{\partial t}\left(\boldsymbol{\lambda}^\dagger\frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}}\right)\right)\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\boldsymbol{g}} + \frac{\partial J}{\partial \boldsymbol{g}} - \boldsymbol{\lambda}^\dagger\frac{\partial \mathbf{F}}{\partial \boldsymbol{g}}\right]\mathrm{d}t - \\ - \left[\boldsymbol{\lambda}^\dagger\frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}}\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}\boldsymbol{g}}\right]_0^T - \boldsymbol{\lambda}_0^\dagger\left(\frac{\partial \mathbf{G}}{\partial \boldsymbol{q}(0)}\frac{\mathrm{d}\boldsymbol{q}(0)}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{G}}{\partial \boldsymbol{g}}\right).\end{aligned} \tag{2.13}$$

It can readily be noticed that the gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{g}$ can be made independent of $\mathrm{d}\boldsymbol{q}/\mathrm{d}\boldsymbol{g}$ by using the adjoint system in equation (2.11). Note that the adjoint equation is also an algebraic differential equation. The backward solution in effect provides gradient or sensitivity information about an optimization problem, using the optimality condition (referred to as the Karush-Kuhn-Tucker equation) that links the result from the adjoint evolution equation to the optimal control strategy. Finally, the gradient reads

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{g}} = \int_0^T \left(\frac{\partial J}{\partial \boldsymbol{g}} - \boldsymbol{\lambda}^\dagger\frac{\partial \mathbf{F}}{\partial \boldsymbol{g}}\right)\mathrm{d}t - \boldsymbol{\lambda}_0^\dagger\frac{\partial \mathbf{G}}{\partial \boldsymbol{g}}. \tag{2.14}$$

Solving simultaneously for $\boldsymbol{g}$, $\boldsymbol{q}(t)$ and $\boldsymbol{\lambda}(t)$ using the optimality system (2.10),(2.11),(2.14) and setting $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{g} = 0$, leads to the so-called one-shot method. In the case of numerical simulations of unsteady flows, the resulting problem becomes computationally intractable due to the large dimensionality of the resulting system. Instead, an iterative approach is usually preferred, where the direct equation (2.10) is solved forward in time, the adjoint equation (2.11) is then initialised using the direct solution and solved backward, from the final time to the initial time. Finally, the gradient can be computed by using equation (2.14). Once the gradient has been extracted, it can be used to perform a step of a gradient descent algorithm, the procedure is then repeated until a local minimum is reached.

While very efficient and flexible, these algorithms still suffer from many challenges. On the algorithmic side, a key challenge is associated with the unsteadiness of the flow and the integration of the reverse problem. The evolution of the adjoint variable is governed by a linear, variable-coefficient dynamical system with a general structure similar to the forward problem, except that the adjoint is integrated backward in time. The variable-coefficient nature of the adjoint equations dictates that the solution of the forward integration is needed at each time step of the adjoint problem. This solution must be either stored in memory or recalculated from forward solutions at specifically chosen time instants, referred to as checkpoints. Checkpointing schemes in which only a small number of time steps is stored provide a remedy. In this approach, the solution is stored at carefully chosen checkpoints, and during the backward integration of the adjoint equations, the discarded intermediate solutions are then restored by starting anew the forward integration from

the respective checkpoint. In unsteady cases, the use of checkpointing algorithms increases the computational costs nearly by a factor of three [202].

2.3.1 Continuous versus discrete adjoint equations

In the previous section, the procedure for deriving the adjoint equations are highlighted, starting from the continuous direct equations and applying a variational principle to the unconstrained optimization problem and setting the first variations with respect to all involved dependent variables to zero to achieve the optimality condition. This results in governing equations for the direct (primal) and the adjoint variables, together with appropriate boundary conditions, initial conditions, and optimality expressions. The optimization problem arising from these continuous equations can not be solved exactly, hence some finite-dimensional approximations have to be performed. Two different approaches coexist in the literature to obtain the discrete form of the adjoint equation, known as continuous (differentiate-then-discretize) and discrete (discretize-then-differentiate) approaches. It is important to mention that, while theoretically both converge to the analytic solution in the limit of infinite grid resolution, the two methods can lead to different discrete approximations of the solution, as the discrete differentiation and discretization operators do not necessarily commute [191].

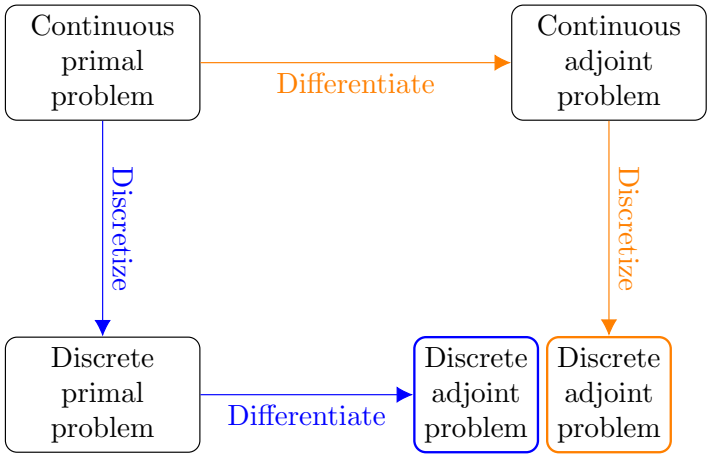


Figure 2.1: Schematics of the two different possible paths to derive the discrete adjoint problem. Orange arrows: differentiate-then-discretize (or continuous) approach; Blue arrows: discretize-then-differentiate (or discrete) approach.

The differentiate-then-discretize approach starts from the continuous form of the adjoint equations which subsequently have to be discretized and implemented [102, 20, 205]. Using this procedure the continuous adjoint equations obtained do not depend on the solver and can be discretized on an appropriate grid (chosen by the requirement of the adjoint equation solely), which may be different from the one used for the forward problem. This can also be particularly useful in shape

optimization problems, where the grid usually changes from one iteration to the next. However, handling the boundary conditions, as well as the stability and convergence of the method remain critical issues [78]. This process, in particular for complex governing equations and/or optimization objectives, is very cumbersome and error-prone, leading eventually to inconsistent gradient of a continuous functional.

In the second approach, the equations of the forward problem are first discretized and then differentiated to obtain the discrete adjoint equations. One common way to create the adjoint code associated with the discrete primal equations is using Automatic Differentiation (AD) [184]. This approach often leads to overly inflated, and thus very inefficient and ultimately impractical codes. Alternatively, Fosas de Pando *et al.* [53] implemented and validated an approach that extracts linearized and adjoint information directly from a nonlinear simulation code. Following this approach, the nonlinear modules are linearized and trans-conjugated, such that by producing the adjoint of a graph, the adjoint solution is produced, leading to the exact gradient of the discrete objective function. In this manner, the adjoint information is simply extracted from the already existing nonlinear simulation code, avoiding significant additional programming effort, and exploiting the discretization schemes of the original solver, causing it to run as efficiently (and parallel) as the original code; the adjoint code is simply embedded in the nonlinear simulation code. In addition, modifications to the code, such as the addition of reactive flow simulation capabilities, are automatically reflected on the adjoint side by local differentiation of the added modules/subroutine, usually by means of complex step differentiation. This strategy has shown great promise in reducing trailing edge noise and improving airfoil shape design in an aeroacoustic application [53] as well as extracting the mechanism governing the frequency response of an M-flame to the surrounding acoustic wave [23], and has been adopted here for extracting the discretised adjoint equations.

2.4 Governing equations and numerical framework

In this section, the differential algebraic equations that arise from the spatial discretization of the Navier–Stokes equations for incompressible flow are presented. Without loss of generality, we consider the projection-based immersed boundary method introduced by [195] and implemented in the nonlinear solver [52]. This method is proposed for incompressible flows over obstacles with prescribed surface motion. One of the advantages of this fractional step approach is that the boundary forces are determined implicitly without any constitutive relations allowing larger time steps, during the time integration process, compared to alternative implementations. Symmetry and positive-definiteness of the system are preserved such that the conjugate gradient method can be used to solve for the flow field.

Following this approach, the physical domain is extended by embedding these obstacles and introducing localized volume forces at the boundaries such that the

boundary conditions are satisfied. In continuous form, we have

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{u} + \int_B \boldsymbol{f}(\boldsymbol{\xi}(\boldsymbol{s},t))\delta(\boldsymbol{\xi}(\boldsymbol{s},t) - \boldsymbol{x})\,\mathrm{d}\boldsymbol{s}, \tag{2.15.1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.15.2}$$

$$\boldsymbol{u}(\boldsymbol{\xi}(\boldsymbol{s},t)) = \int_\Omega \boldsymbol{u}(\boldsymbol{x},t)\delta(\boldsymbol{\xi}(\boldsymbol{s},t) - \boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \boldsymbol{u}_B(\boldsymbol{s},t), \tag{2.15.3}$$

on the domain $\Omega \times [0,T]$, together with the initial condition at $t = 0$ given by $\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0$ and suitable boundary conditions at $\partial\Omega$. In the above, ν is the kinematic viscosity of the fluid, the last term in equation (2.15.1) represents the contribution of the localized forces $\boldsymbol{f}$ at the surface of the obstacle B described by $\boldsymbol{\xi}(\boldsymbol{s},t)$ and δ is the Dirac delta function. Similar to the role of the pressure p in fulfilling the incompressibility constraint (equation (2.15.2)), the localized forces $\boldsymbol{f}$ are introduced such that the velocity field at the boundary of the surfaces coincides with the prescribed value $\boldsymbol{u}_B(\boldsymbol{\xi},t)$ (equation (2.15.3)). In the following, we consider initial and boundary conditions that are parameterized by the vector of design variables $\boldsymbol{g}$, which we seek to optimize the given objective function.
Equations (2.15.1)–(2.15.3) are discretized in space using the finite volume method on a Cartesian grid with a staggered arrangement for the velocity and pressure variables. A set of Lagrangian points is introduced at the surface of the obstacles, and the boundary forces are then applied on these points to satisfy the no-slip constraint along the surface of the immersed bodies. In particular, we have

$$\mathbf{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{g},t) \equiv \begin{pmatrix} \mathbf{M}\dot{\boldsymbol{u}} + \mathbf{N}(\boldsymbol{u},\boldsymbol{g}) + \mathbf{Q}\boldsymbol{\phi} - \mathbf{L}\boldsymbol{u} + \mathbf{b}_1(\boldsymbol{g}) \\ \mathbf{Q}^\dagger \boldsymbol{u} + \mathbf{b}_2(\boldsymbol{g}) \end{pmatrix} = 0 \text{ for } 0 \le t \le T, \tag{2.16.1}$$

$$\boldsymbol{u}(0) - \boldsymbol{u}_0(\mathbf{g}) = 0, \tag{2.16.2}$$

where $\boldsymbol{u}$ and $\boldsymbol{\phi} = (p, \boldsymbol{f})^T$ are now the discrete representation of the velocity, pressure and boundary forces, respectively. In the above, $\mathbf{M}$ is the mass matrix, $\mathbf{N}(\boldsymbol{u},\boldsymbol{g})$ and $\mathbf{L}$ are the discretized advection and diffusion operators, respectively, and $\mathbf{Q}$ represents the discretized gradient and interpolation operators that are respectively applied to the pressure and the localized forces. Finally, the boundary terms $\mathbf{b}_1$ and $\mathbf{b}_2$ arise from the spatial discretization of the diffusion operator and the constraints, i.e. incompressibility and no-slip boundary condition at the obstacles. Note that in these equations the dependency on the design variables $\boldsymbol{g}$ has been indicated explicitly.

The above system of equations can be recast into the form given in equation (2.5.2) by introducing the state vector $\boldsymbol{q} = (\boldsymbol{u}, \boldsymbol{\phi})^T$. The temporal discretization of the governing equations (forward problem) is carried out using the implicit Crank-Nicholson method for the viscous terms and the explicit second order Adams-Bashforth scheme for the convective terms. The fully discrete forward equations in

matrix form are then

$$\begin{bmatrix} \frac{1}{\Delta t}\mathbf{M} - \frac{1}{2}\mathbf{L} & \mathbf{Q} \\ \mathbf{Q}^\dagger & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \phi \end{pmatrix} = \begin{pmatrix} \left(\frac{1}{\Delta t}\mathbf{M} + \frac{1}{2}\mathbf{L}\right)\boldsymbol{u}^n - \frac{3}{2}\mathbf{N}(\boldsymbol{u}^n, \boldsymbol{g}^n) + \frac{1}{2}\mathbf{N}(\boldsymbol{u}^{n-1}, \boldsymbol{g}^{n-1}) \\ 0 \end{pmatrix} + \\ + \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \text{ for } 0 \le t \le T, \tag{2.17.1}$$

$$\boldsymbol{u}(0) - \boldsymbol{u}_0(\mathbf{g}) = 0. \tag{2.17.2}$$

This system of equations is solved by following the fractional step method [47, 160, 44], the reader is referred to [195] for further details.

2.4.1 Discrete adjoint Navier-Stokes equations

Using the discrete form of the equations presented in (2.16), the discrete adjoint equations, following the formalism presented in (2.11) with the inner product (2.9), are

$$\mathbf{M}\frac{\mathrm{d}\tilde{\boldsymbol{u}}}{\mathrm{d}t} - \left(-\mathbf{L} + \frac{\partial \mathbf{N}^\dagger}{\partial \boldsymbol{u}}\right)\tilde{\boldsymbol{u}} - \mathbf{Q}\tilde{\boldsymbol{\phi}} + \frac{\partial J^\dagger}{\partial \boldsymbol{u}} = 0, \tag{2.18.1}$$

$$-\mathbf{Q}^\dagger\tilde{\boldsymbol{u}} + \frac{\partial J^\dagger}{\partial \boldsymbol{\phi}} = 0 \text{ for } 0 \le t \le T, \tag{2.18.2}$$

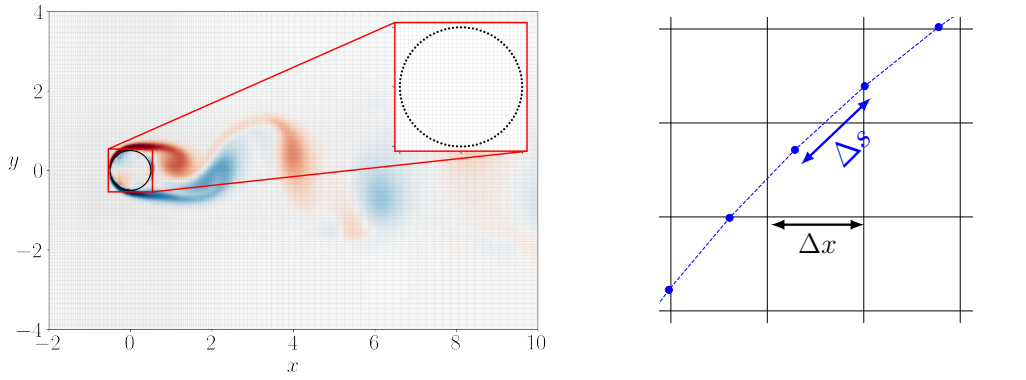
$$\text{and } \tilde{\boldsymbol{u}}(T) = 0. \tag{2.18.3}$$

The adjoint variable is $\boldsymbol{\lambda}^\dagger = (\tilde{\boldsymbol{u}}^\dagger, \tilde{\boldsymbol{\phi}}^\dagger)$, where $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{\phi}}$ are, respectively, the adjoint velocity field and adjoint pressure and localized forces. In the derivation, it has been taken into account that $\mathbf{M}$ is independent of time and the operators $\mathbf{M}$ and $\mathbf{L}$ are symmetric. The adjoint equations are again a system of algebraic differential equations and they are integrated backwards in time starting from $t = T$ once the forward (direct) solution has been computed. These equations have been implemented in the nonlinear numerical code, maintaining the temporal discretization used for the forward problem in equation (2.17). Note that the discrete nonlinear advection operator $\mathbf{N}(\boldsymbol{u}, \boldsymbol{g})$ in (2.16) is time dependent. Its discrete adjoint $\mathbf{N}^\dagger$ must be linearized at each time step around the direct solution and then used according to the chosen temporal discretization scheme.
Once the two equations (2.16, 2.18) are solved, the gradient is given by

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{g}} = \int_0^T \left(\frac{\partial J}{\partial \boldsymbol{g}} - \tilde{\boldsymbol{u}}^\dagger\left(\frac{\mathrm{d}\mathbf{b}_1}{\mathrm{d}\boldsymbol{g}} + \frac{\partial \mathbf{N}}{\partial \boldsymbol{g}}\right) - \boldsymbol{\phi}^\dagger\frac{\mathrm{d}\mathbf{b}_2}{\mathrm{d}\boldsymbol{g}}\right)\mathrm{d}t + \tilde{\boldsymbol{u}}^\dagger(0)\mathbf{M}\frac{\mathrm{d}\boldsymbol{u}_0}{\mathrm{d}\boldsymbol{g}}. \tag{2.19}$$

2.5 Application to flow configurations

The numerical code with immersed boundary capabilities, solving both the forward and the backward equations presented in the previous sections, has been tested and validated on different optimization problems. In this section we show the results obtained for two optimization problems, testing both steady and unsteady actuation.



(a) (b)

Figure 2.2: Flow around a cylinder at $Re = 200$. (a) Vorticity contours and immersed boundary points. (b) Schematic of Lagrangian points (blue) versus the background Cartesian grid.

The difference between the two applications relies in the time dependency of the control variables $\boldsymbol{g}$. These test cases have been chosen for the purpose of validating and testing the performance of the parallel-in-time algorithm, that will be presented in chapter 3.

2.5.1 Drag reduction – steady actuation

The first system considered here is the two-dimensional flow around a cylinder at $Re = 200$. This regime is characterized by the appearance of a von Karman vortex street, the vorticity contours of the uncontrolled case are shown in figure 2.2a, along with the immersed boundary points defining the surface of the cylinder. The spatial discretization of the domain must be uniform and refined in the proximity of the cylinder to ensure the stability of the solution [178], and to prevent penetration of streamlines the arc length Δs between the Lagrange points are selected to be approximately equal to the size of the neighboring Cartesian cells Δx, as shown in figure 2.2b. The evolution of the drag coefficient is shown in figure 2.3. As expected in this regime, after the initial transient, the drag coefficient converges to the average value of 0.032 with oscillations of amplitude 0.001 and a frequency corresponding to the shedding frequency of the cylinder $St = 0.19$. Once the simulation reaches steady state, $t \approx 41$, the optimization procedure is initiated. The actuation is obtained by inducing a blowing/suction control on the surface of the cylinder, allowing the vertical and horizontal velocities at each Lagrangian point to act as actuators. Since the velocity on the Lagrangian points are mathematically described by a delta function, a smoothing filter is applied on the resulting velocity profiles to avoid discontinuities between neighboring points. The discrete cost functional to minimize is then defined as the sum of the squares of the drag coefficient $C_d = 2f_x$, where f_x are the dimensionless boundary forces in the streamwise direction. In addition, a

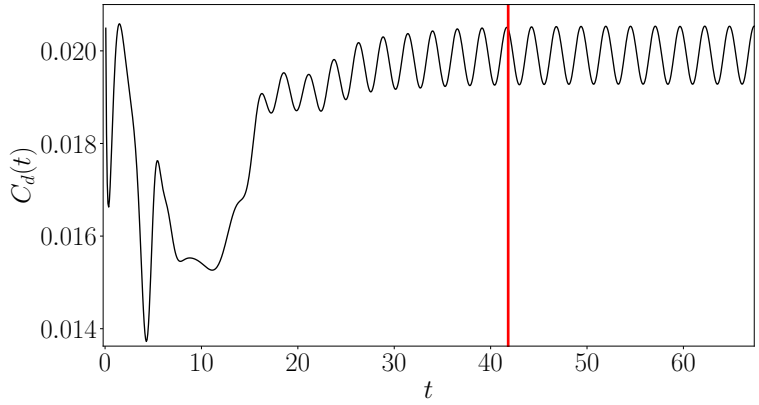


Figure 2.3: Drag coefficient C_d, with the red line signifying the location where actuation starts, $t = 41s$.

penalization of the control variables $\mathbf{g}$ is added to keep their value sufficiently small, resulting in

$$\mathcal{J}(\mathbf{q}, \mathbf{g}) = \frac{1}{T} \int_0^T C_d^2 \, \mathrm{d}t + \alpha \, \|\mathbf{g}\|^2. \tag{2.20}$$

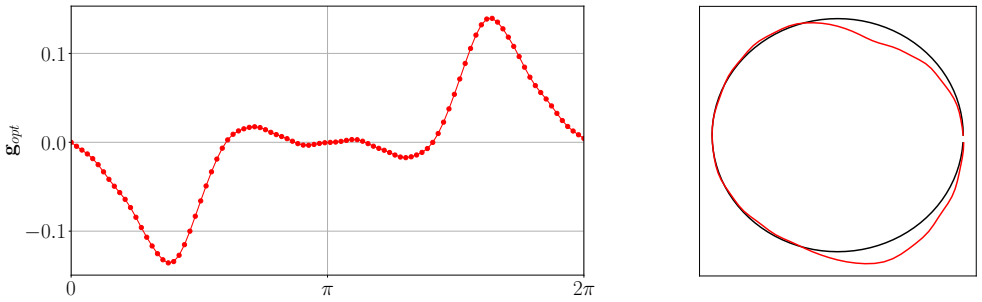
The solution of the optimization process is presented in figure 2.4. The resulting velocity profile in figure 2.4a shows blowing and suction effects on the back of the cylinder, for a better visualization the same profile is shown along the surface of the cylinder in figure2.4b. The actuation results in a reduction of the objective functional $J(\mathbf{q}, \mathbf{g})$ and of the amplitude of the drag coefficient oscillations in time $C_d(t)$, respectively presented in table 2.1 and figure 2.5. The actuation reduces not only the average drag but also the amplitude of the oscillations around this average. There is a slight change in the frequency of the oscillations due to the steady actuation.

	$\mathbf{g} = 0$	$\mathbf{g} = \mathbf{g}_{opt}$
$\mathcal{J}(\mathbf{q}, \mathbf{g})$	3.987	1.700

Table 2.1: Value of the cost functional before and after the actuation process.

2.5.2 Total pressure loss – unsteady actuation

One of the main causes of aerodynamic losses in turbomachinary is due to vortices generated at the tip of the blades as they interact with the outside casing [56]. These vortices are promoted by the pressure gradient on the surface of the blades, as well as the relative motion between the blade tip and the casing of the the rotor. In this section, we present an optimization problem with unsteady actuation inspired by this phenomenon. The objective is to determine whether modifications of the



(a) (b)

Figure 2.4: The resulting optimal actuation. (a) Profile of the boundary velocities; (b) Profile of the boundary velocities (red) projected on the cylinder surface as a reference (black).

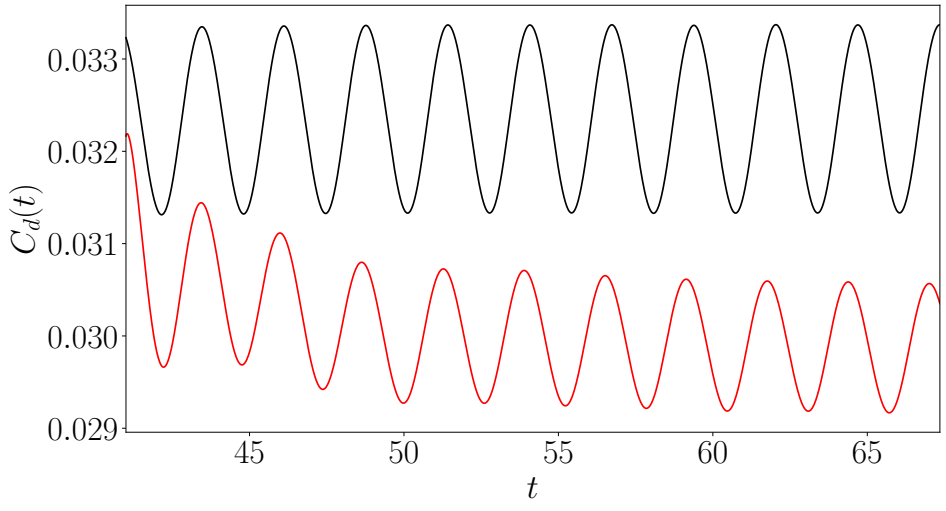


Figure 2.5: The resulting cost functional. Drag coefficient C_d without the actuation (black) and with the optimal control (red).

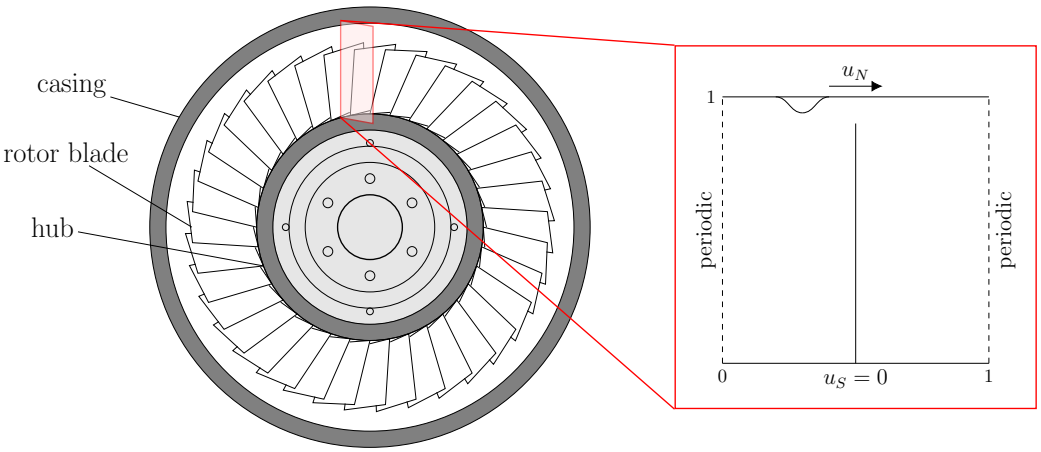


Figure 2.6: Schematic of a rotor of an axial compressor, and the numerical domain in the red box with one blade (periodic boundary conditions on the horizontal axis simulate a series of blades). The upper boundary moves with a constant velocity and the dynamic roughness is added on its profile.

casing of the rotor (in the form of a small perturbation or roughness) would be able to suppress the generation of such vortices. This actuation is of relevance since it is reproducible in real-scale applications with relatively minimal effort.

Figure 2.6 shows the rotor of an axial compressor that rotates with a certain angular velocity around the axis. In order to simplify the description of this complex phenomenon, the domain considered for the numerical simulations consists of a section of a single unit of the casing and blade geometry, defined as a square domain with periodic boundary conditions in the streamwise direction, as shown in the sketch of figure 2.6. The blade is represented by a vertical wall defined by the immersed boundaries and covers 90% of the vertical direction. Periodic boundary conditions in the streamwise direction simulate the series of blades. Curvature effects are neglected in this setup. The relative motion between the casing and the blade is enforced by applying a uniform horizontal velocity on the top boundary (casing) instead of the Lagrangian points to avoid the update of the location of points at each time-step in the case of moving immersed boundaries.

Perturbations on the casing are replicated by adding a roughness on the top boundary. In order to reduce the dimension of the optimization problem, the roughness is considered to have the shape of a Gaussian function, and its width and amplitude is optimized via the optimization process. In addition, the roughness is assumed to have the characteristics of a dynamic roughness element.

Dynamic roughness elements have been investigated by [99] and further explored by [138] and [95]. In this approach, the roughness elements are modeled using linearized boundary conditions representing oscillating bumps with simple geometries. The roughness element is approximated by the streamwise and wall-normal velocity

distribution as

$$\begin{aligned} u(x,y,t)|_w &= -H(x,y,t)U_0'(x)\,, \\ v(x,y,t)|_w &= \dot{H}(x,y,t)\ \ , \end{aligned} \tag{2.21}$$

where H denotes the height of the roughness, varying in space and time, $U_0'(x)$ represents the wall-normal derivative of the mean velocity profile at the wall, and $\dot{H}$ is the derivative of the height with respect to time. Since, the shape of the roughness is Gaussian, its height H can be expressed as

$$H(x,y,t) = \varepsilon \exp\left(-\frac{(x-\mu(t))^2}{2\sigma^2}\right)\,, \tag{2.22}$$

where ε is the height of the curve's peak, σ is its standard deviation or width, and $\mu(t)$ denotes the center of the Gaussian moving with the upper boundary. The control vector is then defined as $\mathbf{g} = [\varepsilon, \sigma]$.

The objective of this optimization process is to extract the most optimal modulation on the casing which will result in a maximum reduction of the average pressure loss across the blade, or in other words, finding the optimal value for $\mathbf{g}$ that minimizes the cost functional

$$\mathcal{J}(\mathbf{q},\mathbf{g}) = \frac{1}{T}\int_0^T \overline{\Delta p} + \alpha\left(\|u_N(\mathbf{g},t)\|^2 + \|v_N(\mathbf{g},t)\|^2\right)\mathrm{d}t\,, \tag{2.23}$$

where $\overline{\Delta p}$ is the spatial average of the total pressure loss around the blade and $u_N(\mathbf{g},t)$ and $v_N(\mathbf{g},t)$ are the respective horizontal and vertical boundary conditions imposed on the top of the domain, depending on the control parameters $\mathbf{g}$. The time interval chosen for the optimization is one period of the roughness motion. At $t = 0$ the Gaussian is centered at $x = 0$, and at $t = T$ it is at $x = 1$.

	$\mathbf{g} = [0, 0]$	$\mathbf{g} = [0.095, 0.184]$
$\mathcal{J}(\mathbf{q},\mathbf{g})$	2.000	1.982

Table 2.2: Pressure loss minimization, listing the cost functional before and after the actuation.

The effect of the unsteady actuation is highlighted in table 2.2 by comparing the resulting pressure loss to the uncontrolled setup, resulting in a 1% improvement. The resulting effect on the evolution of the average pressure gradient across the blade, $\overline{\Delta p}$, is also shown in figure 2.7. This figure shows that the presence of the roughness exerts a large influence on the oscillation frequency of the average pressure signal. While in the uncontrolled setup the pressure oscillated with a frequency proportional to the relative difference between the width of the domain and the gap between the blade and the casing surface, in the case with unsteady control the pressure oscillates with the same frequency as the passing of the roughness element. In addition, the actuation increases the amplitude of the oscillations fourfold. The maximum amplitude is reached when the roughness is at $x = 0.35$, shortly before

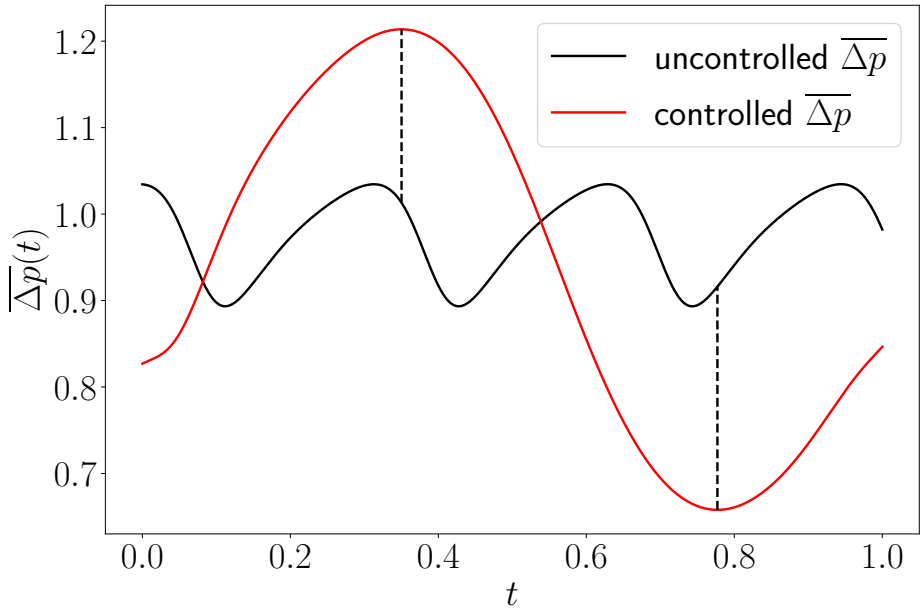


Figure 2.7: Evolution of the average pressure loss in time, comparison of the base case (black line), and the case with actuation (red line).

reaching the gap. The lowest amplitude, however, is encountered when the roughness is at $x = 0.77$ before reaching the outlet. Due to the respective frequencies of the pressure signal in the controlled and uncontrolled scenarios, the optima of both curves nearly coincide. Therefore, in order to assess the quantitative differences of the two cases, vorticity and pressure distributions are compared at the maximum and the minumum of the curve describing the pressure loss of the controlled regime, as highlighted by the dashed lines in figure 2.7.

The spatial distributions of pressure and vorticity profiles are displayed in figures 2.8 and 2.9, at the maximum and the minimum, respectively.

At the time where the average pressure is at its maximum (corresponding to figure 2.8(d)), a large difference between the pressure distribution on the two sides of the blade is noticeable, whereas in figure 2.9(d), corresponding to the point in time where the average pressure is at its minumum, the pressure distribution across the blade is more homogeneous. When comparing the uncontrolled cases in the same two time instances, the main difference is the value of the pressure at the tip of the blade, which appears stronger in the figure 2.8(c) than in figure 2.9(c). However, the pressure distribution along the two sides of the blade seems mostly unaffected. The reason for an overall change in the pressure distribution can be deduced by analyzing the distribution of the vorticity inside the domain. In the uncontrolled case, a vortex is developed on the tip of the blade, which ultimately sheds with a frequency similar to the frequency of the average pressure loss. Due to the periodic

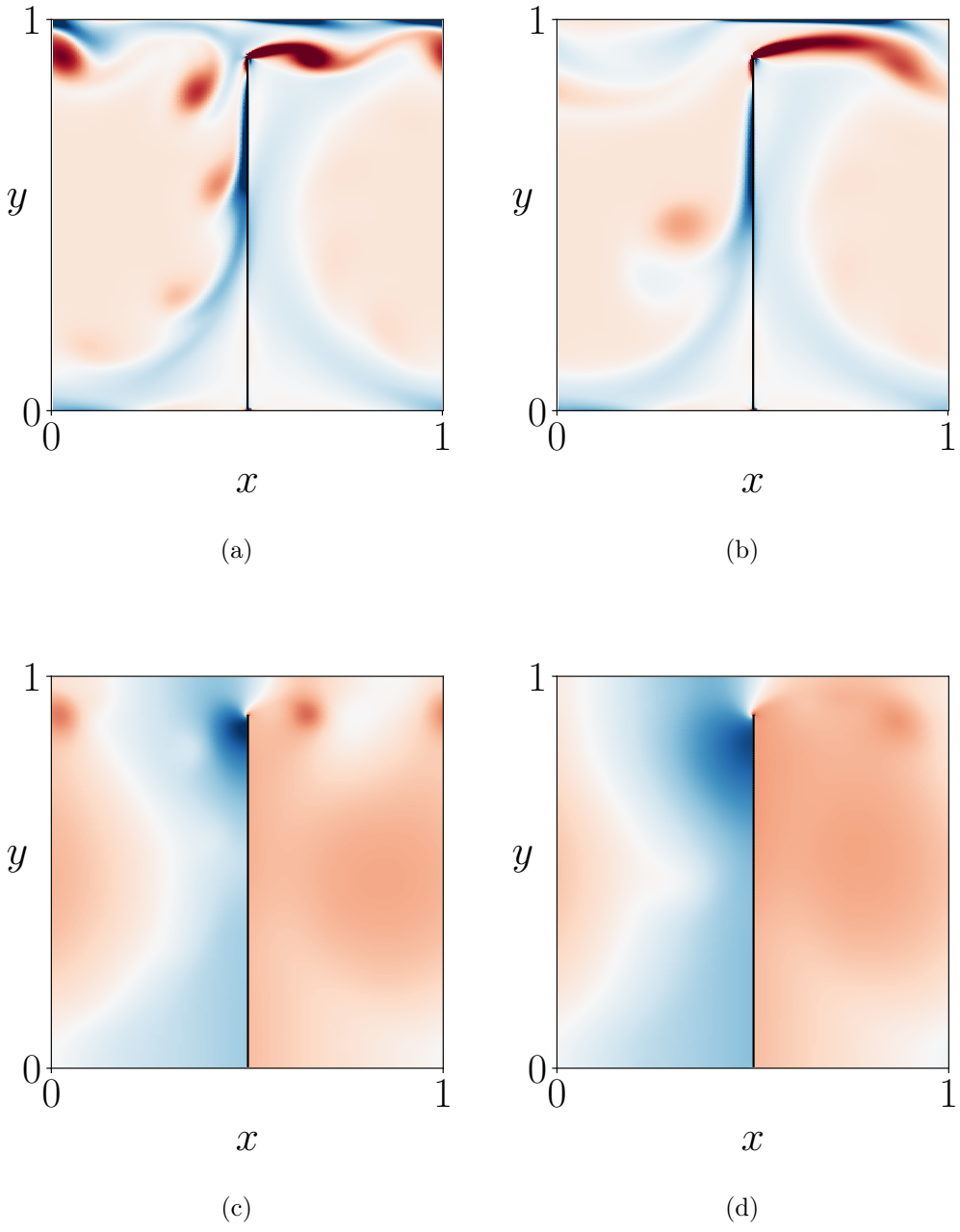


Figure 2.8: Vorticity profile and pressure distribution for $t = 0.35$: (a) uncontrolled vorticity, (b) controlled vorticity, (c) uncontrolled pressure, (d) controlled pressure.

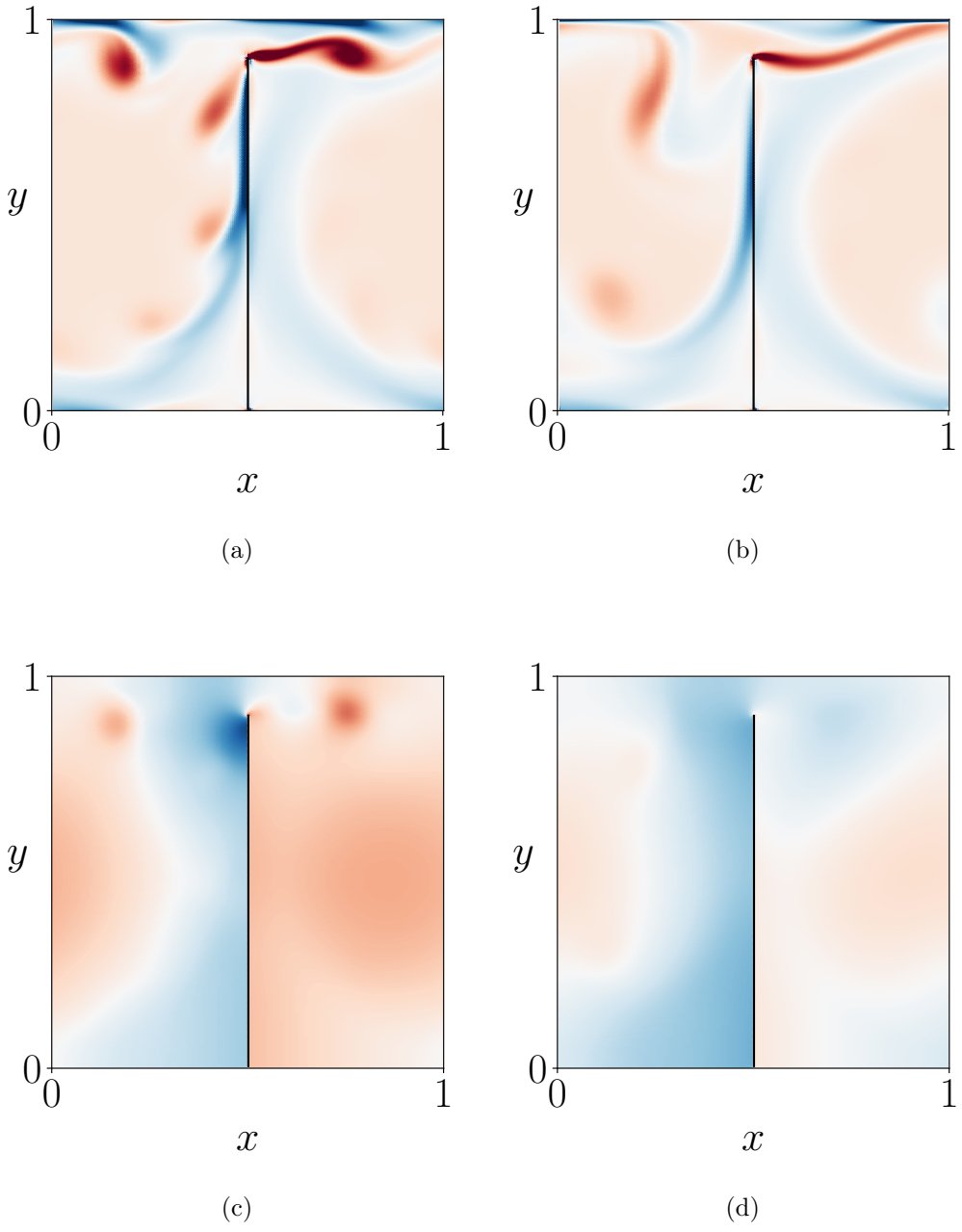


Figure 2.9: Vorticity profile and pressure distribution for $t = 0.77$: (a) uncontrolled vorticity, (b) controlled vorticity, (c) uncontrolled pressure, (d) controlled pressure.

boundary conditions, this vortex reenters the domain and creates a series of vortex pairs developing on the suction side of the blade. The shedding process is entirely suppressed by the presence of the roughness. Instead, vorticity is created as the roughness enters and leaves the domain, respectively, leading to a larger vortical structure.

2.6 Conclusions

We investigated the application of gradient-descent algorithms to unsteady fluid optimization problems and introduced adjoint-based methods, proven to be the most efficient way to extract the gradient for such systems. The derivation of the adjoint equations for gradient extraction was developed using control theory notions, to finally get to the discrete form of the optimality system of equations. The gradient extraction procedure has been implemented in our Navier-Stokes with immersed boundaries solver, adding control and optimization capabilities to the original numerical code. The adjoint solver has been tested and validated on both steady and unsteady actuation of two-dimensional fluid flows. Despite the relatively small size of the fluid systems addressed here and the total absence of checkpointing strategies to reduce the computational cost, the total time spent to reach the optimum and complete the optimization procedure was quite significant compared to the time of one function evaluation. The principal cause for this large time to solution is that each iteration of the optimisation algorithm is at best twice the cost of a full CFD calculation (forward integration). In unsteady cases, the use of checkpointing algorithms increases this cost to almost three times. In addition, the overall time to solution increases proportionally, since these operations (forward and backward integrations) happen sequentially. While the cost of the calculation can not be circumvented (adjoint equations need to be solved in order to have access to the derivative information), strategies can be adopted to reduce the overall time to solution, such that, ideally, once the forward problem has reached the final time, the gradient is also available. In the next chapter we will introduce parallel-in-time techniques and we propose an algorithm applied to the direct Navier-Stokes equations and its adjoint system.

Chapter 3

Parallel in time algorithm of adjoint-based techniques

Contents

3.1 Introduction 35
3.2 State-of-the-art and background 36
3.2.1 ParaExp algorithm 36
3.2.2 Hybrid serial-direct-parallel-adjoint algorithm 38
3.3 Exponential time integrators 43
3.3.1 Projection-based methods 44
3.3.2 Polynomial interpolation based methods 45
3.3.3 Projection on Navier-Stokes equations 45
3.4 Results 46
3.4.1 Temporal Energy Growth – examining the exponential time integrator 46
3.4.2 Drag reduction – steady actuation 48
3.4.3 Total pressure loss – unsteady actuation 51
3.5 Conclusions 54

3.1 Introduction

In the previous chapter we introduced adjoint-based methods and described the procedure of using them in order to extract the gradient in the context of the control and the optimization of incompressible flows. This process can be prohibitively slow due to sequential nature of the method (direct-adjoint loop), the unsteadiness of the system and the nonlinearity of the Navier-Stokes equations used here. For nonlinear unsteady forward problems, the adjoint equations are built by performing a linearization of the nonlinear operators around the direct solution, meaning that the full direct solution has to be stored in memory for the whole time interval considered. To reduce this considerable memory cost, checkpointing algorithms can be used, to save the direct solution just at certain checkpoints in time and solve the direct-adjoint loop between them [82]. While reducing the memory cost, the use of checkpointing in unsteady cases increases the computational costs nearly by a factor of three. Different strategies can be adopted to speed up the calculation of the optimization loop, such as reduced order models or spatial parallelisation.

However sometimes these techniques can not be used or the benefit obtained by their employment is not significant.
In this chapter we introduce parallelisation in time as an alternative to speed up the resolution of the direct-adjoint loop. In section §3.2 we give a brief overview of parallel-in-time methods and in particular we focus on the *ParaExp* algorithm and the hybrid serial-direct-parallel-adjoint algorithm introduced by Skene *et al.*.
Finally, modification to the existing algorithms due to the use of incompressible Navier-Stokes equations are also described. In particular, exponential integrators and the projection needed to include the incompressibility constraint are introduced in section §3.3. In addition, the resulting parallel in time algorithm is applied to the optimization problems already showed in the previous chapter. The performance of the parallel algorithm is evaluated for a single gradient extraction.

3.2 State-of-the-art and background

Time-parallel integration has been an active area of research, which started with the pioneering work of Nievergelt [151]. A brief account of development and implementation of various parallel-in-time algorithms can be found in [75] and [157]. The existing space-time parallel methods can be divided into four main categories: (i) methods based on multiple shooting, (ii) methods based on domain decomposition and waveform relaxation, (iii) methods based on multigrid, and (iv) direct time-parallel methods. While most of these efforts concentrated on accelerating the integration of the direct(forward) simulations, some efforts also considered incorporating a direct-adjoint optimization procedure, such as Maday *et al.* [131] and Skene *et al.* [193]. Here, we will concentrate on accelerating the adjoint equations using a parallel-in-time approach introduced by Gander and Güttel [73]. While the applicability of this parallel-in-time methodology to adjoint-based optimization has been analyzed by Skene *et al.* [193], and multiple possible algorithms have been suggested, the system of equations considered here add multiple layers of intricacy which have not been addressed in previous studies. One such challenges is due to the complexity of the problem, i.e. the nonlinear unsteady Navier-Stokes equations, compared to Burgers' equation studied previously. However, the main difficulty is due to the algebraic-differential nature of the governing equations (owing to the divergence-free constraint), which makes the application of the exponential time integrator nontrivial and also leads to an algebraic formulation of the adjoint equations. For the parallel-in-time strategy to be relevant for real scale applications, these underlying issues need to be properly addressed, thus motivating the work presented here.

3.2.1 ParaExp algorithm

The *ParaExp* algorithm, proposed by Gander and Güttel [75] belongs to the category of the direct time-parallel methods. It is based on an overlapping time-domain

decomposition. Considering a linear initial value problem

$$\dot{\boldsymbol{q}}(t) = A\boldsymbol{q}(t) + \mathbf{A}(t), \quad \boldsymbol{q}(0) = \boldsymbol{q}_0, \quad t \in [0, T], \tag{3.1}$$

where A is a linear time independent square operator, $\boldsymbol{q}$ is the state variable and $\mathbf{A}$ is a source term. The time domain is decomposed into smaller segments of size ΔT and the linear initial-value problem is separated into subproblems on overlapping time intervals. In particular, the initial-value problem is split into

- the homogeneous problem:

$$\dot{\boldsymbol{q}}_H(t) = A\boldsymbol{q}_H(t), \quad \boldsymbol{q}_H(0) = \boldsymbol{q}_0, \tag{3.2}$$

- and the inhomogeneous problem:

$$\dot{\boldsymbol{q}}_I(t) = A\boldsymbol{q}_I(t) + \mathbf{A}(t), \quad \boldsymbol{q}_I(0) = 0, \tag{3.3}$$

The solution of the original problem (3.1) is then obtained by the superposition of the solutions of the two subproblems as

$$\boldsymbol{q}(t) = \boldsymbol{q}_I(t) + \boldsymbol{q}_H(t). \tag{3.4}$$

When solved in parallel, the inhomogeneous component of the problem with zero initial condition is simultaneously solved in each segment. Once the proper boundary conditions at the end of each time segment are evaluated, the homogeneous problem is then integrated up to the final time T, this process is illustrated schematically in figure 3.1. For each processor we solve:

- the inhomogeneous problem, for $j = 1, \ldots, p$, as:

$$\dot{\boldsymbol{q}}_{I,j}(t) = A\boldsymbol{q}_{I,j}(t) + \mathbf{A}(t), \quad \boldsymbol{q}_{I,j}(T_{j-1}) = 0, \quad t \in [T_{j-1}, T_j] \tag{3.5}$$

- and the homogeneous problem, for $j = 1, \ldots, p$

$$\dot{\boldsymbol{q}}_{H,j}(t) = A\boldsymbol{q}_{H,j}(t) \tag{3.6}$$

$$\boldsymbol{q}_{H,j}(T_{j-1}) = \begin{cases} \boldsymbol{q}_0, & t \in [T_0, T] \quad \text{if } j = 1; \\ \boldsymbol{q}_{I,j}(T_{j-1}), & t \in [T_{j-1}, T] \quad \text{otherwise.} \end{cases} \tag{3.7}$$

The complete result consists of the superposition of the final solutions of each segment, as illustrated in figure 3.1, and given by

$$\boldsymbol{q}(t) = \boldsymbol{q}_{I,k}(t) + \sum_{j=1}^{k} \boldsymbol{q}_{H,j}(t), \quad t \in [T_{k-1}, T_k]. \tag{3.8}$$

Integrating the homogeneous problem in time can be expensive and time consuming, therefore, fast time integrators such as, exponential time integrator, are essential in

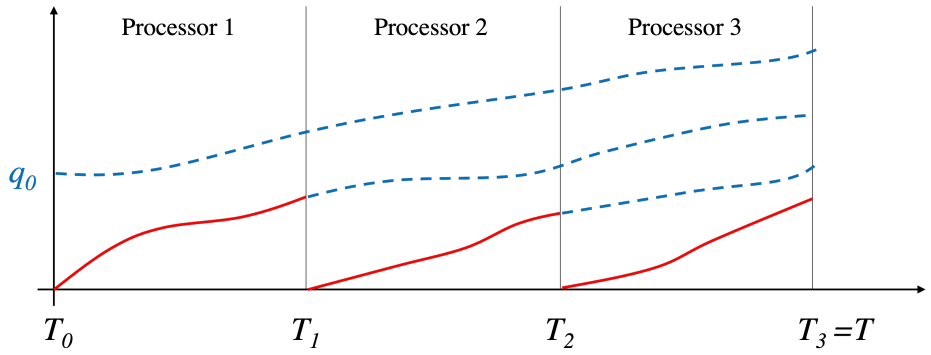


Figure 3.1: Overlapping time decomposition of an initial-value problem into four inhomogeneous problems with zero initial guess (solid red curves) and four homogeneous problems (dashed blue curves), the latter are exponentially propagated. The solution of the original problem is obtained by summation of all these curves [73].

order to speed up the process. The exact solution of the homogeneous initial-value problem in equation (3.2) solved by the exponential time integrator is then,

$$\boldsymbol{q}_H(t) = \exp(tA)\boldsymbol{q}_0. \tag{3.9}$$

To benefit from this method, the computation of the matrix exponential and the propagation of the solution in time has to be faster than the integration of the inhomogeneous subproblems. More details on exponential integrators and the approximation of the matrix exponential will be discussed in section §3.3.

Among the existing algorithms, *ParaExp* shows great promise due to the following advantages: this method performs particularly well if the existing inhomogeneity is difficult to integrate, which is a common scenario in complex unsteady flows. In addition, it allows the use of any existing serial time integration method. Moreover, this direct method is non-iterative and requires a single communication between processors at the end of the algorithm. As a result, the achieved parallel efficiency is higher than the maximal achievable parallel efficiency of the (Krylov-enhanced) parareal algorithms, and in particular the algorithm by Farhat *et al.* [65] for linear initial-value problems, which require more than a single iteration in general. Due to these advantages, this method is employed in this study for time-parallelism of the adjoint equation. Although *ParaExp* has been mainly applied to linear systems, an extension of the algorithm to a simplified nonlinear problem is now also available [74].

3.2.2 Hybrid serial-direct-parallel-adjoint algorithm

Parallel-in-time algorithms have been mostly employed in accelerating the primal (forward) problem. Some applications to the optimization procedure is also available, in particular, in the recent work of Skene *et al.* [193], where algorithms are

proposed based on the linear *ParaExp* algorithm developed by Güttel [73] and its extension to nonlinear PDE by Kooij [113].

In their study Skene *et al.*, investigated various strategies of accelerating both the forward (nonlinear) problem and the backward (linear) problem by using various combinations of *ParaExp* for the linear problems and its nonlinear adaptation by Kooij [113]. In particular, they proposed two algorithms to accelerate the resolution of the optimization problem raising from nonlinear forward problems. The first algorithm solves the nonlinear problems in parallel using the method of Kooij. The nonlinear term is linearised using an iterative procedure and its average Jacobian is added as a forcing term to improve the stability of the equation and ensure convergence. Once the nonlinear equation is linearised, the *ParaExp* algorithm is used for the time integration. The theoretical scaling analysis conducted by Skene *et al.* proved that the the iterations required to converge the nonlinear forward problem, slows the algorithm down when few processors are used. This cost can be overcome however by increasing the number of processors. In addition, depending on the type of nonlinearity, the forward problem might not reach convergence at all. Alternatively, the second algorithm (hybrid) solves the nonlinear direct equation in series and concentrates the parallelization on the linear adjoint. This algorithm on the other hand reaches the maximum speedup with a smaller number of processors. Therefore, in this work, we concentrate on accelerating the linear part of the optimization algorithm concerned with time integration of the adjoint equations with algebraic constraint, resulting in a hybrid serial-direct-parallel-adjoint algorithm. For this purpose, as illustrated in figure 3.2, the direct problem is solved in serial, while accounting for appropriate time partitioning, and the adjoint problem is split into its homogeneous and inhomogenous components. The resulting inhomogeneous algebraic adjoint equation, in general form, solved backward in time by processor $p \in \{N, \dots 1\}$ on $[T_p, T_{p-1}]$, is

$$\frac{\partial J}{\partial \boldsymbol{q}} - \boldsymbol{\lambda}_{I,p}^{\dagger} \frac{\partial \mathbf{F}}{\partial \boldsymbol{q}} + \frac{\partial}{\partial t}\left(\boldsymbol{\lambda}_{I,p}^{\dagger} \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}}\right) = 0 \tag{3.10.1}$$

$$\boldsymbol{\lambda}_{I,p}(T_p) = 0. \tag{3.10.2}$$

Once the inhomogeneous equation is integrated down to T_{p-1} by processor p, this processor initializes and solves the homogeneous problem, given below, for the time partition $[T_{p-1}, 0]$,

$$-\boldsymbol{\lambda}_{H,p}^{\dagger} \frac{\partial \mathbf{F}}{\partial \boldsymbol{q}} + \frac{\partial}{\partial t}\left(\boldsymbol{\lambda}_{H,p}^{\dagger} \frac{\partial \mathbf{F}}{\partial \dot{\boldsymbol{q}}}\right) = 0, \tag{3.11.1}$$

$$\boldsymbol{\lambda}_{H,p}(T_p) = \boldsymbol{\lambda}_{I,p}(T_p). \tag{3.11.2}$$

Finally, the solution of the full adjoint problem is given by adding the contribution from all the processors. Note that all processors except for $p = 1$ solve the homogeneous problem. Once processor p solves the direct problem up to T_p the last time solution is communicated as an initial condition to the next processor, which will continue integrating the forward problem further in time. In the meantime,

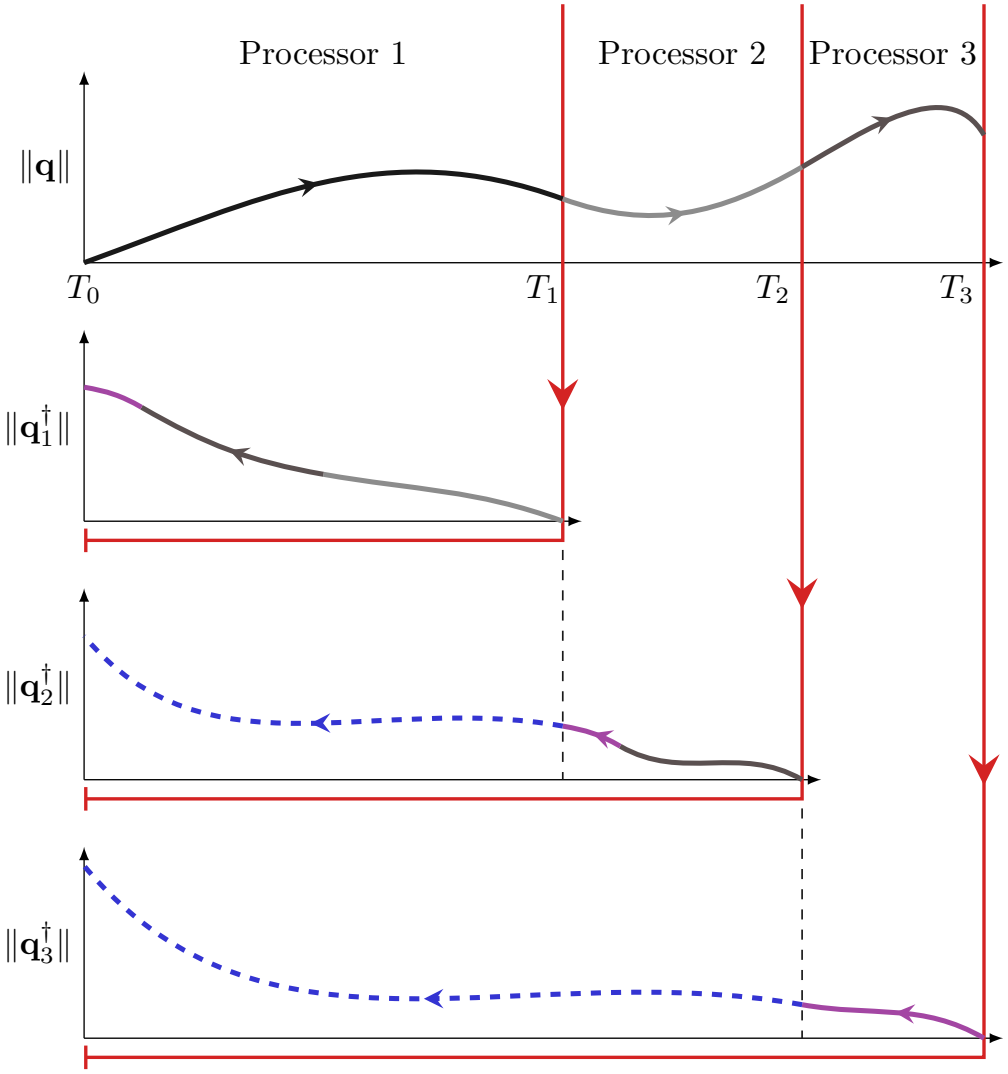


Figure 3.2: A schematic of a direct-adjoint loop with three processors, using a parallel-in-time procedure. Solid lines follow the forward evolution of the direct equation (top) and backward integration of the inhomogeneous adjoint equations on the three processors (bottom). Once the direct and inhomogeneous adjoint equations are solved, the direct solution is communicated (red lines) to each processor to initialize the homogeneous adjoint equation. This equation is then solved up to T_0 (dashed blue lines) by each processor.

processor p solves the inhomogeneous adjoint problem (with zero initial condition) backward in time, as demonstrated in (3.10.2).

The application of this parallel-in-time strategy to the problem of interest to this work, presented by equation (2.18), results in the following inhomogeneous and homogeneous set of equations (for a generic processor p) respectively,

$$\mathbf{M}^{\dagger}\dot{\tilde{\boldsymbol{u}}}_{I,p} + \left(\mathbf{N}_u(\boldsymbol{u})^{\dagger} - \mathbf{L} + \mathbf{A}^{\dagger}_{1,u}(\boldsymbol{u},\boldsymbol{g})\right)\tilde{\boldsymbol{u}}_{I,p} + \mathbf{Q}\tilde{\boldsymbol{\phi}}_{I,p} + \mathbf{J}^{\dagger}_{u} = 0 \tag{3.12.1}$$

$$\mathbf{Q}^{\dagger}\tilde{\boldsymbol{u}}_{I,p} - J^{\dagger}_{\phi}\tilde{\boldsymbol{\phi}}_{I,p} = 0 \;\text{ for }\; T_{p-1} \leq t \leq T_p, \quad \text{and} \quad \boldsymbol{\lambda_I}(T_p) = 0, \tag{3.12.2}$$

and

$$\mathbf{M}^{\dagger}\dot{\tilde{\boldsymbol{u}}}_{H,p} + \left(\mathbf{N}_u(\boldsymbol{u})^{\dagger} - \mathbf{L}\right)\tilde{\boldsymbol{u}}_{H,p} + \mathbf{Q}\tilde{\boldsymbol{\phi}}_{H,p} = 0 \tag{3.12.3}$$

$$\mathbf{Q}^{\dagger}\tilde{\boldsymbol{u}}_{H,p} = 0 \;\text{ for }\; 0 \leq t \leq T_{p-1}, \quad \text{and} \quad \boldsymbol{\lambda}_{H,p}(T_{p-1}) = \quad \boldsymbol{\lambda}_{I,p}(T_{p-1}), \tag{3.12.4}$$

A non-uniform time partitioning is used in order to ensure the simultaneity of the resolution of the direct problem and the adjoint inhomogeneous problems and prevent waiting time between processors. The analytic expression for the time partition is given by

$$T_p = T\left(\frac{1-\left(\frac{k}{k+1}\right)^{p}}{1-\left(\frac{k}{k+1}\right)^{N}}\right). \tag{3.13}$$

In this equation T_p is the final time of processor p, T is the total final time, N is the number of processors and $k = \tau_I^{\dagger}/\tau_I$ is the ratio between the time taken per time unit for the inhomogeneous adjoint and the direct solve. Further details on the time partitioning can be found in [193]. When the direct problem is solved by the last processor N, the inhomogeneous equations are almost completely solved on the antecedent time partitions. The non-uniform time partitioning assigns a smaller time interval to the last processor, so that the time to solve the inhomogeneous adjoint on this time partition is ensured to be short. The initial conditions needed to solve the homogeneous equations is then made available to each processor to integrate the homogeneous problem until $T = 0$. The total time to solution is therefore,

$$T^N = T\tau_I + (T_N - T_{N-1})\,\tau_I^{\dagger} + T_{N-1}\tau_H^{\dagger}, \tag{3.14}$$

where the terms are the time needed to solve the direct equation on the full time interval, the inhomogeneous adjoint on the last time partition $[T_N, T_{N-1}]$ (purple line in figure 3.2) and the homogeneous adjoint on $[T_{N-1}, 0]$ by the last processor. The resolution of the adjoint problem requires the knowledge of the direct state at each time step. While the inhomogeneous equation is solved on the same time partition as the forward problem and maintains the same discretization scheme, the homogeneous equation is solved on $[0, T_{p-1}]$. Due to the time partitioning described in figure 3.2, the generic processor p does not have access to the full direct solution up until $T = 0$. This solution must therefore be distributed to all processors,

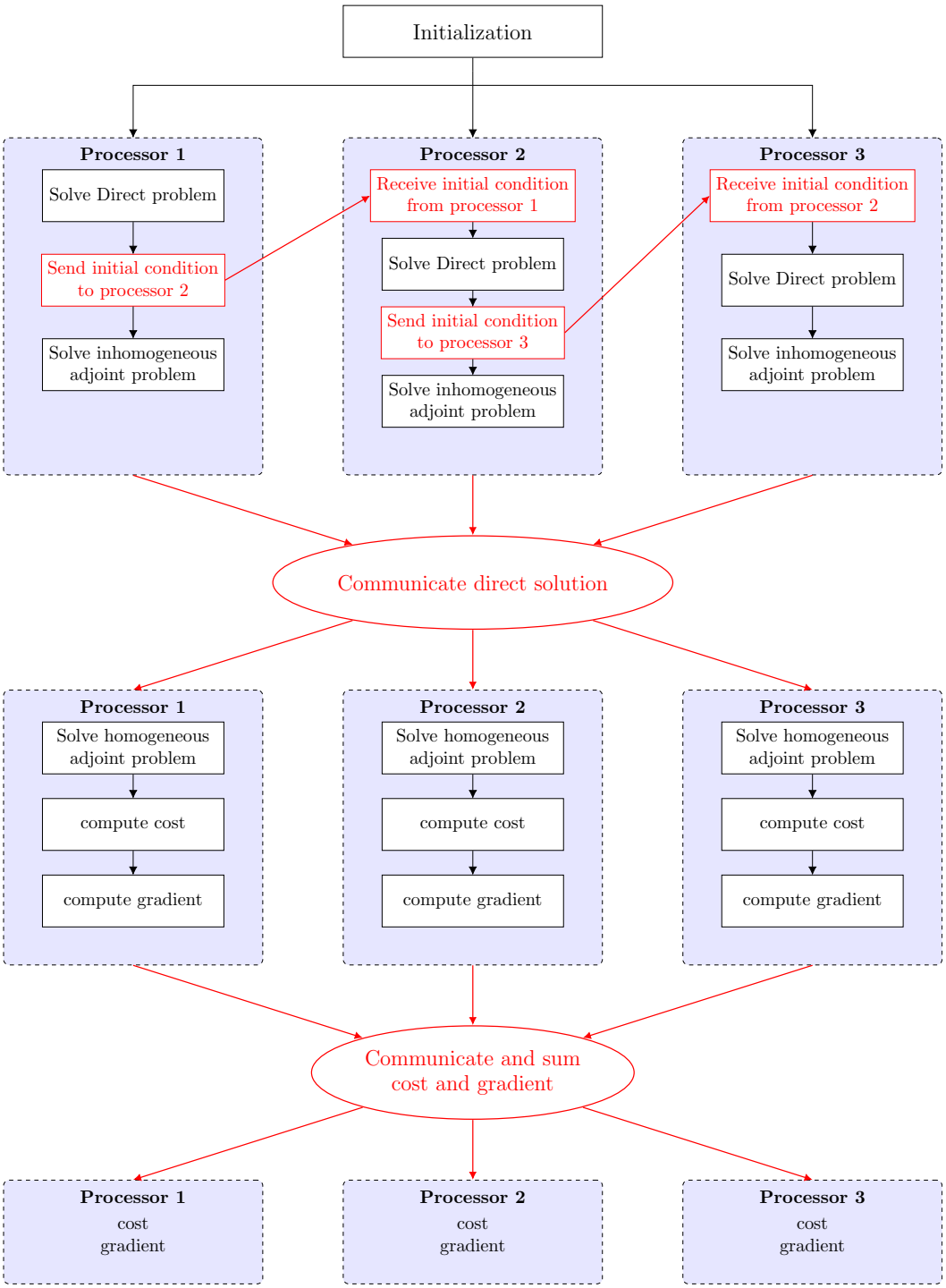


Figure 3.3: A schematic of the algorithm highlighting the communication between processors. Red lines represent the communication operations.

resulting in an unavoidable increase of time to solution. To reduce this one-time cost, each processor communicates a fraction of the direct solution to all. The submatrices are then stacked together and used by each processor to perform a linear interpolation on the coarser exponential time grid. The $N-1$ homogeneous adjoint equations are solved simultaneously and their solution is then stack to the corresponding inhomogeneous adjoint solution. The original *ParaExp* computes the total solution as a superposition of the solution computed by each processor, however, in cases where the size of the problem is large, the resulting communication can become rather expensive. When applied to the adjoint equation, the knowledge of the total solution is required only to compute the gradient. As a result, the partial gradients can be computed locally by each core using the optimality condition (2.19) and finally distributed to all processors and summed. This approach is employed both for the gradient and the value of the cost functional and reduces the size of the communication considerably, a schematic of the communications is given in figure 3.3.
The efficiency of this parallel-in-time algorithm relies mainly on the speed of the time integrator used for solving the homogeneous adjoint problems, the last term of (3.14). Using the same time integrator as that of the inhomogeneous equations leads to an unavoidable increase of the resolution time. A way to overcome this problem is to use an exponential integrator to solve the homogeneous adjoint problems.

3.3 Exponential time integrators

As mentioned in the previous section, the bottleneck of time acceleration using the proposed parallel-in-time procedure is the cost associated with integrating the homogeneous problem on each processor. Exponential time integrators can be employed to reduce this cost due to their superior accuracy with a minimal number of time-steps, and are directly applicable since the solution of the homogeneous problem can be expressed analytically in terms of the matrix exponential of the state-matrix. Remaining in the context of this work, we consider the linear initial value problem (3.1) and its exact solution

$$\boldsymbol{q}(t) = \boldsymbol{q}_0 e^{-tA} + \int_0^t e^{-(t-\tau)A}\mathbf{A}(\tau)\mathrm{d}\tau. \tag{3.15}$$

Different methods were proposed in the literature to solve this equation numerically by approximating the integrand. An interesting review of exponential integrators for semi-linear and linear problems can be found in the work of Hochbruck [91] and Minchev [140]. The homogeneous form of equation (3.1), and in our case, the homogeneous adjoint introduced in (3.11.1), reduces to

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} &= \mathbf{A}\boldsymbol{q}, \\ \boldsymbol{q}(0) &= \boldsymbol{q}_0, \end{aligned} \tag{3.16}$$

and its solution can be expressed in terms of the matrix exponential of the state matrix as

$$\boldsymbol{q} = \exp\left[(\Delta t)\,\mathbf{A}\right]\boldsymbol{q}_0, \tag{3.17}$$

where $\mathbf{A}$ is the system state matrix (or Jacobian). Here, matrix $\mathbf{A}$ is assumed to be time independent. Direct access to $\mathbf{A}$ becomes challenging as the dimension of the discretized problem increases. In such cases the exponential matrix is approximated directly using two categories of methods: projection-based methods and polynomial-interpolation-based methods.

3.3.1 Projection-based methods

Projection based methods approximate the matrix exponential on the orthogonal, lower dimensional Krylov subspace of $\Delta t\mathbf{A}$ [182, 183], defined by the power iteration as

$$\mathcal{K}_m(\mathbf{A}, \boldsymbol{q}) \equiv \text{span}\left\{\boldsymbol{q}, \mathbf{A}\mathbf{q}, \mathbf{A}^2\boldsymbol{q}, \ldots, \mathbf{A}^{m-1}\boldsymbol{q}\right\}, \tag{3.18}$$

where m is the dimension of the subspace and $m \ll n$, with n the dimension of the operator $\mathbf{A}$, and $\boldsymbol{q}$ is the vector propagated to the solution for Δt. This subspace is ill-conditioned, and vectors become linearly dependent with increasing m. The orthogonal basis of the Krylov subspace $\mathcal{K}_m$ can be computed using the Arnoldi algorithm [8] that creates a set of orthogonal basis that span $\mathcal{K}_m$. The procedure starts by building $v_1 = q/\|q\|$ and iteratively produces an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ of the Krylov subspace $\mathcal{K}_m$ via a modified Gram-Schmidt iterative process. The lower-dimensional representation of $\mathbf{A}$ is expressed as $\mathbf{A} \approx \mathbf{V}_m\mathbf{H}_m\mathbf{V}_m^T$,

Algorithm 1 Arnoldi algorithm

```
Given v_1, with ||v_1|| = 1
for j = 1, ..., m do
    for i = 1, ..., m do
        h_ij = (A v_j, v_i)
    end for
    w_j = A v_j − Σ_{i=1}^{j} h_ij v_i
    h_{j+1,j} = ||w_j||
    if h_{j+1,j} = 0 then
        stop
    end if
    v_{j+1} = w_j / h_{j+1,j}
end for
```

where $\mathbf{H}_m = \mathbf{V}_m^T\mathbf{A}_m\mathbf{V}_m$ is the $m \times m$ upper Hessenberg matrix computed in the Arnoldi algorithm, representing the projection of matrix $\mathbf{A}$ onto $\mathcal{K}_m$. Finally, since $\mathbf{V}_m^T(\Delta t\mathbf{A})\mathbf{V}_m = \Delta t\mathbf{H}_m$, and the Krylov subspace associated to $\mathbf{A}$ and $\Delta t\mathbf{A}$ are the same, the matrix exponential is approximated as

$$\exp(\Delta t\mathbf{A}) \approx \mathbf{V}_m \exp(\Delta t\mathbf{H}_m)\mathbf{V}_m^T, \tag{3.19}$$

Based on this, equation (3.17) can be reformulated in a matrix-free form as

$$\exp(\Delta t\mathbf{A})\boldsymbol{q} \approx \beta \mathbf{V}_m \exp(\Delta t \mathbf{H}_m)\hat{\mathbf{e}}_1. \tag{3.20}$$

where $\beta = \|\boldsymbol{q}\|^2$, and $\hat{\mathbf{e}}_1$ is the first unit basis vector, as $\boldsymbol{q}$ is orthogonal to all the $\mathbf{V}_m$ except for the first column. For further details on the approximation of the matrix exponential operator using Krylov subspaces, refer to [182] and [90].

3.3.2 Polynomial interpolation based methods

Polynomial interpolation based methods, on the other hand, consist of Newton interpolation of the matrix exponential propagator, $\exp(\mathbf{A})\boldsymbol{q}$ or $\phi(\mathbf{A})\boldsymbol{q}$, where $\phi(z) = (e^z - 1)/z$ [33]. Once this function is obtained on a defined set of interpolation points $\{\xi_j\}_{j=0}^{N}$, the matrix exponential is approximated as, $\exp(\mathbf{A})\boldsymbol{q} = \mathbf{A}\phi(\mathbf{A})\boldsymbol{q}$. One of the critical aspect of this method is therefore the choice of the interpolation points, from which the convergence of the approximation depends. Leja points [16, 33, 34, 10, 171] are convenient in these applications because they guarantee maximal and superlinear convergence while being independent of the degree of interpolation. In other words, the degree of the interpolant polynomial can be increased without recomputing the previous terms. When the matrix exponential is large and sparse, the interpolation on Leja points is more convenient than Krylov projection methods due to a lower memory cost.

3.3.3 Projection on Navier-Stokes equations

The homogeneous algebraic adjoint equation presented in (3.11) has to be reformulated for an exponential time integrator to be applicable. An additional projection procedure needs to be devised to include the constraint in equation (3.11.1) and to reformulate the equation in the form of a partial differential equation presented in (3.16) (without a constraint). The divergence-free constraint of the incompressible formulation of the governing equations, along with the constraint formed by no-slip conditions on the immersed boundaries makes the application of such schemes non-trivial. To accommodate these constraints, the momentum equation must be reformulated [114]. To this end, the semi-discrete homogeneous adjoint system given in (3.12.3) and (3.12.4), for a generic processor p, with appropriate initial conditions, multiplied by $\mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}$ and differentiated with respect to time, gives

$$\mathbf{Q}^\dagger\dot{\tilde{\boldsymbol{u}}}_{H,n} = \mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\left(\mathbf{L} - \mathbf{N}_u(\boldsymbol{u}_n)^\dagger\right)\tilde{\boldsymbol{u}}_{H,n} + \mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\mathbf{Q}\tilde{\boldsymbol{\phi}}_{H,n}, \tag{3.21.1}$$

$$\mathbf{Q}^\dagger\tilde{\boldsymbol{u}}_{H,n} = 0. \tag{3.21.2}$$

Substituting (3.21.2) in (3.21.1) and solving for $\tilde{\boldsymbol{\phi}}_{H,n}$, results in

$$\tilde{\boldsymbol{\phi}}_{H,n} = -\left(\mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\mathbf{Q}\right)^{-1}\mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\left(\mathbf{L} - \mathbf{N}_u(\boldsymbol{u}_n)^\dagger\right)\tilde{\boldsymbol{u}}_{H,n}. \tag{3.22}$$

This formulation for $\tilde{\phi}_{H,n}$ is then used in (3.21.1) to get the projection of the momentum equation onto the subspace defined by the constraints (3.21.2). The homogeneous adjoint equation (3.12.3) and its projected initial condition are then rewritten as

$$\begin{aligned} \dot{\tilde{\boldsymbol{u}}}_{H,n} &= \mathbf{P}\left(\mathbf{L} - \mathbf{N}_u(\boldsymbol{u}_n)^\dagger\right)\tilde{\boldsymbol{u}}_{H,n}, \\ \tilde{\boldsymbol{u}}_{H,n}(T_p) &= \mathbf{P}\,\tilde{\boldsymbol{u}}_{In}(T_p). \end{aligned} \tag{3.23}$$

with the associated projection operator

$$\mathbf{P} = \left[\mathbf{M}^{\dagger-1} - \mathbf{M}^{\dagger-1}\mathbf{Q}^\dagger\left(\mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\mathbf{Q}\right)^{-1}\mathbf{Q}^\dagger\mathbf{M}^{\dagger-1}\right]. \tag{3.24}$$

Using this projection (3.23) allows the application of an exponential integrator. It should however be noted that the right-hand side of equation (3.23) depends on time, whereas the system matrix $\mathbf{A}$ is considered to be independent of time. In this case, the exact solution of the linear initial value problem (3.17) should be

$$\boldsymbol{q} = \exp\left(\int_0^t \mathbf{A}(\boldsymbol{u}^n(\tau))\mathrm{d}\tau\right)\boldsymbol{q}_0. \tag{3.25}$$

In order to use the exact solution given in (3.17), the transconjugate advection operator $\mathbf{N}_u(\boldsymbol{u}_n)^\dagger$ is updated at each time iteration and is considered piecewise-constant in time. The accuracy of the method is further discussed in section 3.4. The computation and storage of the projection operator in (3.3.3) is not practical, therefore a fractional step method [47] has to be added into the Arnoldi algorithm everytime the product $\mathbf{A}\mathbf{v}_j$ is computed (see algorithm 1). To avoid large memory cost, the approximation of the matrix exponential is done by Krylov projection, that allows a matrix-free implementation.

3.4 Results

In this section, the convergence of the Krylov-based exponential time integrator is investigated, then the performance of the parallel-in-time adjoint algorithm is presented using the cases presented in section §2.5: (i) drag reduction of a flow around a cylinder, and (ii) reducing pressure loss across a blade using boundary control. In the first case, steady actuation is imposed using immersed boundary forces and in the second case, unsteady actuation is performed at a domain boundary, introducing new challenges for the parallel-in-time algorithm, which will be discussed in the following. The efficiency of the parallelization and the decrease of the computational cost has been evaluated for a single gradient extraction (one direct-adjoint loop).

3.4.1 Temporal Energy Growth – examining the exponential time integrator

Before discussing the performance of the parallel-in-time optimization algorithm, the convergence of the Krylov-based exponential time integrator is assessed and

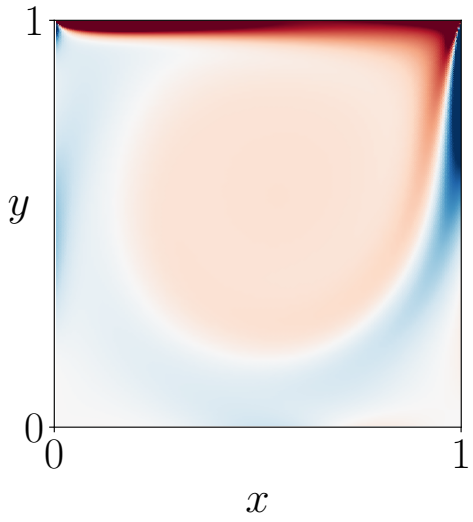


Figure 3.4: Temporal energy growth: vorticity field in a lid driven cavity at $Re = 1000$.

compared to the original explicit second-order Adams-Bashforth method. As mentioned in the previous sections, the exponential time integrator is applied to the homogeneous part of the equation only. Therefore, the optimization problem has to be designed such that the resulting adjoint equation becomes homogeneous and can be integrated by either integration method independently. Optimizing the initial condition of the forward flow solver, such that an energy norm, $G(T)$, at a selected time T is maximized offers an ideal test case. In this approach, the optimization procedure aims at maximizing the ratio between an energy norm at $t = T$ and $t = 0$, resulting in the following cost functional

$$\mathcal{J}(\boldsymbol{q}, \boldsymbol{g}) = \frac{(\boldsymbol{g} \cdot \boldsymbol{g})}{(\boldsymbol{q}(T) \cdot \boldsymbol{q}(T))}, \tag{3.26}$$

where $\boldsymbol{g} = \boldsymbol{q}(0)$ is the control parameter (the optimal initial condition), and the energy is computed with a simple ℓ_2-norm, using the full state vector. Since the cost functional does not depend on the time evolution of the forward problem, the first term of the adjoint equation (2.11) is equal to zero, resulting in a homogeneous problem. The exponential integrator can therefore be used to propagate the entire equation backward in time. The optimization is applied to a case of a lid driven cavity at $Re = 1000$ on a very short time interval in order to analyze the convergence of the two time integrators, the vorticity field of the flow is represented in figure 3.4. It should be noted, however, that due to the presence of the convection term (a nonlinear operator), the solution of the adjoint problem is dependent on the forward problem. In order to remove the errors due to the forward integration of the

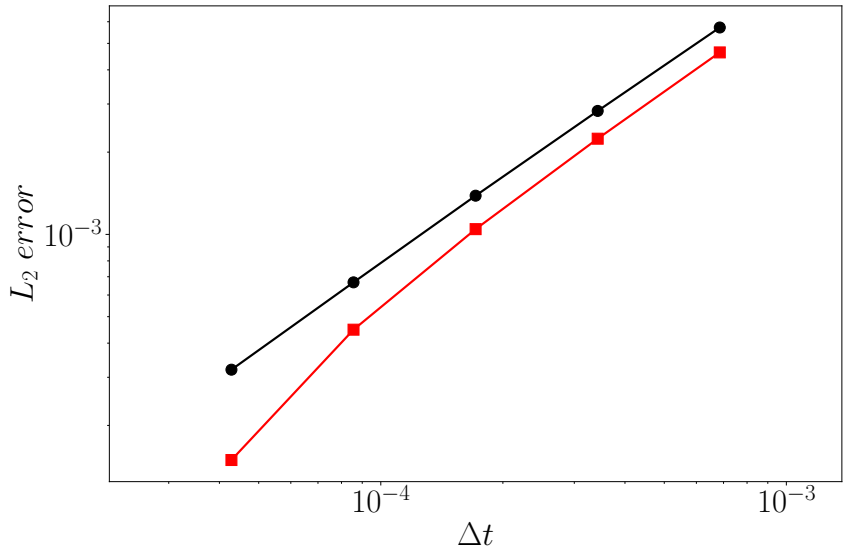


Figure 3.5: Convergence history of time integrators: (black) Explicit Adam-Bashforth method; (red) Krylov-based exponential Euler method.

primal problem from the convergence of the adjoint equation, the primal problem is integrated once using the most refined Δt. This solution is then stored and used to linearize the advection operator for the adjoint equations when the time step is increased. The convergence results obtained using the exponential integrator and the explicit Adam-Bashforth method are displayed in figure 3.5. The ℓ_2-error is computed using the difference between the final solution of the adjoint equation $q^\dagger(0)$, from each time integration, and the reference solution, from a highly refined simulation. The optimization is performed without time parallelism, in order to focus on the time integrator. The errors due to the parallel-in-time algorithm will be assessed in the following sections. This figure shows the explicit Adam-Bashforth method converging with a first-order slope, which is less than the expected second-order convergence. This deterioration is caused by the presence of immersed boundary forces and the use of fractional step to solve the direct problem. As expected, the exponential integrator shows a convergence rate that increases when Δt is refined. This behavior allows the use of a coarser time-step without affecting the accuracy of the final solution.

3.4.2 Drag reduction – steady actuation

The parallel-in-time algorithm is applied on the flow around a cylinder at $Re = 200$, the optimization procedure and the results have already been shown in chapter 2, the vorticity contour is shown again in figure 3.6 for reference. In this chapter, we study the performance of the temporal parallelization on a single direct-adjoint

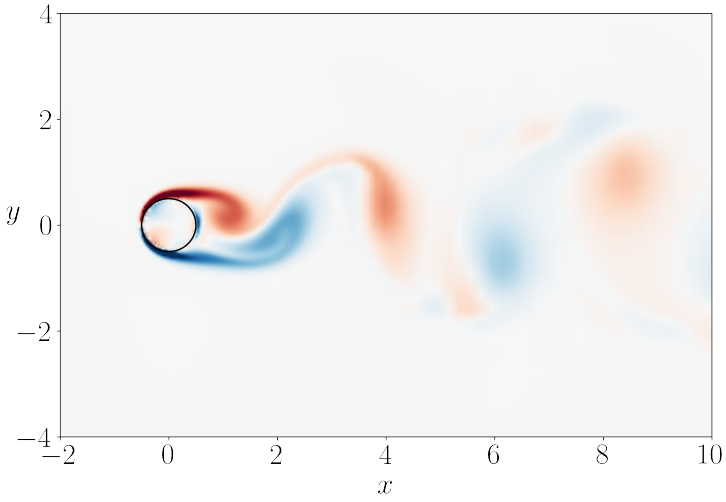


Figure 3.6: Drag reduction: vorticity field of the flow around a cylinder at $Re = 200$.

loop. The performance of the parallel-in-time algorithm is shown in figure 3.7. Here, the total time to solution is represented for both the serial and the parallel cases. For a direct comparison, no checkpointing algorithm is included (the solution of the forward integration at each time-step is accessible on memory). As a result, the time necessary to integrate the adjoint equation "in serial" is the same as that of the forward problem, leading to twice the cost. The figure shows that using the parallel-in-time algorithm, the time needed to compute the gradient converges to a value very close to the time needed for simply solving the direct problem, as the number of processors is increased. In other words, the value of the cost functional and the gradient are obtained almost at the same time as the end of the forward integration, resulting in a reduction of 94.5% of the total time of the optimization process. The efficiency of the parallel algorithm, however, depends on the size of the optimization problem. As mentioned in section 3.2.2, the time partitioning is non-uniform and each processor solves the direct problem and the inhomogeneous adjoint on a shorter time interval than the previous processor. Increasing the number of processors involved in the resolution of the system reduces the number of time steps assigned to the last threads, reaching a limit where the time partition is smaller than the minimum time-step needed to solve the equation. In the case of the cylinder, the maximum number of processors that can be used is $N = 8$. When eight processors are used, the algorithm assigns three time-steps to the last core, which is the minimum allowed to form the advection term in equation (2.16). For $N > 4$ the figure shows a loss of efficiency of the algorithm, beyond this point, the time needed to communicate the direct solution to each core overwhelms the time saved by time parallelism.

The accuracy of the resulting gradient using the parallel-in-time algorithm is shown in figure 3.8. The gradient obtained using the serial adjoint algorithm (with-

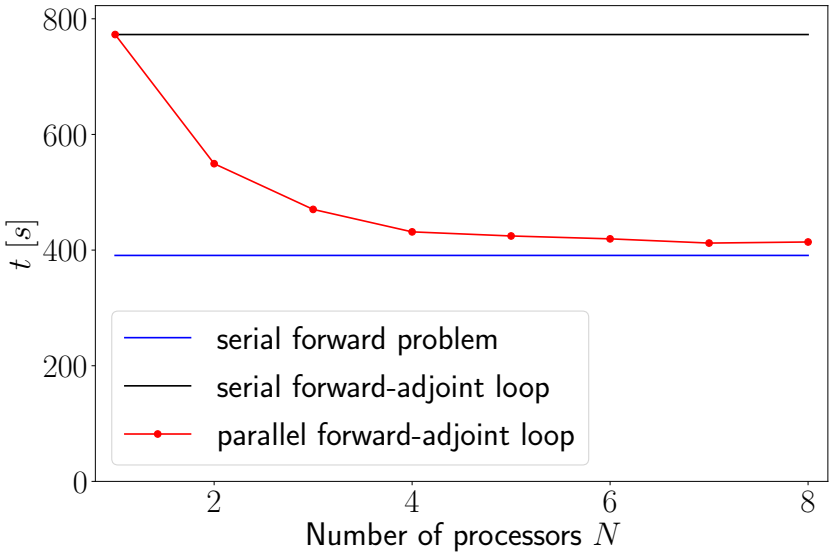


Figure 3.7: Performance of the parallel-in-time algorithm (red line) compared to the serial counterpart (black line) reported for a single iteration of the optimization loop, using steady control.

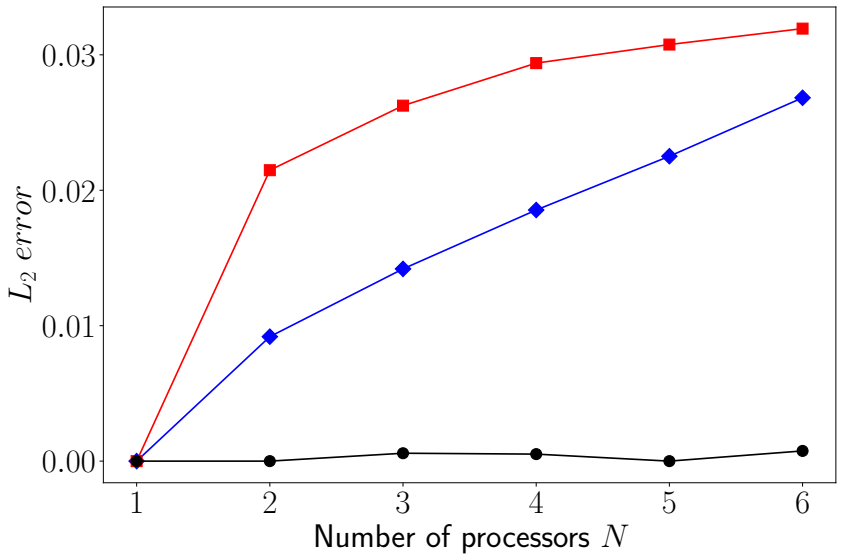
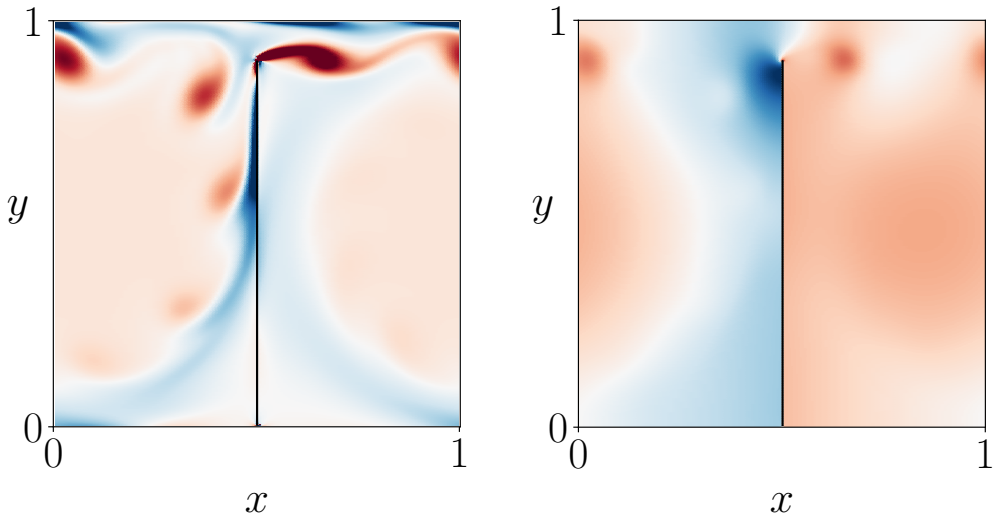


Figure 3.8: Accuracy of the estimated gradient computed for a single optimization loop: —, Adams-Bashforth method; —, exponential integrator without interpolation; — Exponential integrator with interpolation.

out the parallel-in-time treatment) is used as the reference in computing the error. Two major differences of the parallel-in-time implementation and its serial counterpart are that, firstly, in the parallel-in-time algorithm, the problem is separated into the homogeneous and inhomogeneous equations and an exponential integrator is used to propagate the solution of the homogeneous problem backward in time. In addition, in order to speed-up the adjoint loop, larger time steps are used to integrate the homogeneous problem. The latter step necessitates an interpolation of the forward problem from a finer resolution in time (used by the lower-order time integrator) on the coarser grid used by the exponential time integrator. This interpolation can impact the accuracy of the final optimization algorithm. In order to assess the ramification of each of these steps on the overall accuracy, three different implementations of the parallel-in-time algorithm are compared in figure 3.8. In the first implementation, highlighted by the black line in the figure, the homogeneous adjoint equations are solved using the same time integrator and time discretization as the inhomogeneous equations. Therefore, the error accrued due to the interpolation step and the change in the time integrator is eliminated. As expected, the final solution remains the same as the original gradient, independent of the number of processors. In the second implementation, denoted by the blue line, the homogeneous equations are solved using the exponential time integrator but the same time discretization as the inhomogeneous problem, removing the error due to interpolation. The exponential integrator is more accurate than its counterpart resulting in a small error between the two solutions. This error increases as the number of processors increase since a larger portion of the problem is solved using the exponential integrator. Finally, in the last implementation (the implementation suggested in this study), denoted by the red line, the homogeneous equations are solved using the exponential time integrator on larger time intervals compared to the inhomogeneous equations. This figure shows that the error due to interpolation (the difference between the blue and red curves) decreases when increasing the number of processors. Due to a non-uniform time partitioning, smaller portions of the time domain are integrated using the exponential time integrator as the number of processors increase, causing the error due to interpolation to saturate.

3.4.3 Total pressure loss – unsteady actuation

We consider again the optimization problem proposed in section §2.5.2, representing an approximation of the flow in a section of a single unit of the casing and blade geometry of a coaxial turbomachine. The flow is dominated by the presence of vortices generated at the tip of the blade due to the distribution of the pressure on the faces of the blade and the pressure difference across them (figure 3.9, these vortices represent a source of energy loss that we aim to minimize. The optimization is performed by introducing a shape modification on the surface of the casing acting as a roughness and modelled with a gaussian shape. The design variables $\boldsymbol{g}$ are then the width and the height of the roughness which moves with respect to the blade with uniform velocity. The design variables, and therefore the gradient, are



(a) (b)

Figure 3.9: Total pressure loss: vorticity profile and pressure distribution for $t = 0.35$. (a) uncontrolled vorticity, (b) uncontrolled pressure.

here time dependent, adding an additional dimension to the operators acting in the optimality system of equations presented in chapter 2.

While the resolution and the simultaneity of the forward-adjoint (inhomogeneous) loop is unchanged, the optimality condition (equation (2.19)) used to compute the gradient, is now time-dependent, increasing the size of the operators needed for the gradient extraction. The performance of the parallel-in-time algorithm evaluated for one iteration of the optimization algorithm, is shown in figure 3.10. As in the previous case, for $N > 4$, the time needed for the communication of the direct solution to all processors exceeds the time saved by the time partitioning and no improvement is accomplished by the parallel-in time procedure. The time to solution reduces towards a maximum of 58% of the total serial time. The magnitude of the gain obtained here is less significant than for the previous case, because of the use of unsteady control. Here, each processor builds the operators needed to solve this equation for its time partition, resulting in additional time. The time required to construct the operators is unavoidable and increases with the time interval chosen for the simulation. It is also the main cause of efficiency loss for the parallel-in-time algorithm with unsteady control. Nonetheless, the additional gain obtained by using the parallel-in-time algorithm is non-negligible, even in the presence of unsteady control.

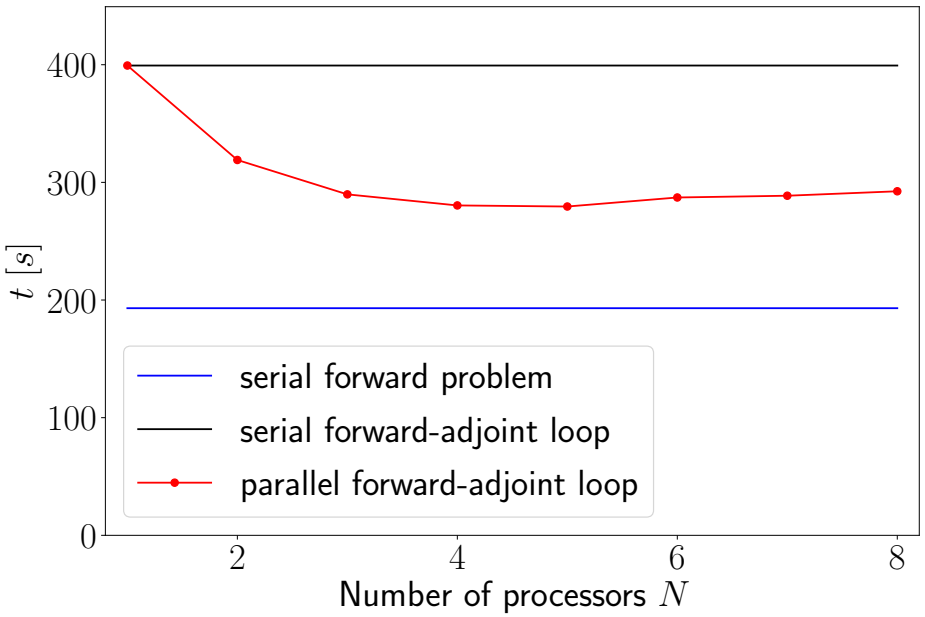


Figure 3.10: Performance of the parallel-in-time algorithm (red line) compared to the serial counterpart (black line) reported for a single iteration of the optimization loop, using unsteady control.

3.5 Conclusions

In this chapter we presented an algorithm for accelerating gradient-based optimization problems. The algorithm is the extension of the parallel-in-time algorithm for direct-adjoint loops by Skene *et al.* [193] to the two-dimensional Navier-Stokes equation with immersed boundaries. The pressure and boundary forces are treated by introducing a projection operator to allow the exponential integration of the linear homogeneous adjoint equations using Krylov subspace projection methods.
The performance of this method has been tested on two different optimization cases using steady and unsteady control for one gradient evaluation. In both cases the time to solution has been significantly reduced, following a trend consistent with the numerical and theoretical results derived by Skene *et al.*[193]. Better results have been observed for the steady control optimization. In this case, the time required to solve the adjoint-loop converged asymptotically to the time needed to solve the direct equation in serial, suggesting that the computation of the gradient can be obtained with a negligible additional penalty in overall time. The use of time-dependent control has proven to affect the efficiency of the parallel-in-time procedure. In this case, the gain obtained by using the proposed algorithm is appreciable, but less substantial.

Chapter 4

Model Reduction - projection based methods

Contents

4.1 Introduction 55
4.2 Projection-based model reduction 57
4.3 Snapshot proper-orthogonal-decomposition 58
4.4 Variable parameter ROM - Grassmann manifold 61
4.5 Galerkin and Petrov-Galerkin projection 66
4.6 Gauss-Newton with approximated tensors 68
4.6.1 Least-squares Petrov-Galerkin projection 68
4.6.2 Tensors approximation and hyper-reduction 69
4.7 Results 71
4.7.1 Reduced-order model - GNAT 71
4.7.2 Variable Parameter GNAT 75
4.8 Conclusions 81

4.1 Introduction

Numerical simulations of multiphysics and multiscale phenomena in fluid mechanics have advanced remarkably over the past decades. Complex physical processes, including among others multiphase and reactive flows, aero-acoustics, and turbulence, can now be simulated with an astonishing degree of fidelity and accuracy, and many industrial, technological, and fundamental problems have greatly benefited from computational sciences. However, in order to remain predictive, these computations rely on time consuming operations. In control and optimization applications, where quick access to the final solution is necessary, or many function evaluations are required, these simulations become too slow or costly, motivating the development and use of reduced-order models in their stead.

Projection-based model reduction methods are possible candidates for reducing the system of nonlinear equations, governing the evolution of the flow. However, in their basic form, such models scale with the dimensionality of the underlying high-fidelity simulation and are therefore efficient for primarily linear and steady problems. One possible approach is to leverage the data from the detailed simulation by employing data-decomposition techniques such as POD [192, 11, 92], DMD [185],

and greedy reduced basis [130, 126] and EIM [14, 45, 159] with its generalized version [128, 129, 21]. The advantage of the POD approach over alternate basis functions, for example the reduced-basis method [98], is the ability of the modes to adequately represent the physics of the system and to minimize sensitivities to operating conditions. Since this approach is based on a sequence of flow snapshots, it is equally applicable to experiments and numerical simulations. Through this procedure, the most energetic modes are extracted, which then form a hierarchical basis for describing the flow. Alternatively, POD has been also employed in Data Assimilation (DA) to reconstruct the flow from incomplete (gappy) numerical and experimental dataset [32, 128, 142], and sensor placement for control applications [7, 14, 45, 127, 207].

One of drawbacks of model reduction techniques is the commonly *ad hoc* truncation of the basis functions. Once truncated, the information in the discarded basis is lost, resulting in prediction errors and lack of robustness in the resulting model. As an alternative, Carlberg et al. [41] proposed the Gauss-Newton with approximated tensors GNAT method, which aims at optimising the model coefficients using the estimated error from the discarded basis. This reduction technique, is shown to perform satisfactorily in both structural-dynamics and CFD problems and is chosen as model reduction strategy for the following chapter.

Despite their increase in efficiency by dimensionally reducing the original problem, projection-based methods lack robustness when considering changes in the parameter space, where commonly a new ROM must be built for each new set of parameters. This is particularly disadvantageous in control applications, where design parameters are modified at each optimization iteration. A way to overcome this obstacle and to make the reduced-order model adaptable to parameter changes, is to interpolate and create a new set of orthogonal basis for the new operating point. However, the direct interpolation of orthogonal basis does not necessarily result in a new set of orthogonal basis. An alternative to standard interpolation is therefore an interpolation in a vector space, based on differential geometry and in particular on the Grassmann manifold and its tangent space [4, 5].

This interpolation strategy together with the GNAT methodology is employed in this chapter in order to construct an optimised POD basis, and its respective ROM for a variable operating condition.

The chapter is structured as follows: in sections §4.2 and §4.3 we introduce a brief overview of projection-based model reduction methods and snapshot POD, respectovely. The Grassmann manifold is briefly presented in §4.4. The modes obtained with the snapshot POD method are then used to build the GNAT-ROM in section §4.6, and finally the performance of the algorithm applied to an incompressible flow and its adaptativity are shown and analysed in the results section §4.7.

4.2 Projection-based model reduction

Model reduction techniques aim to reduce the dimensions of a high-dimensional problem using a lower dimensional approximation that captures the main physical features of the original system. The generic nonlinear dynamical system, governed by a semi-discretized system of equations, can be written as

$$\frac{\mathrm{d}\boldsymbol{q}(\boldsymbol{x},t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{q}(\boldsymbol{x},t)) \tag{4.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the time-dependent unknown state vector discretized on a spatial grid of dimension N_x and $\boldsymbol{f}(\boldsymbol{q}(t))$ is a generic nonlinear term. The physical system might include different variables k, generating a high-order dimensional coupled system of size $n = k \times N_x$, which will be referred to as a full order model (FOM). Projection-based model reduction methods rely on the assumption that the unknown state vector can be correctly approximated by projecting it on a low-dimensional affine trial subspace $\mathcal{W} \subset \mathbb{R}^n$ spanned by a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n_w}$, with $n_w \ll n$, whose columns $\boldsymbol{\Phi}_i \in \mathbb{R}^n$ with $i = 1, 2, \ldots, n_w$ are the time invariant basis for the subspace. The reduced-order state vector can then be formulated as a linear combination of this spatial basis, also known as Galerkin expansion

$$\widetilde{\boldsymbol{q}}(\boldsymbol{x},t) \approx \sum_{i=1}^{n_w} \boldsymbol{\Phi}_i(\boldsymbol{x}) a_i(t) = \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{a}(t), \tag{4.2}$$

where $a_i(t)$ are the expansion coefficients containing information on the time evolution of each element of this basis. Substituting (4.2) into (4.1), we obtain a reduced-order system of equations

$$\boldsymbol{\Phi}(\boldsymbol{x})\frac{d\boldsymbol{a}(t)}{dt} = \boldsymbol{f}(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{a}(t)). \tag{4.3}$$

This system is formed by n nonlinear equations and $n_w \ll n$ unknowns, and is hence overdetermined. Therefore, n_w constraints have to be introduced by enforcing the orthogonality of the residual of the system on a test subspace $\mathcal{L} \subset \mathbb{R}^n$ spanned by $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n_w}$. The projection of (4.3) on a chosen test subspace $\mathcal{L}$ leads to a system of n_w nonlinear equations and n_w unknowns, representing a generic reduced-order model (ROM) obtained by projection methods

$$\boldsymbol{\Psi}^T\boldsymbol{\Phi}(\boldsymbol{x})\frac{d\boldsymbol{a}(t)}{dt} = \boldsymbol{\Psi}^T\boldsymbol{f}(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{a}(t)), \tag{4.4}$$

where the inner product $\langle \boldsymbol{\Psi}, \boldsymbol{\Phi} \rangle = \boldsymbol{\Psi}^T\boldsymbol{\Phi}$ is used. It should be noted that in the case where the left subspace is chosen to have $\boldsymbol{\Psi} = \boldsymbol{\Phi}$, then this projection lies under the class of *Galerkin* projections, otherwise if $\boldsymbol{\Psi} \neq \boldsymbol{\Phi}$, it describes a *Petrov-Galerkin* projection. More details on the latter will be provided in section §4.5.

The construction of the ROM is made with an offline-online strategy. During the offline phase, the FOM data are acquired and the time-invariant spatial operators and basis needed in (4.4) are computed. This phase usually requires high computational costs. Afterwards, during the online phase, the nonlinear ROM (4.4) is integrated in time. A survey of projection-based method can be found in [15, 46, 156].

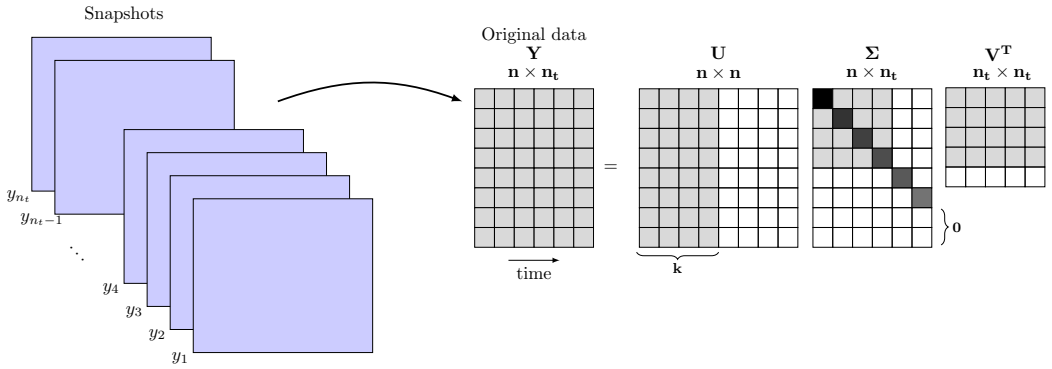


Figure 4.1: Schematic of snapshot-POD. Temporal snapshots are collected and then a singular value decomposition is employed to extract the POD basis $\mathbf{U}$ and the singular values $\mathbf{\Sigma}$.

4.3 Snapshot proper-orthogonal-decomposition

In most complex fluid systems, the dominant dynamical features of the flow can be encoded in low-dimensional patterns, known in the literature as *coherent structures*. Coherent structures are organized spatial features, and their existence in turbulent flows have been widely studied since the second half of the last century [36, 124]. The presence of these structures in fluid flows suggests the possibility of reducing the complexity and the dimension of a high-dimensional flow regime by expressing its dynamical behaviour as a combination of dominant dynamical features.

Proper orthogonal decomposition POD, also known as Karhunen-Loève expansion and Prinicipal-Component Analysis (PCA), is a linear procedure to extract uncorrelated modes from a flow. The resulting features (modes) are then used to define a basis that optimally approximates the high-dimensional data [19, 199]. In the method of *snapshots*, introduced by Sirovich [192], large dimensional discrete data, that can be obtained from both numerical simulations or experiments, are first collected and then used to identify optimal modes. A snapshot is the instantaneous flow vector field resulting from a complex fluid system. For an unsteady flow, we consider a set of samples $\mathbf{Y}$, forming the discrete state vectors $\boldsymbol{y}_i$, collected at different time instants t_i, $i = 1, \ldots, n_t$,

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_{n_t} \\ | & | & & | \end{bmatrix}, \tag{4.5}$$

with $\mathbf{Y} \in \mathbb{R}^{n \times n_t}$, and each column $\mathbf{y}_i = \mathbf{y}(t_i) \in \mathbb{R}^n$, and n the dimension of the high-dimensional system, tipically $n \gg n_t$. One possible procedure to extract orthogonal basis from a set of snapshots is the SVD [80], which provides a unique matrix decomposition of $\mathbf{Y}$,

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\dagger}, \tag{4.6}$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{n_t \times n_t}$ have orthogonal columns, called respectively left and right singular vectors, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n_t}$ is a diagonal matrix. The elements on the diagonal, called singular values, are nonnegative and ordered from the largest to the smallest. When $n \gg n_t$ the last rows of $\mathbf{\Sigma}$ are equal to zero. The SVD provides a low-rank approximation of the original data, where the rank is defined by the first k singular values and the respective vectors of the matrix decomposition while ignoring the rest. This approximation is proven to be optimal as it minimizes the ℓ_2-norm of the approximation error. In the case where k is equal to the number of non zero elements of $\mathbf{\Sigma}$, the approximation is exact. From a physical point of view, when SVD is applied on a set of fluid flow snapshots, $\mathbf{U}$ represents the orthogonal spatial modes that form the basis (coherent structures), the singular values $\sigma_i = diag(\mathbf{\Sigma})$ are the weights or the energies of each mode, and $\mathbf{V}$ contains the time history of the approximated data. A schematic of this methodology is given in figure 4.1.

POD is often used to generate an optimal orthogonal basis for the trial subspace $\mathcal{W}$ to approximate solutions of a nonlinear FOM. Following the method of snapshots previously introduced, the snapshots $\boldsymbol{q}_i = \boldsymbol{q}(\boldsymbol{x}, t_i)$ are collected at discrete time levels t_i, where $i = 1, \ldots, n_t$, during the offline phase. Usually the mean flow $\overline{\boldsymbol{q}}(\boldsymbol{x})$ is subtracted from the dataset to extract modal structures related to the fluctuations. The number of collected snapshots n_t has to be chosen to ensure that the important fluctuation of the flow are well resolved in time. This difference can be expressed as a linear combination of orthogonal spatial modes $\mathbf{\Phi}_j(\boldsymbol{x})$ and time dependent expansion coefficients $a_j(t)$

$$\boldsymbol{q}(\boldsymbol{x}, t) - \overline{\boldsymbol{q}}(\boldsymbol{x}) \approx \sum_{j=1}^{k} \mathbf{\Phi}_j(\boldsymbol{x}) a_j(t). \tag{4.7}$$

Here, $\boldsymbol{x}$ represents the spatial domain. The modes obtained from SVD are orthogonal with respect to the inner product

$$\iint_{\Omega} (\mathbf{\Phi}_i)^T \mathbf{\Phi}_j dV = \langle \mathbf{\Phi}_i, \mathbf{\Phi}_j \rangle = \delta_{ij}, \tag{4.8}$$

δ_{ij} is the Kronecker delta defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{4.9}$$

The orthogonality of the basis functions $\mathbf{\Phi}$ enable us to use the approximation given in (4.7). As a result, the temporal coefficients $a_j(t)$ can be computed as

$$a_j(t) = \langle \boldsymbol{q}(\boldsymbol{x}, t) - \overline{\boldsymbol{q}}(\boldsymbol{x}), \mathbf{\Phi}_j \rangle. \tag{4.10}$$

Finally, the high-dimensional flow can be approximated by using a finite number of POD modes (k) as

$$\widetilde{\boldsymbol{q}}(\boldsymbol{x}, t) = \overline{\boldsymbol{q}}(\boldsymbol{x}) + \sum_{j=1}^{k} \mathbf{\Phi}_j(\boldsymbol{x}) a_j(t), \tag{4.11}$$

reducing the size of the problem, n (dependent on the size of the grid and the number of variables), to the number of POD modes selected in the truncation, with $k \ll n$. The rank of the approximation k is usually determined using an energy criterion, such that the selected modes capture most of the energy of the flow, typically 99% of the total energy :

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n_t} \sigma_i^2} > 0.99, \tag{4.12}$$

where $k \leq n_t$.
These steps are summarized in algorithm 2.

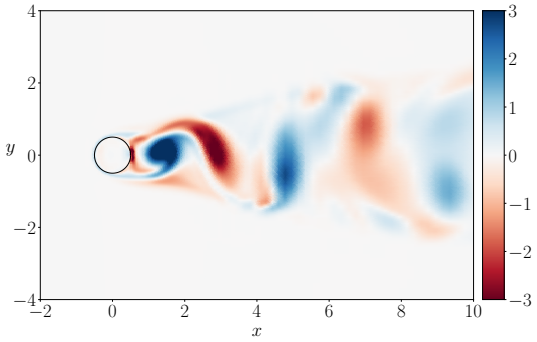
Algorithm 2 POD basis computation

Input Snapshot matrix $\mathbf{Y} \in \mathbb{R}^{n \times n_t}$
Output $\mathbf{\Phi}(\boldsymbol{x}) \in \mathbb{R}^{n \times n_w}$

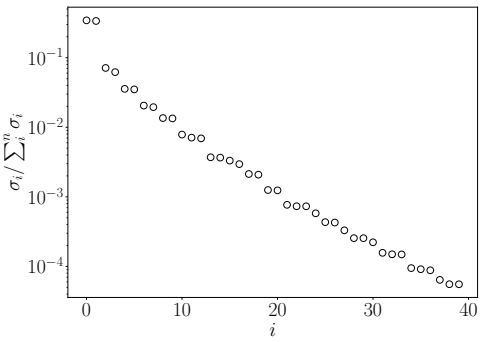
1: Compute SVD: $\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
2: Choose basis dimensions $n_w \in \{1, 2, \ldots, n_t\}$
3: Truncate basis $\mathbf{\Phi}(\boldsymbol{x}) \in \mathbb{R}^{n \times n_w}$

Algorithm 2 is applied to a flow over a circular cylinder in order to extract the dominant coherent structures through POD modes. This flow has been widely used in fluid mechanics to investigate bluff body flows both numerically and experimentally [54, 153, 152, 125]. Here we consider a two-dimensional flow over a cylinder at $Re = 200$, non-dimensionalised based on the diameter of the cylinder. At this regime the flow exhibits the von Kármán shedding wake. This time periodic wake is easily distinguishable and its time evolution suggests the existence of a limit-cycle and its associated coherent structures that can be represented using a low-dimensional dynamical system.

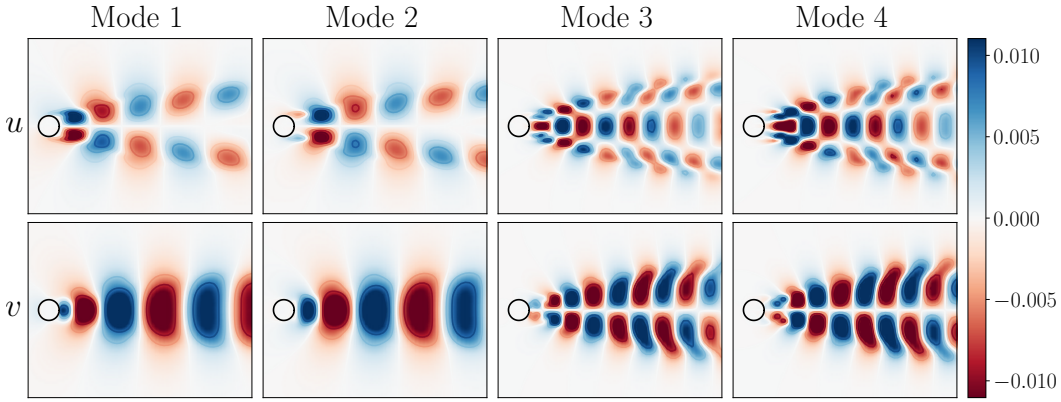
We collect 380 velocity snapshots over 2 shedding periods and we remove the mean flow. An illustration of a vorticity snapshot without the mean flow is reported in figure 4.2a. The POD modes are extracted from the snapshots representing the fluctuations $\boldsymbol{q}(\boldsymbol{x}, t) - \overline{\boldsymbol{q}}(\boldsymbol{x})$, by performing a singular value decomposition. Figure 4.2b shows the singular values in descending magnitude order. We observe that the first modes capture most of the total energy, suggesting that a small number of modes is sufficient to reconstruct an accurate approximation of the original high-dimensional flow (4.11). Due to the oscillatory nature of the flow the modes appear in pairs. For simplicity, figure 4.2c shows only the first 4 modes. The coherent structures of the flow fluctuations are visible in the spatial shape of the dominant modes and represent the symmetry and periodicity of the flow.



(a) Example of a vorticity snapshot $\boldsymbol{q} - \overline{\boldsymbol{q}}$.



(b) Singular values in descending order.



(c) Spatial POD modes for horizontal and vertical velocity.

Figure 4.2: Results of the SVD on a sample of snapshots. (a) an example of a vorticity snapshot, (b) singular values, (c) POD modes.

4.4 Variable parameter ROM - Grassmann manifold

The procedure described in the previous section might perform well for a set of parameters for which the data was collected and reduced. However these reduced-order models commonly lack predictability and therefore robustness as the given set of parameters change. In order to accommodate varying parameters, the reduction process can be applied to a range of parameters separately and a model can be deduced for each case. Then a dedicated interpolation procedure is required in order to ensure predictability and robustness across a continuous range of sample parameters for the construction of the basis functions. Standard interpolation of a set of orthogonal basis does not necessarily preserve orthogonality. Ansallem *et al.* [5] proposed an interpolation strategy based on the Grassmann manifold and its tangent space at a point in order to adapt ROMs to new operating points while preserving the orthogonality property of the interpolated basis, which will be adopted and demonstrated here.

Consider N_R distinct ROMs constructed with the same number of POD basis

$\boldsymbol{\Phi}_j \in \mathbb{R}^{n \times n_w}$. The number of distinct models N_R is in fact the number of parameters (or operating points) λ_j considered in the interpolation procedure. Each of these subspaces $\mathcal{W}$ belongs to the Grassmann manifold $\mathcal{G}(n, n_w)$, and can be considered as a point on the manifold, represented by its basis $\boldsymbol{\Phi}_j$ (figure 4.3). The subspaces are then projected on a flat constraint-free space $\mathcal{T}_{\mathcal{W}_0}$, tangent to the manifold at a reference point $\mathcal{W}_0$, where standard interpolation techniques can be easily applied. This interpolation method for ROM adaptation is proven to be robust with respect to the choice of the reference point as long as all the other operating points used in the process lay in the neighborhood of the reference point [4]. Each subspace $\mathcal{W}_j$ is mapped to a matrix $\boldsymbol{\Gamma}_j$ using a logarithm map and is represented by a point χ_j

$$\left(\boldsymbol{I} - \boldsymbol{\Phi}_0 \boldsymbol{\Phi}_0^T\right) \boldsymbol{\Phi}_j \left(\boldsymbol{\Phi}_0^T \boldsymbol{\Phi}_j\right)^{-1} = \boldsymbol{U}_j \boldsymbol{\Sigma}_j \boldsymbol{V}_j^T \tag{4.13.1}$$

$$\boldsymbol{\Gamma}_j = \boldsymbol{U}_j \tan^{-1}\left(\boldsymbol{\Sigma}_j\right) \boldsymbol{V}_j^T. \tag{4.13.2}$$

Matrix $\boldsymbol{\Gamma}^*$ corresponding to the target operating point λ^*, is then computed by interpolation of the matrices $\boldsymbol{\Gamma}_j$, using Lagrangian polynomial interpolation

$$\boldsymbol{\Gamma}^* = \sum_{j=1}^{N_R-1} \left(\boldsymbol{\Gamma}_j \prod_{\substack{k=1 \\ k \neq j}}^{N_R-1} \frac{\lambda^* - \lambda_k}{\lambda^j - \lambda_k} \right). \tag{4.14}$$

Finally, the interpolated matrix $\boldsymbol{\Gamma}^*$ representing $\tilde{\chi} \subset \mathcal{T}_{\mathcal{W}_0}$ is exponentially mapped back on the Grassmann manifold into the target subspace $\mathcal{W}^*$ spanned by $\boldsymbol{\Phi}^*$, corresponding to the set of POD basis needed for the construction of a projected reduced-order model for a new operating point λ^*

$$\boldsymbol{\Gamma}^* = \boldsymbol{U}^* \boldsymbol{\Sigma}^* {\boldsymbol{V}^*}^T \tag{4.15.1}$$

$$\boldsymbol{\Phi}^* = \boldsymbol{\Phi}_0 \boldsymbol{V}^* \cos \boldsymbol{\Sigma}^* + \boldsymbol{U}^* \sin \boldsymbol{\Sigma}^*. \tag{4.15.2}$$

The procedure used to perform the interpolation is given in algorithm 3, while a schematic of the mapping between the manifold and its tangent space at a point is shown in figure 4.3.

Algorithm 3 Interpolation on a tangent space of the Grassmann manifold

Input Set of parameters $\boldsymbol{\Lambda} : \{\lambda_0, \lambda_1, \ldots, \lambda_{N_R-1}\}$, corresponding POD basis $\boldsymbol{\Phi} : \{\boldsymbol{\Phi}_0, \boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_{N_R-1}\}$

Output POD for new parameter $\boldsymbol{\Phi}^*$, $\lambda^* \notin \boldsymbol{\Lambda}$

1: Select a point $\mathcal{W}_0$ of the manifold as a reference and origin point for the interpolation
2: Logarithmically map $\boldsymbol{\Phi}_j \in \boldsymbol{\Phi} - \boldsymbol{\Phi}_0$ onto the tangent space using a thin SVD (4.13)
3: Perform Lagrangian interpolation of matrices $\boldsymbol{\Gamma}_j$ (4.14)
4: Compute $\boldsymbol{\Phi}^*$ by exponentially mapping $\boldsymbol{\Gamma}^*$ to the Grassmann manifold (4.15)

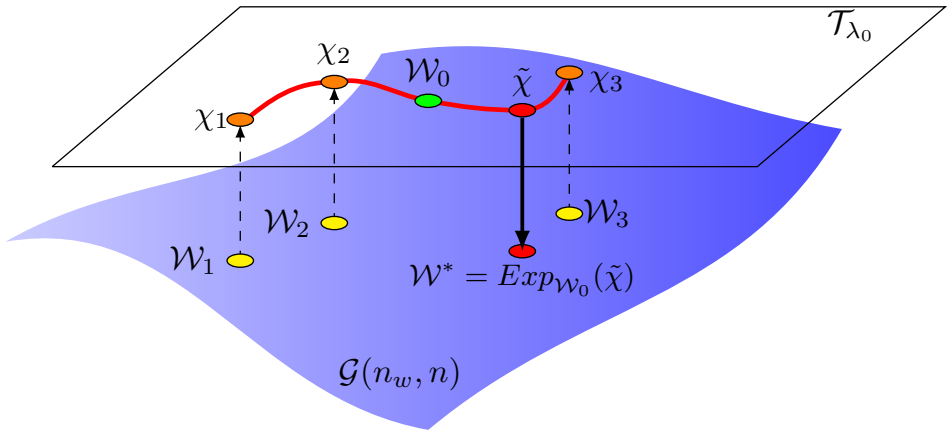


Figure 4.3: Interpolation of four subspaces using the Grassmann manifold $\mathcal{G}(n_w, n)$ and its tangent space $\mathcal{T}_{\lambda_0}$ at the point $\mathcal{W}_0$. The subspaces are mapped onto the tangent space to perform the interpolation, then the resulting point $\widetilde{\chi}$ is projected back onto the manifold using an exponential mapping.

We apply this interpolation procedure on a two-dimensional flow over a circular cylinder, analysed previously, where the Reynolds number is chosen as the variable parameter. The flow is simulated for $Re = \{100, 120, 140, 160, 180, 200\}$, the snapshots are collected over two shedding periods and the modes are built independently for each set of data using the snapshot-POD introduced in section §4.3. Figure 4.4 shows the lift coefficients and the first modes for each value of the Reynolds number. We aim to recover the POD basis needed to construct a reduced-order model at $Re^* = 150$, by interpolating between the existing data-points. The reference and origin point for the tangent constraint-free space is selected to be $Re_0 = 140$ and each set of basis $\mathbf{\Phi}_j \in \mathbb{R}^{n \times n_w}$ is truncated with $n_w = 20$. The resulting POD modes $\mathbf{\Phi}^* \in \mathbb{R}^{n \times n_w}$ spanning the target subspace $\mathcal{W}^*$ at $Re^* = 150$ are computed using algorithm 3, and the results are shown in figure 4.5. The obtained basis can be then used in any projection-based model reduction technique, allowing the creation of a reduced-order model without having to collect the snapshots from the FOM.

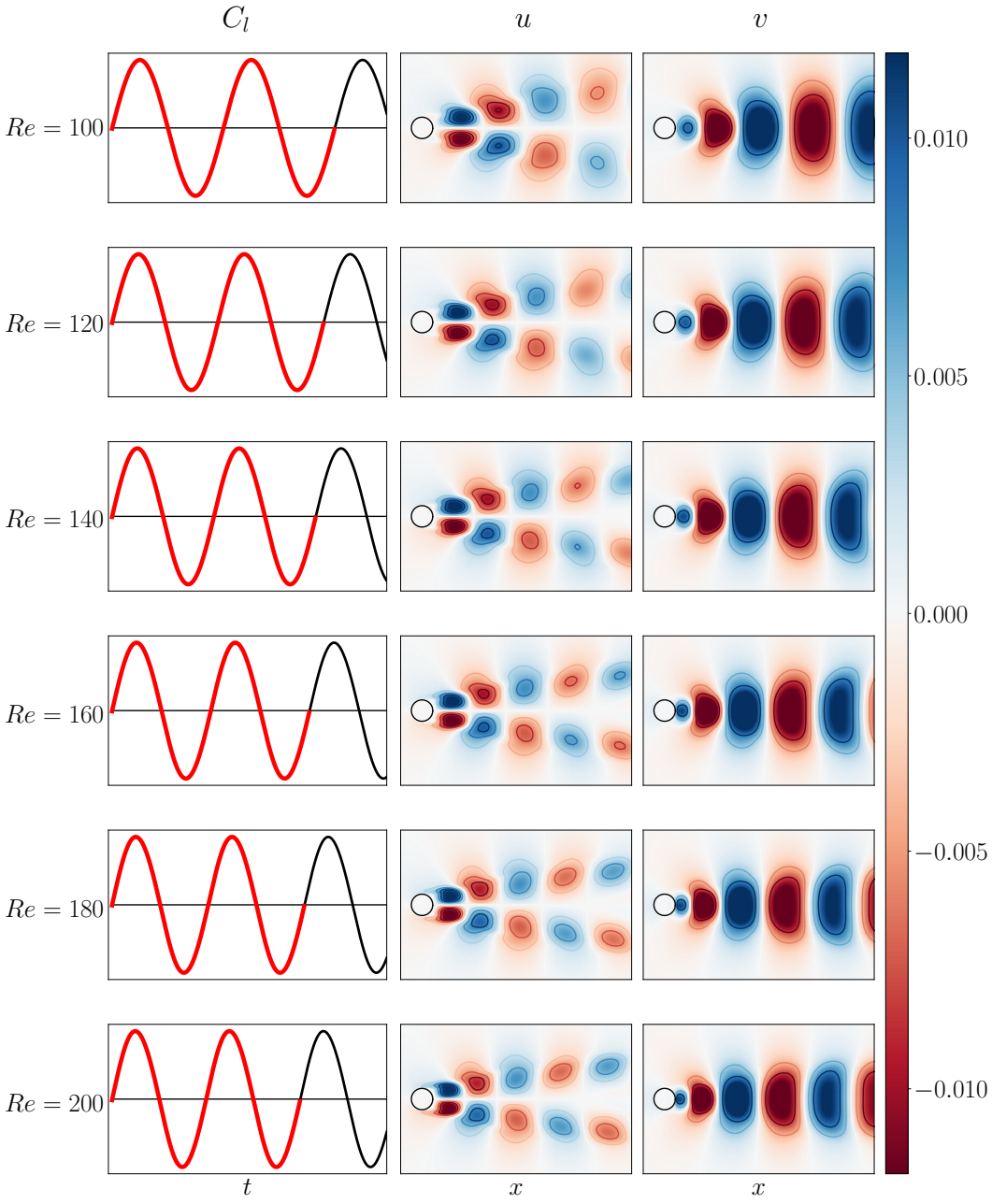


Figure 4.4: Training set of modes used for the interpolation for different Reynolds numbers. Left column: lift coefficient, the time interval used for the extraction of the modes is highlighted in red; center: horizontal velocity first POD mode; right: vertical velocity first POD mode.

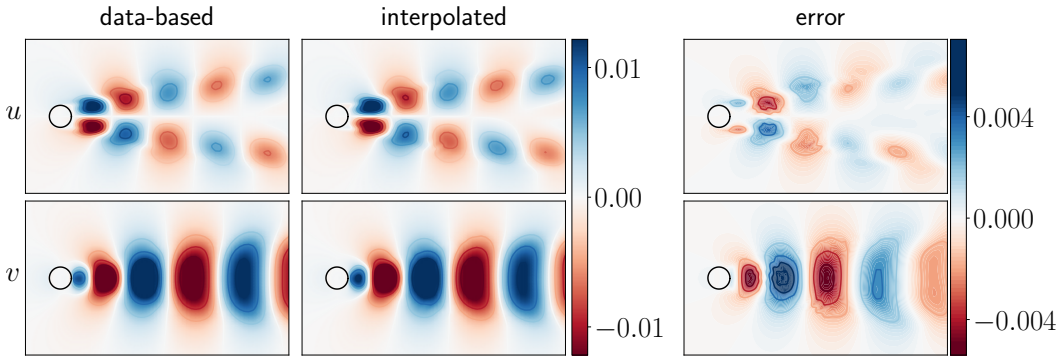


Figure 4.5: First POD modes for horizontal and vertical velocity field at $Re^* = 150$. (Left): data-based POD modes extracted from the SVD of the training FOM data; (center): POD modes obtained with the interpolation on the Grassmann manifold and its tangent space at $Re_0 = 140$, using $N_R = 6$; (right): error between the data-based and the interpolated modes.

4.5 Galerkin and Petrov-Galerkin projection

In section §4.2, we introduced projection-based model reduction methods. Using this technique the full-order model is projected on a low-dimensional trial space $\mathcal{W}$ by using a Galerkin expansion of the original high-dimensional data. Orthogonality between the high-fidelity model and a low-dimensional test space $\mathcal{L}$ is then enforced to extract the evolution equations governing the dimensionally reduced model, i.a. the expansion coefficients $\boldsymbol{a}(t)$ as

$$\boldsymbol{\Psi}^T\boldsymbol{\Phi}(\boldsymbol{x})\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{\Psi}^T\boldsymbol{f}(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{a}(t)). \tag{4.16}$$

$\boldsymbol{\Phi} \in \mathbb{R}^{n\times n_w}$ is the basis spanning $\mathcal{W}$, usually computed by using the method of snapshots and employing the orthogonal properties of the POD basis, and $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n_w}$ is the basis for the test subspace $\mathcal{L}$.

When $\boldsymbol{\Psi} = \boldsymbol{\Phi}$, the test basis is equal to the trial basis $\mathcal{L} = \mathcal{W}$, resulting in Galerkin projection. In particular, since the POD basis is orthogonal we have $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{I}$ and the reduced-order equation become

$$\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = \boldsymbol{\Phi}^T\boldsymbol{f}(\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{a}(t)). \tag{4.17}$$

The velocity fields can then be expressed by using a Galerkin expansion

$$\boldsymbol{u}(t,\boldsymbol{x}) = \sum_{j=0}^{k}\boldsymbol{\Phi}_j(\boldsymbol{x})a_j(t). \tag{4.18}$$

Here, $\boldsymbol{\Phi}$ are POD modes, and $a_0 = 1$ and $\boldsymbol{\Phi}_0$ denoting the mean field. Plugging (4.18) into the incompressible Navier-Stokes equations to perform a Galerkin projection gives

$$\frac{\partial}{\partial t}\sum_{j=0}^{k}\boldsymbol{\Phi}_j a_j + \sum_{j=0}^{k}\boldsymbol{\Phi}_j a_j\cdot\nabla\sum_{j=0}^{k}\boldsymbol{\Phi}_j a_j = -\nabla p + \frac{1}{Re}\nabla^2\sum_{j=0}^{k}\boldsymbol{\Phi}_j a_j, \tag{4.19.1}$$

$$\nabla\cdot\sum_{j=0}^{k}\boldsymbol{\Phi}_j a_j = 0. \tag{4.19.2}$$

The continuity equation (4.19.2) is automatically satisfied by each POD mode as well as boundary conditions. To recover the reduced-order model, we enforce the orthogonality to obtain the evolution equation for the expansion coefficients

$$\frac{\mathrm{d}a_i}{\mathrm{d}t} = \sum_{j=0}^{k} -Re^{-1}\langle\boldsymbol{\Phi}_i,\nabla^2\boldsymbol{\Phi}_j\rangle + \sum_{j=0}^{k}\sum_{s=0}^{k}\langle-\boldsymbol{\Phi}_i,\boldsymbol{\Phi}_j\cdot\nabla\boldsymbol{\Phi}_s\rangle. \tag{4.20}$$

Pressure can be neglected in the Galerkin projection of an incompressible flow, since it is seen as a constraint on the modal amplitude, and it is then considered as a Lagrange multiplier and can be computed from the velocity field yielding the pressure-Poisson equation. More details on the pressure-term representation in Galerkin

projection methods can be found in the work of Noack [154, 155] and Holmes [93]. Bergmann *et al.* [17] has also proposed a method to include the pressure basis to improve the stability of POD models.

Galerkin projection is commonly used in literature especially on problems governed by linear dynamics. However, Galerkin-based ROMs are proven to lack robustness when applied to nonlinear time dependent problems. The limitation of this approach can be attributed to the *ad hoc* truncation of the Galerkin basis. The truncation of the expansion ignores the cumulative effect of truncated scales on the retained degrees of freedom, making the model-reduced system prone to detrimental long-term instabilities. Different methods have been proposed to overcome these problems and improve the stability of the Galerkin projection method introducing numerical dissipation via closure models [96, 97], including a numerical model for the pressure [92, 155], using eddy viscosity [9, 18, 163, 190]. To further improve the computational efficiency and the dimension of the reduced models, empirical techniques were proposed for capturing the non-linearities and evaluate the non-linear terms at a subset of points [14, 21, 45, 128, 150, 149, 159, 129].

In the following we focus on an alternative *a priori* stabilization technique based on Petrov-Galerkin projection and tensors approximation, proposed by Carlberg *et al.* [40, 41]. In Petrov-Galerkin projection, the test space $\mathcal{L}$ is chosen to be different than the trail space $\mathcal{W}$, hence $\boldsymbol{\Phi} \neq \boldsymbol{\Psi}$. Reformulating equation (4.16) in terms of residuals

$$\boldsymbol{\Psi}^T R(\boldsymbol{a}(t)) = 0, \tag{4.21}$$

the test basis is $\boldsymbol{\Psi} = \boldsymbol{J}\boldsymbol{\Phi}$ and $\boldsymbol{J} = \partial R(\boldsymbol{a}(t))/\partial(\boldsymbol{\Phi}\boldsymbol{a}(t))$ is the non-linear Jacobian. One of the advantages of Petrov-Galerkin projection compared to Galerkin is that it performs a projection at the fully discrete level, computing a solution that minimizes the ℓ_2-norm of the time discrete residuals. For a detailed comparison between the two projection methods, the reader is referred to [39].
The dimension of the reduced-order-models obtained both with Galerkin and Petrov-Galerkin projection still scales with the large n-dimensional full-order-model, making the storage of the matrices and the computation of the matrix products costly. In their proposed method GNAT, Carlberg et al. [40, 41] apply a tensors approximation based on gappy data reconstruction [61] to select a subset of points where the nonlinearities of the flow are concentrated. Using this method, the final ROM dimensions depend only on the number of points selected by the algorithm.

The GNAT method showed good results in terms of dimensionality reduction and accuracy of the approximation for nonlinear problems where Galerkin projection showed instability. In the next sections we provide an overview of the GNAT method along with the results obtained for a two-dimensional incompressible flow.

4.6 Gauss-Newton with approximated tensors

The Gauss-Newton with approximated tensor GNAT method, introduced by Carlberg [40, 41], is a nonlinear model reduction method for discrete nonlinear static and dynamical problems. The method is based on a Least-Squares Petrov-Galerkin (LSPG) projection, where the test basis is chosen to minimize the ℓ_2-norm of the residual at each Newton iteration. The minimization is then associated with a gappy POD procedure [61], which is least-square optimal, to approximate the nonlinear residual and Jacobian at each iteration, introducing a sample mesh of relevant points to be used in the online stage and reducing the dimension associated to the least-square problem.

4.6.1 Least-squares Petrov-Galerkin projection

The first step of the GNAT method consists of a dimensionality reduction obtained with the use of a Petrov-Galerkin projection. We consider the generic nonlinear time dependent semi-discrete full-order model, introduced in (4.1) in residual form

$$R(\boldsymbol{q}(\boldsymbol{x}, t)) = 0, \tag{4.22}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the state, $R : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear mapping, and the initial condition is $\boldsymbol{q}(0) = \boldsymbol{q}_0$. Following the procedure given in the previous section for the Petrov-Galerkin projection, a first projection on the affine trial space $\mathcal{W} \subset \mathbb{R}^n$, spanned by POD basis $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n_w}$, is performed to reduce the dimensions of (4.22). The basis is computed with the snapshots-POD technique.

The snapshots collection procedure used to compute the POD basis $\boldsymbol{\Phi}$ is chosen to be $\mathbf{Y} = \{\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(0)} | n = 1, \ldots, n_t\}$, where $\boldsymbol{q}^{(0)}$ is the initial condition.
This projection leads to an approximated solution $\widetilde{\boldsymbol{q}}$ on the trial subspace $\boldsymbol{q}^{(0)} + \mathcal{W}$

$$\widetilde{\boldsymbol{q}} = \boldsymbol{q}^{(0)} + \boldsymbol{\Phi}\boldsymbol{q}_w, \tag{4.23}$$

where $\boldsymbol{q}^{(0)} \in \mathbb{R}^n$ is an initial guess at Newton iteration $k = 0$ (usually taken to be the optimal solution of the previous time-step) and $\boldsymbol{q}_w \in \mathbb{R}^{n_w}$ are the generalized coordinates of the state, optimized at each Newton iteration by residual minimization. Substituting (4.23) into (4.22) and enforcing the orthogonality of the test subspace $\mathcal{L}$ nonlinear residual, we get

$$\boldsymbol{\Psi}^T R(\boldsymbol{q}^{(0)} + \boldsymbol{\Phi}\boldsymbol{q}_w) = 0. \tag{4.24}$$

The residual minimization problem used to compute $\widetilde{\boldsymbol{q}}$ reads

$$\min_{\widetilde{\boldsymbol{q}} \in \boldsymbol{q}^{(0)} + \mathcal{W}} \|R(\widetilde{\boldsymbol{q}})\|_2. \tag{4.25}$$

GNAT solves the nonlinear least-square problem using a Gauss-Newton iterative method, seeking a solution that is discrete optimal at each time step. The test

subspace $\mathcal{L}$ is chosen to minimize the error between the reduced-order and the full-order solutions, i.e. to minimize the error in the search direction

$$\boldsymbol{s}^{(k)} = \arg\min_{a \in \mathbb{R}^{n_w}} \|\boldsymbol{\Phi} a - \boldsymbol{J}^{(k)^{-1}} R^{(k)}\|_{\Theta}. \tag{4.26}$$

For a nonlinear problem, the Jacobian arising at each iteration is usually not symmetric positive definite (SPD). In [40] the authors give a mathematical proof that justifies the use of a Petrov-Galerkin projection, with $\mathcal{L} = \boldsymbol{J}^{(k)}\mathcal{W}$, which then changes from one Newton iteration to another. The least-square problem (4.26) therefore becomes

$$\boldsymbol{s}^{(k)} = \arg\min_{a \in \mathbb{R}^{n_w}} \|\boldsymbol{J}^{(k)}\boldsymbol{\Phi} a + R^{(k)}\|_2, \tag{4.27.1}$$

$$\boldsymbol{q}_w^{(k+1)} = \boldsymbol{q}_w^{(k)} + \alpha^{(k)}\boldsymbol{s}^{(k)}, \tag{4.27.2}$$

where the computed search directions are optimal for $\Theta = \boldsymbol{J}^{(k)^T}\boldsymbol{J}^{(k)}$, and α is the step length computed with the use of a line search algorithm. The Petrov-Galerkin projection defined in this way, produces the following reduced Newton iterations for each time step, $k = 1, \ldots, K$ of the Newton iteration

$$\boldsymbol{\Phi}^T\boldsymbol{J}^{(k)^T}\boldsymbol{J}^{(k)}\boldsymbol{\Phi}\boldsymbol{s}^{(k)} = -\boldsymbol{\Phi}^T\boldsymbol{J}^{(k)^T}\boldsymbol{R}^{(k)}, \tag{4.28.1}$$

$$\boldsymbol{q}_w^{(k+1)} = \boldsymbol{q}_w^{(k)} + \alpha^{(k)}\boldsymbol{s}^{(k)}, \tag{4.28.2}$$

where K is defined by a stopping convergence criterion based on the residual error. The above equation then substitutes the general equation for the least-square problem (4.27). As mentioned in the previous section, the computational cost associated with the resolution of the nonlinear least square problem (4.27) scales with the large dimensions n of the full-order model, because of the dimensions of the two-dimensional tensors $\boldsymbol{J}^{(k)}\boldsymbol{\Phi}$ and $\boldsymbol{R}^{(k)}$ involved in the resolution of the normal equation. To additionally reduce the dimensions of the Gauss-Newton iterations, GNAT introduces a hyper-reduction step based on data reconstruction techniques.

4.6.2 Tensors approximation and hyper-reduction

The large dimensions of the matrices in equation (4.27) can lead to computational bottleneck, that can be handled by constructing approximations for these large operators. GNAT employs the gappy POD data reconstruction technique [61], originally used for image reconstruction, to approximate these operators. This technique acts like a masked projection, by selecting a small subset of the full dimensional data in order to obtain a smaller system to solve. The subset is defined as $\mathcal{I} \equiv \{I_1, I_2, \ldots, I_{n_i}\}$ with $n_i \ll n$, and it includes the indices for the selected sample points where the nonlinear operators are evaluated. In this way a sample matrix defined as $\mathbf{Z} \equiv \mathbf{Z}(\mathcal{I}) \in \mathbb{R}^{n \times n_i}$ can be used to approximate $\boldsymbol{J}^{(k)}\boldsymbol{\Phi}$ and $\boldsymbol{R}^{(k)}$ at each

Gauss-Newton iteration

$$\widetilde{\boldsymbol{R}}^{(k)} = \arg \min_{\boldsymbol{x} \in \text{range}(\boldsymbol{\Phi}_R)} \|\mathbf{Z}\boldsymbol{R}^{(k)} - \mathbf{Z}\boldsymbol{x}\|_2, \tag{4.29.1}$$

$$\widetilde{\boldsymbol{J}^{(k)}\boldsymbol{\Phi}} = \arg \min_{\boldsymbol{x} \in \text{range}(\boldsymbol{\Phi}_J)} \|\mathbf{Z}\boldsymbol{J}^{(k)}\boldsymbol{\Phi} - \mathbf{Z}\boldsymbol{x}\|_2. \tag{4.29.2}$$

The approximated tensors lie in the low-dimensional subspaces spanned by $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$, computed by using snapshots POD on the nonlinear residuals and column-reduced Jacobian snapshots collected during the resolution of problem (4.27). Different strategies to collect these snapshots in order to meet the consistency requirement for the approximation are provided in [40]. Finally substituting the reduced operators in equation (4.27) we get the dimensionally reduced GNAT iterations

$$\boldsymbol{s}^{(k)} = \arg \min_{a \in \mathbb{R}^{n_w}} \|\mathbf{A}\mathbf{Z}\boldsymbol{J}^{(k)}\boldsymbol{\Phi}\boldsymbol{a} + \mathbf{B}\mathbf{Z}\boldsymbol{R}^{(k)}\|_2, \tag{4.30.1}$$

$$\boldsymbol{q}_w^{(k+1)} = \boldsymbol{q}_w^{(k)} + \alpha^{(k)}\boldsymbol{s}^{(k)}, \tag{4.30.2}$$

where $\mathbf{A} = [\mathbf{Z}\boldsymbol{\Phi}_J]^+$ and $\mathbf{B} = \boldsymbol{\Phi}_J^T\boldsymbol{\Phi}_R\,[\mathbf{Z}\boldsymbol{\Phi}_R]^+$ can be computed "a priori" during the offline stage. In particular, $\mathbf{A} \in \mathbb{R}^{n_J \times n_i}$, $\mathbf{B} \in \mathbb{R}^{n_J \times n_i}$, and n_J is the number of POD basis used to approximate the column-reduced Jacobian. Also n_i is the size of the subset $\mathcal{I}$ and the superscript "+" denotes the Moore-Penrose pseudo-inverse.
The Jacobian matrix arising from CFD simulations is usually sparse, therefore the dimension of the problem can be further reduced by considering only its nonzero elements. Another set $\mathcal{J}$, containing the n_j indices of the nonzero elements of the Jacobian, is then introduced. The reduced state is then computed as $\overline{\mathbf{Z}}^T\overline{\mathbf{Z}}\boldsymbol{q} \in \mathbb{R}^{n_j}$ with $n_j \ll n$, and $\overline{\mathbf{Z}} \equiv \mathbf{Z}(\mathcal{J})$ denoting a masking operator. The final reduced-order least-square problem can be then reformulated as

$$\boldsymbol{s}^{(k)} = \arg \min_{\boldsymbol{a} \in \mathbb{R}^{n_w}} \|\mathbf{A}\mathbf{C}^{(k)}\boldsymbol{a} + \mathbf{B}\mathbf{D}^{(k)}\|_2, \tag{4.31.1}$$

$$\boldsymbol{q}_w^{(k+1)} = \boldsymbol{q}_w^{(k)} + \alpha^{(k)}\boldsymbol{s}^{(k)}, \tag{4.31.2}$$

where $\mathbf{C}^{(k)} = \mathbf{Z}\boldsymbol{J}(\overline{\mathbf{Z}}^T\overline{\mathbf{Z}}\widetilde{\boldsymbol{q}}^{(k)})\overline{\mathbf{Z}}^T\overline{\mathbf{Z}}\boldsymbol{\Phi}$ and $\mathbf{D}^{(k)} = \mathbf{Z}\boldsymbol{R}(\overline{\mathbf{Z}}^T\overline{\mathbf{Z}}\widetilde{\boldsymbol{q}}^{(k)})$. The dimension of the least-squares problem in (4.31) depends only on the size of the chosen subset of points $\mathcal{I}$. Therefore, the cost of the online stage is considerably smaller then the cost of the full-order model (4.22). A summary of the algorithm used in GNAT is provided in figure 4.6. For the details on the determination of the index sets $\mathcal{I}$ and $\mathcal{J}$ the reader is referred to [41].

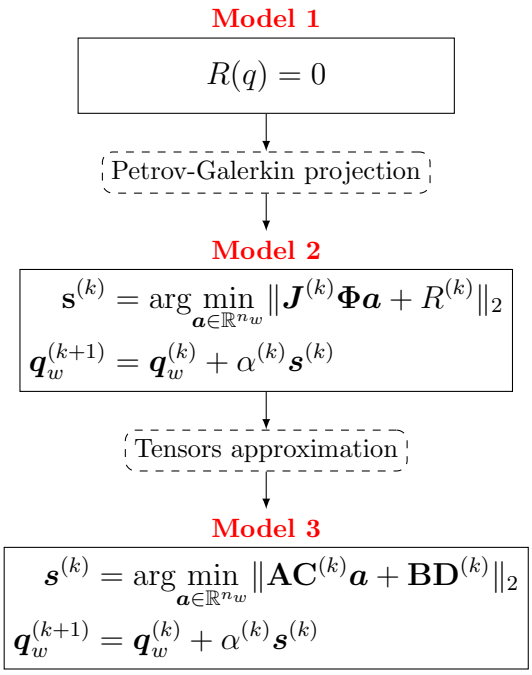


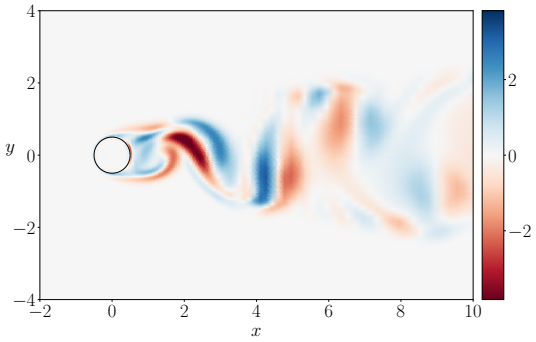
Figure 4.6: A schematic of the GNAT method. Model 1 corresponds to the FOM simulation. Model 2 is the first dimensional reduction performed by a Petrov-Galerkin Least-Square problem. Model 3 is the online phase involving a tensors approximation on a subset of points selected with the use of a greedy algorithm.

4.7 Results

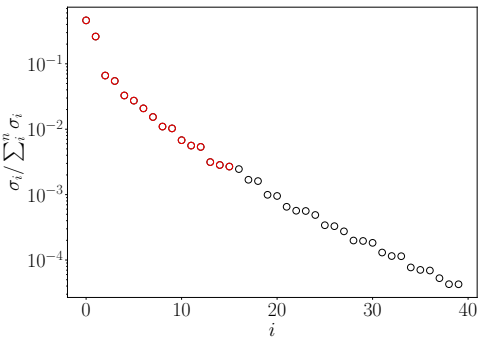
In this section, the GNAT method is first tested on a two dimensional flow around a cylinder at $Re = 200$, then the interpolation via the Grassmann manifold and its tangent space at a point is applied to recover the basis for a new operating point. The resulting basis is then used to build a reduced-order model following the GNAT procedure.

4.7.1 Reduced-order model - GNAT

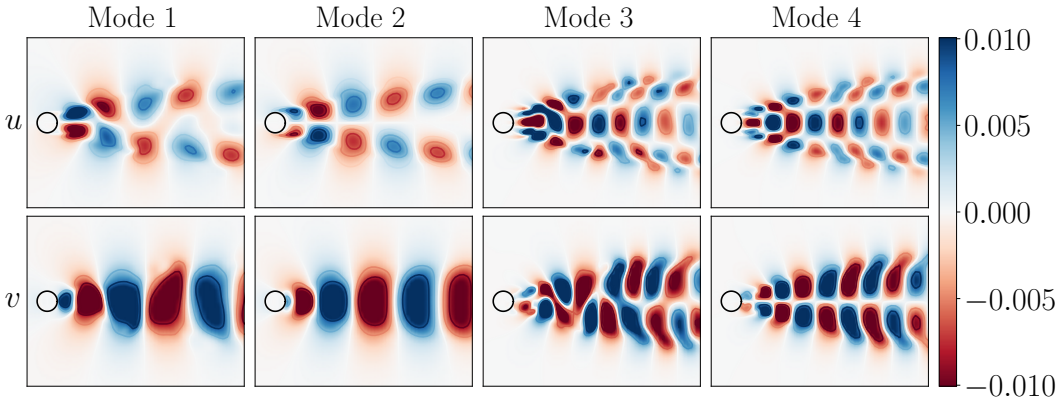
We consider the two dimensional flow around a cylinder at $Re = 200$. The objective is to construct a reduced order model capable of reproducing the resulting lift and drag coefficient. The system is discretized over a cartesian non-uniform staggered grid of size $N_x = 382 \times 382 = 146689$. The immersed boundary method, described in section §2.4 is used to capture the cylinder shape. For this purpose $n_{IB} = 96$ Lagrangian points are employed. The dimension of the state vector, including the immersed boundary points, is $n = 437581$, where $n = n_u + n_v + n_p + 2\,n_{IB}$ represents respectively the dimension of the discretized horizontal and vertical velocity, along with pressure on the cartesian grid, as well as, horizontal and vertical velocities on the surface points.



(a) Example of a vorticity snapshot $\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(0)}$.



(b) Singular values in descending order.

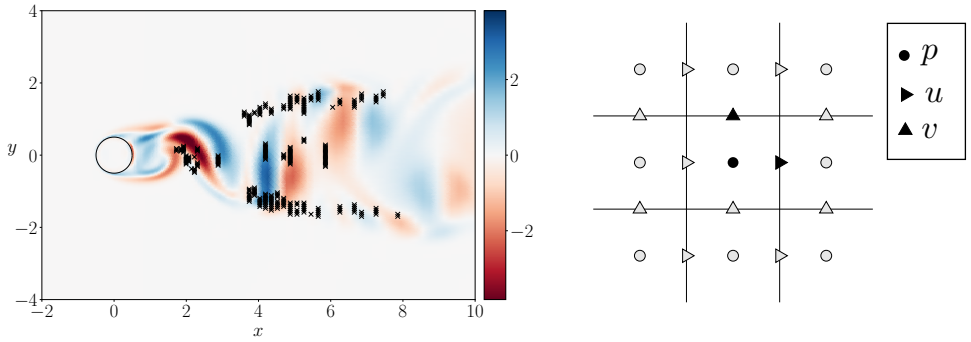


(c) Spatial POD modes for horizontal and vertical velocity.

Figure 4.7: Results of the SVD on a sample of incremental snapshots for the GNAT method. (a) an example of a vorticity snapshot; (b) singular values, highlighted in red are modes selected for the projection, preserving the 99% of the energy; (c) incremental POD modes.

The snapshots are collected during the training FOM simulation (model 1) and the POD modes are computed using the method of snapshots introduced in section §4.3. In particular, we collect $n_t = 340$ snapshots over two shedding periods. Note that GNAT is an incremental method, therefore, the snapshots are collected as $\mathbf{Y} = \{\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(0)} | n = 1, \dots, n_t\}$, where $\boldsymbol{q}^{(0)}$ is the initial condition, and the POD modes represent the solution increment at each time step of the FOM simulation. This way of collecting data affects the shape of the modes and the corresponding singular values. Figure 4.7a shows an example of an incremental snapshot. Here, in contrast to the case where the mean is removed, the singular values are not in pair resulting in non symmetrical mode shapes, shown in 4.7c. The modes are truncated to preserve the 99% of the total energy (4.12), leading to a matrix $\boldsymbol{\Phi}_w \in \mathbb{R}^{n \times n_w}$, with $n_w = 16$. The chosen modes are highlighted in red in figure 4.7b.

When the POD modes are collected, a Petrov-Galerkin projection (4.28) is employed leading to the first dimensional reduction (model 2). Model 2 has size



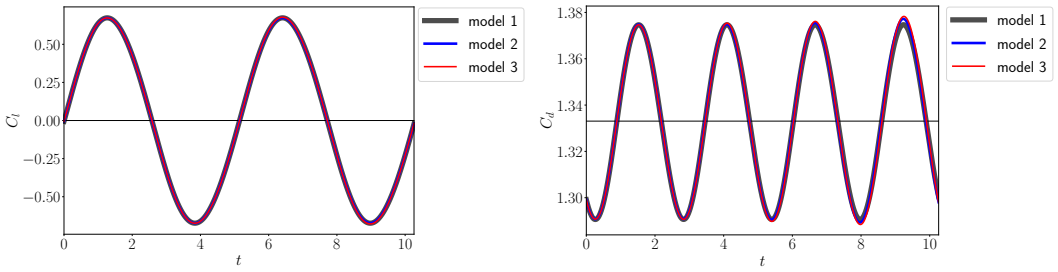
(a) (b)

Figure 4.8: Greedy points selection. (a) Subset of points $\mathcal{I}$ used for the hyper-reduction; (b) selection of the triad points, (u, v, p), on a cartesian staggered grid.

$n \times n_w$, which still scales with the dimension of the high-dimensional problem. During this stage the residuals $\boldsymbol{R}^{(k)}$ and the column-reduced Jacobian $\boldsymbol{J}^{(k)}\boldsymbol{\Phi}_w$ arising at each Gauss-Newton iteration "k" are stored and then approximated in the corresponding low-dimensional subspace spanned by $\boldsymbol{\Phi}_R$ and $\boldsymbol{\Phi}_J$. A small subset of the full dimensional domain is then selected by using the Gappy-POD technique to further reduce the dimension of the system, leading to model 3.

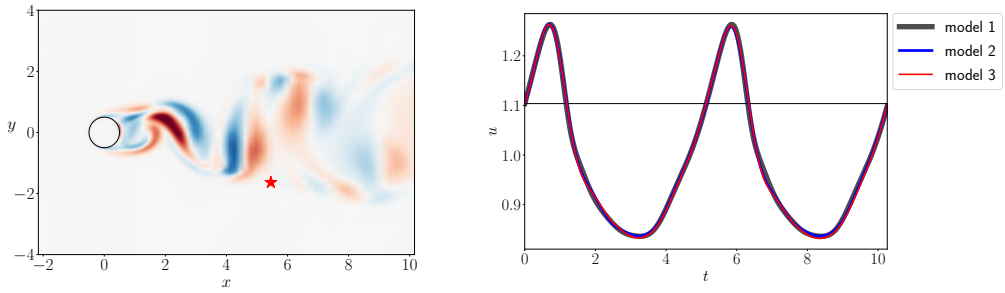
The greedy algorithm is employed to select n_G points inside the domain. The numerical solver employed for the simulations is spatially discretized on a staggered grid, where pressure is defined at the center of the cell and the velocity components on its boundaries, as indicated in figure 4.8b. The greedy algorithm selects the triad points at the center of the cell and the respective edges as highlighted in figure 4.8b. The indices representing the corresponding triad (u, v, p) are then added to the subset $\mathcal{I}$. In order to avoid numerical instabilities, the algorithm is enforced to select points in a window $2 \leq x \leq 20$ excluding points that are too close to the cylinder or to the outflow boundary. Figure 4.8a shows the subset of final selected points. It can be noticed that these points lie where the truncated high order modes have the largest foot-print. The hyper-reduction step is then accomplished and the online problem (4.31) can be solved.

The performance of this reduction is assessed by the capability of the model to predict the lift and drag coefficients accurately (figure 4.9). The solver employs immersed boundary points to represent the cylinder surface. The lift and drag coefficients are then computed from the exact value of the velocity at the Lagrangian points. These points were excluded from the greedy point selection used for the construction of Model 3. In addition, the capability of the reduced-order model in reconstructing the full flow field is also assessed by comparing the reconstructed velocity profile to the exact value at a random point inside the domain (figure 4.10).



(a) Lift coefficient. (b) Drag coefficient.

Figure 4.9: Reproduction of the lift (a) and drag coefficients (b) for model 2 and model 3, using GNAT compared to the high-dimensional model (model 1).



(a) Random point inside the domain. (b) Horizontal velocity approximation.

Figure 4.10: GNAT results: Reproduction of the horizontal velocity for a random point in the domain.

The time-averaged relative error,

$$\frac{1}{n_t}\sum_{n=1}^{n_t}\frac{\|\boldsymbol{q}(\cdot,t^n)-\widetilde{\boldsymbol{q}}(\cdot,t^n)\|_2}{\|\boldsymbol{q}(\cdot,t^n)\|_2}, \tag{4.32}$$

using modes 2 and 3 is reported in table 4.1. Here $\boldsymbol{q}$ is the FOM flow field solution and $\widetilde{\boldsymbol{q}}$ is the reconstructed solution of the ROM. The error obtained with the full GNAT method is one order of magnitude higher than the one obtained with the simple Petrov-Galerkin projection. However, the dimension of the Gauss-Newton iteration has been reduced from $n \times n_w$ to $n_i \times n_i$ leading to a consistent reduction of the computational time (table 4.2).

	model 2	model 3
error	0.003	0.028

Table 4.1: Time-averaged ℓ_2-error (4.32) of the approximation for model 2 and model 3.

	model 1	model 2	model 3
wall-time	121 s	118 s	97 s

Table 4.2: Wall time for each level of approximation of acGNAT.

4.7.2 Variable Parameter GNAT

In this section, we build ROMs for a set of parameters and apply the interpolation between subspaces using the Grassmann manifold and its tangent space (section §4.4) to construct a ROM for a new operating point.

We consider again a two dimensional flow around a cylinder. The variable parameter is the Reynolds number. The flow is evaluated for six values of the Reynolds number, and the POD modes $\mathbf{\Phi}_j \in \mathbb{R}^{n \times n_w}$ are obtained using the incremental snapshots collection procedure. Twenty modes are retained for each parameter and then used to perform the interpolation procedure. The snapshots are collected for two shedding periods for all Reynolds numbers. Figure 4.11 shows the lift coefficient and the corresponding first mode of the horizontal and vertical velocity, obtained by using an incremental snapshot collection procedure.

The different values of the Reynolds number **Re** used for the interpolation are given in table 4.3. To examine the impact of the sampling set on the interpolated solution, we incrementally increase the number of operating points involved in the interpolation. The new operating point is $Re^* = 150$, not included in the set. The reference point needed to build the tangent space to the Grassmann manifold is chosen to be close enough to the new operating point and has been set to $Re_0 = 140$ for all the interpolations performed.

The shape of the first modes obtained with the interpolation on the Grassmann manifold and its tangent space at $Re_0 = 140$ using $N_R = 6$ are reported in figure 4.12. The first column represents the data-based modes, computed by performing SVD on the original snapshots. The shape of the interpolated modes is comparable to the data-based one. The third column shows low values for the difference between the two, however the contour lines are not smooth due to the difference in frequency of the flows used for the interpolation(figure 4.11).

To evaluate the impact of the interpolation on the reconstructed ROM at the new operating point, the modes obtained using the different set of parameters are plugged into model 2 of GNAT. The performance is checked on the reconstruction of the lift and drag coefficients at the surface of the cylinder. The results of the reduction are compared to the FOM and the data-based ROM (figure 4.13). From the figure, it can be noted that the addition of operating points in the parameter set used for the interpolation does not improve significantly the accuracy of the approximation.

The relative errors in time for the different approximations are evaluated using

	Re	Re_0
$N_R = 2$	$\{140, 160\}$	140
$N_R = 3$	$\{120, 140, 160\}$	140
$N_R = 4$	$\{120, 140,\ 160, 180\}$	140
$N_R = 5$	$\{100, 120, 140,\ 160, 180\}$	140
$N_R = 6$	$\{100, 120, 140,\ 160, 180, 200\}$	140

Table 4.3: Training set of operating points Re used in the interpolation procedure for different N_R.

the ℓ_2-norm formulation as

$$err(t) = \frac{\|\boldsymbol{q}(\boldsymbol{x}, t) - \widetilde{\boldsymbol{q}}(\boldsymbol{x}, t)\|_2}{\|\boldsymbol{q}(\boldsymbol{x}, t)\|_2}, \tag{4.33}$$

where $\widetilde{\boldsymbol{q}}$ is the reduced solution. The evolution in time of this error for all the parameters considered is shown in figure 4.14. The lowest error is found to be the one for $N_R = 3$, that corresponds to Reynolds numbers close to the new operating point. The quality of the approximation deteriorates when moving away from this value. As a matter of fact, we chose a uniform distribution for the parameters selection, receding both from Re_0 and Re^*. The approximation accuracy could be improved changing the distribution of the operating points selected for the interpolation and employing an adaptive addition of points during the process.

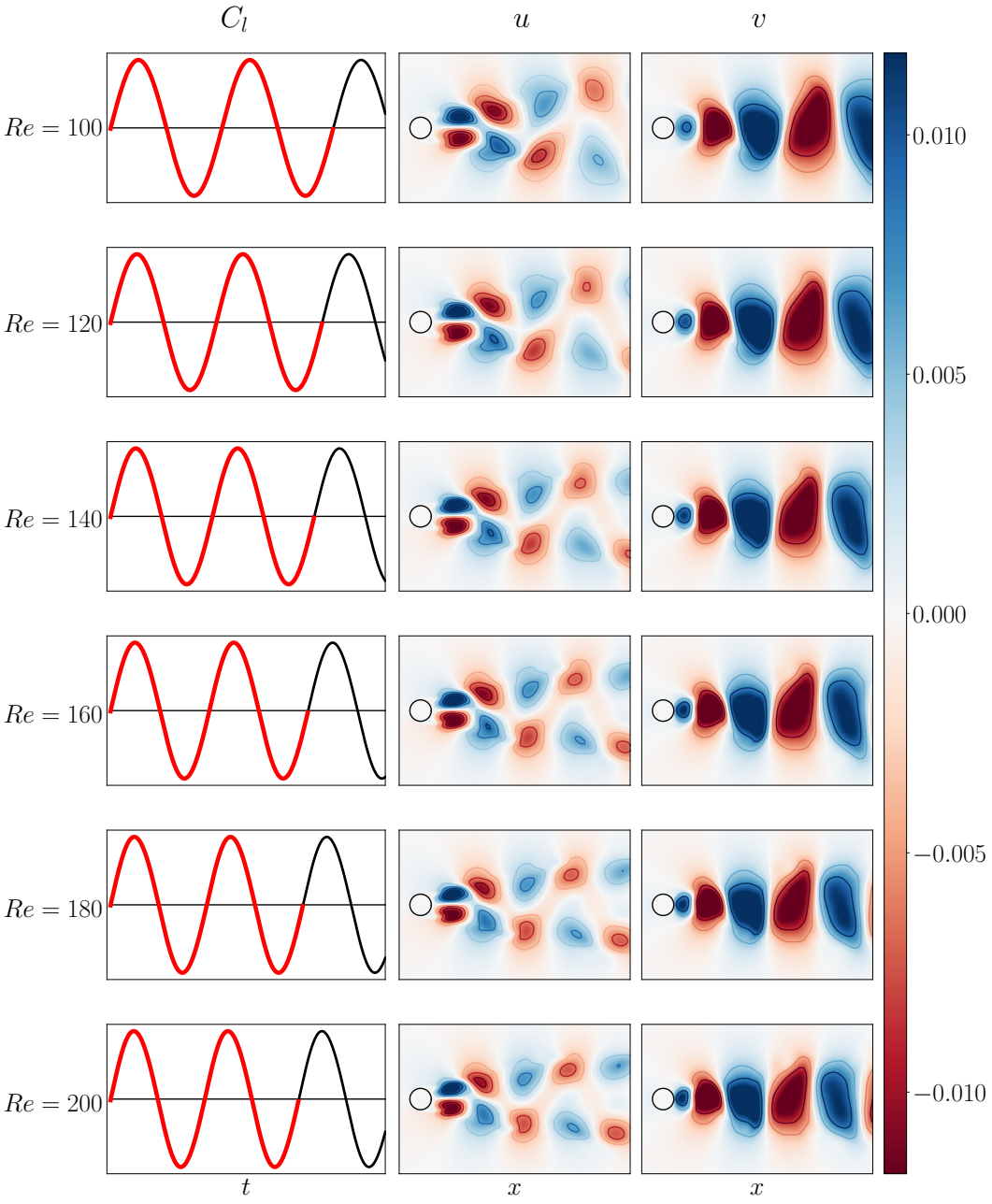


Figure 4.11: Set of modes used for the interpolation for different Reynolds numbers. The snapshots are collected using an incremental strategy. (Left): lift coefficient, the time interval used for the extraction of the modes is highlighted in red; (center): horizontal velocity first POD mode; (right): vertical velocity first POD mode.

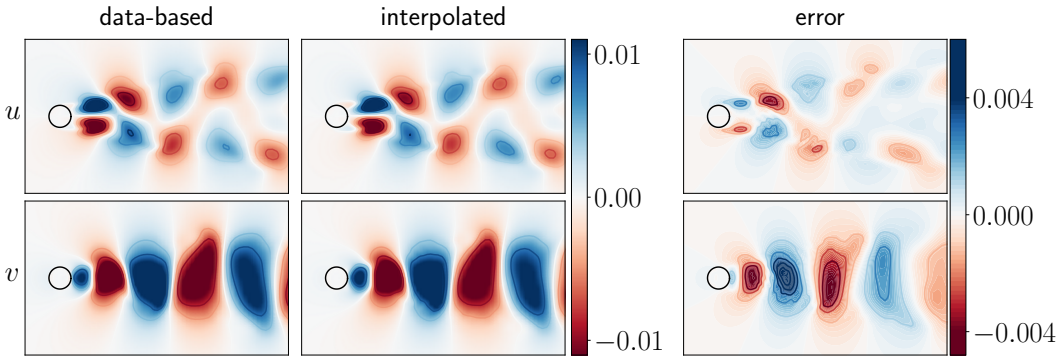


Figure 4.12: First incremental POD modes for horizontal and vertical velocity fields at $Re^* = 150$, with $N_R = 6$. (Left), data-based POD modes extracted by performing an SVD on the data; (center), POD modes obtained with the interpolation of the Grassmann manifold and its tangent space at $Re_0 = 140$; (right), error between the data-based and interpolated modes.

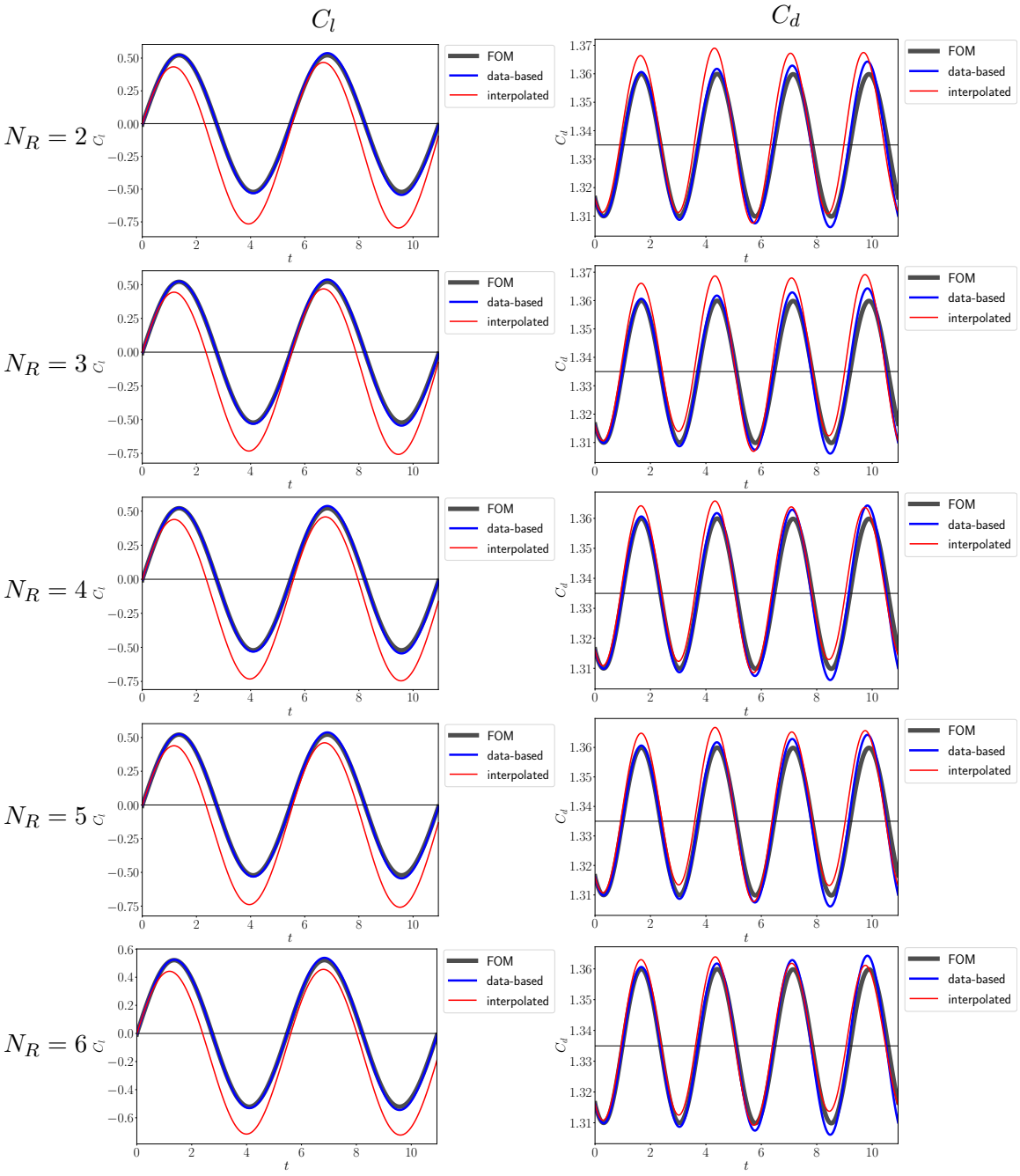


Figure 4.13: Interpolation of reduced-order models: reconstruction of lift coefficient C_l (left column) and drag coefficient C_d (right column) using different sets of operating points N_R given in table 4.3.

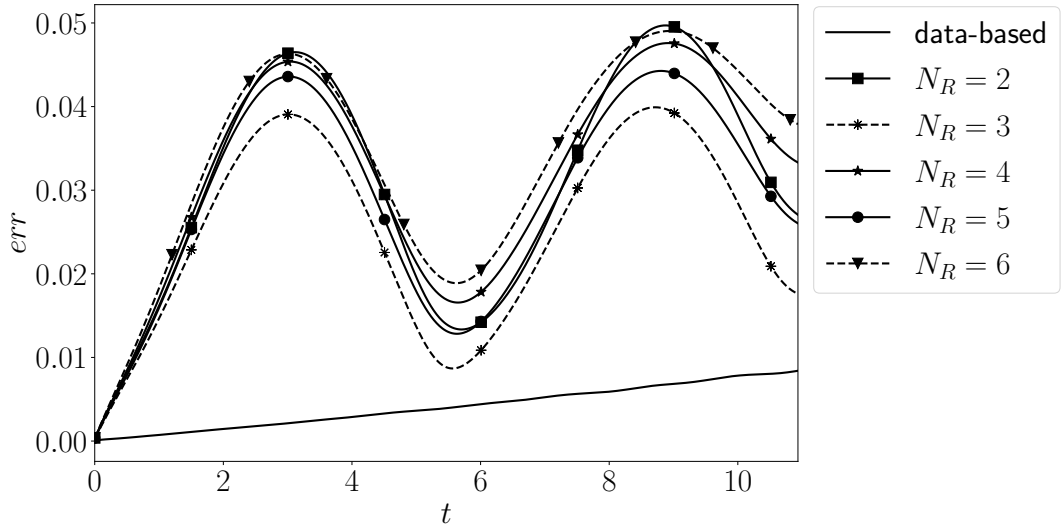


Figure 4.14: Relative error in time in ℓ_2 for the approximated flow using model 2 of the GNAT method for $Re^* = 150$ using the interpolated modes obtained from the training dataset in table 4.3 and the data-based modes.

4.8 Conclusions

In this chapter we gave an overview of projection-based techniques to reduce the dimension of unsteady problems. These methods rely on the possibility of reducing the dimension of a high-dimensional problem by combining its dominant dynamical features, or coherent structures, using a Galerkin expansion on a lower-dimensional subspace spanned by a set of orthogonal basis. Galerkin and Petrov-Galerkin projections have been studied and compared [39], finally the Gauss-Newton with approximated tensors method [41] has been chosen as a reference for building an equation-based reduced-order model. GNAT has been tested on a two-dimensional flow around a cylinder at $Re = 200$ to reconstruct significant flow features such as drag and lift coefficients.

The results obtained show an accurate reproduction of the flow with a dimensional reduction of the order of 93%. However, such models are not robust as model parameters change. The capability of the model to adapt to varying parameters has been tested by introducing an interpolation strategy between operating points. The interpolation used here is based on the Grassmann manifold and its tangent space at a point and it has been used to reconstruct a reduced model for a new operating point laying between the interpolating set. The results obtained are promising but further improvement could be obtained by optimizing the distribution and the number of the operating points used in the interpolation.
The employment of reduced-order models, eventually able to adapt to parameter variations, can significantly reduce the computational time for flow optimization and control applications. For example in derivative-free optimization algorithms, where the cost function has to be evaluated for different operating points in the design space, and in gradient descent algorithms where the values of the control parameters are changed at each iteration.

Chapter 5

Data-driven identification of inherent flow dynamics

Contents

5.1 Introduction 83
5.2 Equation identification - SINDy algorithm 84
5.3 SINDy applied to the Navier-Stokes equation 86
5.4 Clustering algorithms 88
5.4.1 Networks 89
5.4.2 Community detection - Leicht-Newman network 91
5.4.3 Stochastic model of turbulent axisymmetric wakes behind a bluff body 95
5.5 Combining regression and clustering procedures for dynamical identification 100
5.6 Conclusions 104

5.1 Introduction

With advances in computing power, larger and more accurate simulations are being performed, capable of capturing detailed interactions of various physical phenomena present in the flow. One of the main products of such expensive calculations is the resulting data, which is also increasing in size, opening the door to data-driven analysis, such as system identification and machine learning techniques.

This abundant data can be utilized to infer the equations governing the existing dynamics, which has been shown to scale to high-dimensional systems [30]. This approach had merit following the argument that dominant physical processes, present in a complex system, can vary and might not be well described by the same equation. For example, depending on the application and regime of interest, the Navier-Stokes equation can be simplified using nondimensional parameters and modified to neglect the effect of compressibility and modeled using turbulence models. Identifying and locating the different dynamics present in complex physical systems and eventually understanding how they affect the overall solution can increase the accuracy of the numerical model employed.

In many regimes, the dynamics of physical systems, usually described by complex partial differential equations, are governed by only a few nonlinear terms, allowing the production of simpler models capable of predicting the main features of the underlying system. The determination of these structures inherent in such physical systems can be accomplished through the use of data-driven models and system identification [35].

System identification is sometimes considered as a black-box approach (the fidelity of the identification process was described by Wiener [206]), where the dynamics are determined by identifying the response of the overall flow process. In this form, in contrast to the projection-based models discussed previously this approach ignores the subtleties of the underlying dynamics in the modelling process and uses no information regarding the existing structures. However, in an alternative approach (gray-box), information regarding the dynamics of the flow is included in the identifications process. Used this way, system identification is utilised to estimate the unknown parameters, which derive the underlying predetermined model. This method results in models that are particularly suited for control applications, since, by design, the dynamics taken into account are part of the input-output behaviour of the system. Linear system identification [204], has been applied in flow control [88, 94]. However, its application to nonlinear problems has proven to be very challenging leading to instabilities in the case of unsteady measurements [94].

Nonlinear algorithms are then employed to determine the existing nonlinear correlation between the underlying structures, and ultimately result in a more predictive model suitable for unsteady and nonlinear environments. Recently, Brunton, Proctor & Kutz [30] introduced the SINDy, which identifies parsimonious nonlinear models from data. Using this approach, nonlinear dynamics are represented as a linear combination of candidate nonlinear functions, as a result the method is readily extended to incorporate known physical constraints [122], leading to the intended gray-box approach, mentioned above, and has been applied recently in fluid mechanics [123], showing great promise.

In this chapter we propose a data-driven dynamics identification procedure realized by coupling sparse linear regression with network partitioning used for clustering purposes. The chapter is organized as follow, the SINDy algorithm is shortly described in section §5.2. The proposed procedure for dynamics identification is introduced in section §5.3. An introduction of network science is given in section §5.4, with a particular focus on network partitioning based on the Leicht-Newman network, summarized in section §5.4.2. The validation of the clustering algorithm and the results obtained with the proposed method are showed in section §5.4.3 and §5.5.

5.2 Equation identification - SINDy algorithm

The SINDy algorithm, proposed by Brunton *et al.* [30], leverages advances in machine learning and sparsity techniques to discover the governing equations from data measurements, without having any a priori information on the form of the expected

model. Considering the generic nonlinear dynamical system of the form of

$$\frac{\mathrm{d}\boldsymbol{y}(t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{y}(t)), \tag{5.1}$$

the system identification procedure aims to determine the form of the unknown generic nonlinear term $\boldsymbol{f}(\boldsymbol{y}(t))$ by using the information extracted from the time dependent data $\boldsymbol{y}(t) \in \mathbb{R}^n$. The time history of the data is collected and stored in a matrix $\mathbf{Y} \in \mathbb{R}^{n_t \times n}$, with each row containing the state vector $\boldsymbol{y}_i = \boldsymbol{y}(t_i)$ collected at different time instants $i = 1, \ldots, n_t$. A second matrix contains the time derivatives of $\mathbf{Y}$ collected a the same time instants. If not directly available, the derivatives can be computed numerically using a finite difference scheme, resulting in the set of two matrices below

$$\mathbf{Y} = \begin{bmatrix} \boldsymbol{y}^T(t_1) \\ \boldsymbol{y}^T(t_2) \\ \vdots \\ \boldsymbol{y}^T(t_{n_t}) \end{bmatrix}, \qquad \dot{\mathbf{Y}} = \begin{bmatrix} \dot{\boldsymbol{y}}^T(t_1) \\ \dot{\boldsymbol{y}}^T(t_2) \\ \vdots \\ \dot{\boldsymbol{y}}^T(t_{n_t}) \end{bmatrix}. \tag{5.2}$$

The right hand side of the equation $\boldsymbol{f}(\boldsymbol{y}(t))$ is then formulated as a linear combination of linear and nonlinear candidate functions of the columns of $\mathbf{Y}$. The candidate functions can assume any mathematical form, therefore, a library $\boldsymbol{\Theta}(\mathbf{Y})$ can be built from polynomial of any order or trigonometric terms. For partial differential equations, this library might also include partial derivatives and external forcing. An example of such a collection can be:

$$\boldsymbol{\Theta}(\mathbf{Y}) = \begin{bmatrix} | & | & | & & | & \\ 1 & \mathbf{Y} & \mathbf{Y}^{P_2} & \cdots & \sin(\mathbf{Y}) & \cdots \\ | & | & | & & | & \end{bmatrix}. \tag{5.3}$$

The nonlinear governing equation (5.1) can then be rewritten as

$$\dot{\mathbf{Y}} = \boldsymbol{\Theta}(\mathbf{Y})\boldsymbol{\Xi}, \tag{5.4}$$

where the unknown nonlinear term on the right hand side is now defined as a linear combination of the nonlinear candidate functions included in the library, and $\boldsymbol{\Xi} = [\xi_1\, \xi_2 \cdots \xi_n]$ represents the matrix containing the coefficients ξ_i of the linear combination.

Knowing that only few of these nonlinear terms affect the dynamics of the system and are therefore active in the operator representing $\boldsymbol{f}$, the nonzero coefficients in $\boldsymbol{\Xi}$ can be identified by using any sparse regression problem. One option is to use the Least Absolute Shrinkage and Selection Operator (LASSO) algorithm [198]

$$\boldsymbol{\xi} = \underset{\xi'}{\operatorname{argmin}} \|\boldsymbol{\Theta}\boldsymbol{\xi}' - \dot{\boldsymbol{y}}\|_2 + \lambda\|\boldsymbol{\xi}'\|_1, \tag{5.5}$$

where the solution is penalized in ℓ_1 to promote sparsity. There are several ways to identify the active terms of the library of candidate functions, enhancing the

sparsity of the solution. Among these methods are the sequentially thresholded least squares (STLSQ) [211], the sparse relaxed regularized regression (SR3) [212], the stepwise sparse regression (SSR) [24] and Bayesian approaches [158].

When matrix $\boldsymbol{\Xi}$ of the linear combination coefficients has been determined, the model for the evolution equations (5.1) is constructed as

$$\dot{\boldsymbol{y}} = \boldsymbol{\Xi}^T \left(\boldsymbol{\Theta}(\boldsymbol{y}^T)\right)^T. \tag{5.6}$$

This algorithm has been used for different applications such as identification of partial different equations [181, 180, 106], model reduction [123, 122, 121], dynamics identification [35], and control [31, 66, 107]. It will be used in this chapter to identify the existing dynamics in a flow, as detailed in section §5.3.

5.3 SINDy applied to the Navier-Stokes equation

In this paragraph we propose a method to discover dominant dynamics governing a fluid flow, by coupling the results of sparse linear regression with a community identification strategy that will be described in detail in section §5.4.2.

Consider the incompressible Navier-Stokes equation without external forcing

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} &= -\nabla p + \frac{1}{Re}\nabla^2\boldsymbol{u}, \\ \nabla\cdot\boldsymbol{u} &= 0, \end{aligned} \tag{5.7}$$

where $\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}$ are the inertial forces, $-\nabla p$ the pressure forces, and $\nabla^2\boldsymbol{u}$ the viscous force. The magnitudes of these terms determine the importance of such forces in different regions of the flow, allowing the identification of different regimes. For each discrete spatial point of the domain, the coefficients in front of these terms govern the behaviour of the system at that location. These local coefficients can be identified by using a data-driven approach similar to SINDy, described below.

Consider a two dimensional incompressible flow, governed by (5.7) with $\boldsymbol{u} \equiv (u, v)$, the momentum equation can be rewritten as a coupled system of equations for the horizontal and vertical velocity

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{5.8.1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \tag{5.8.2}$$

The procedure starts with the collection of the discrete state $\boldsymbol{q} = (\boldsymbol{u}, p) \in \mathbb{R}^n$ for a discrete number of time iterations n_t. A subset of spatial points of the domain $\mathcal{I} \in \mathbb{R}^{N_p}$ is selected randomly, forming $j = 1, \ldots, N_p$ data matrices $\mathbf{Y}_j \in \mathbb{R}^{n\times n_t}$. The spatial partial derivatives up to the second order are then computed using

Figure 5.1: Vorticity field for a flow around a cylinder, $Re = 200$, selection of three random points in the domain.

numerical differentiation at each point. The state and the spatial derivatives are nonlinearly combined to form the terms existing in the Navier-Stokes momentum equation (5.8), creating a library of candidate functions for each point j

$$\begin{aligned}\boldsymbol{\Theta}(\boldsymbol{q}) = \Bigg[\quad & \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \\ & \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}, \frac{1}{Re}\frac{\partial^2 u}{\partial y^2}, \frac{1}{Re}\frac{\partial^2 v}{\partial x^2}, \frac{1}{Re}\frac{\partial^2 v}{\partial y^2}, \\ & u\frac{\partial u}{\partial x}, v\frac{\partial u}{\partial y}, u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} \quad \Bigg].\end{aligned} \tag{5.9}$$

The N_p libraries of candidate functions are then used in the corresponding sparse linear regression algorithm to identify the active terms and their coefficients as in equation (5.5) of the SINDy algorithm.

We apply the algorithm to a two dimensional flow around a cylinder at $Re = 200$. Figure 5.1 shows the vorticity field of the flow, three random points are selected at different distances from the wake generated behind the cylinder.

After the regression, the points selected are defined by a set of coefficients indicating the active terms of the library of candidate function, and therefore the local active dynamics at their location. Figure 5.11 shows the result of the regression for the three selected points. The different active coefficients for each point suggest that these three regions might be governed by different dynamics.

Applying this strategy to a large set of random points, allows us to identify the existing dynamics on the domain as a whole. However this information needs to be classified in order to provide some physical insight. To this end, we would need to introduce a strategy which would allow us to regroup the points based on similar dynamical behaviour. For this purpose, we introduce a clustering strategy based

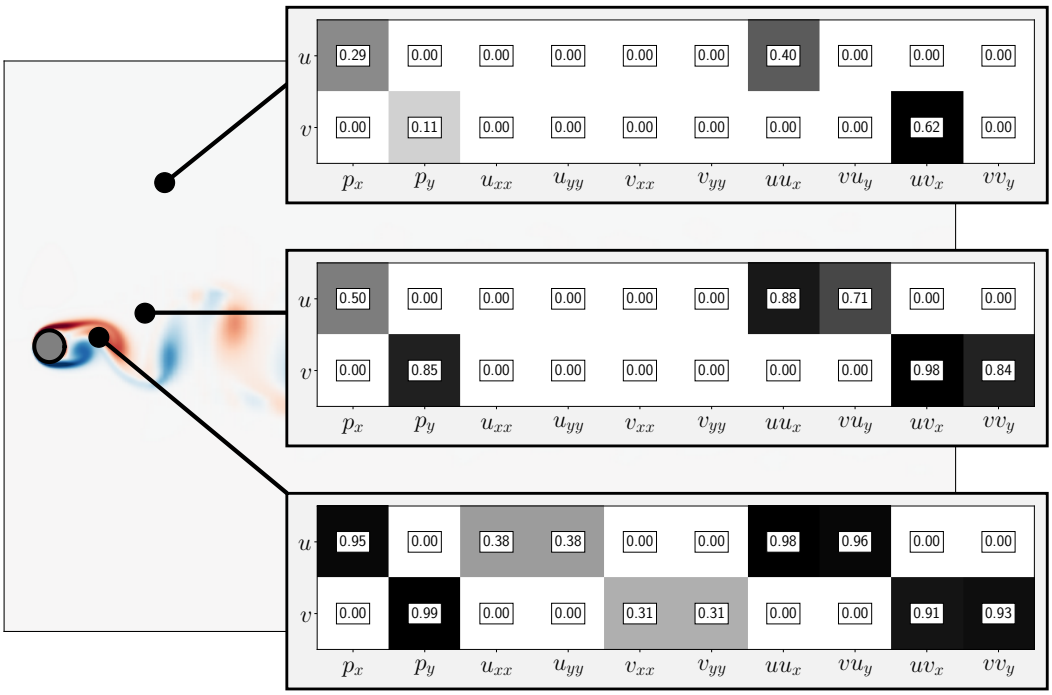


Figure 5.2: Identified active terms of the library of candidate functions for three random points.

on the relative euclidian distance on the subspace spanned by the set of terms defined in the library, in order to find communities in the flow that share the same dynamics. The algorithm chosen for the clustering procedure is the Leicht-Newman network, based on modularity optimization, and is introduced in the following section. One of the advantages of Leicht-Newman network is that it does not need an *a priori* knowledge on the number and size of the final clusters. This is a necessary requirement, which allows the identification process to remain user independent.

5.4 Clustering algorithms

In the context of machine learning, clustering algorithms belong to the unsupervised learning techniques. The goal of a clustering algorithms is to identify groups of data, sharing similar meaningful features within the original dataset. The partition in different clusters is done by assigning a label for each data point of the dataset.

Consider a set of data points $\boldsymbol{x}_j \in \mathbb{R}^n$, at the end of the clustering process each of these points will be provided a label $\boldsymbol{y}_j$, where $j = 1, ..., m$ with m the size of the dataset. Several clustering algorithms exist today in the literature, among the most common are *k-means*, *dendograms*, and *Gaussian Mixture models* algorithms [26, 27, 28, 169]. In the following we focus on a class of clustering algorithms which rely on identifying communities using networks.

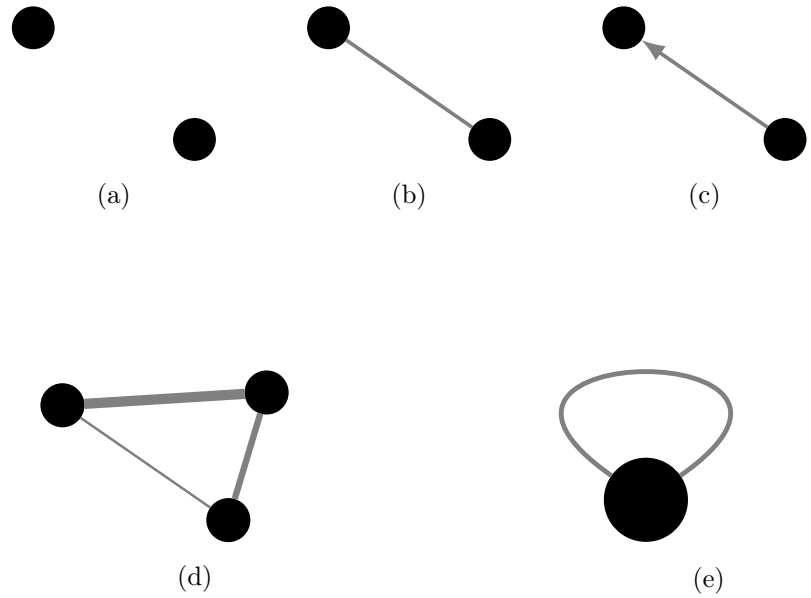


Figure 5.3: Graphic representation of nodes and edges of a network. (a) nodes; (b) nodes connected by a simple edge; (c) nodes connected by a directed edge; (d) nodes connected by weighted edges; (e) node with a self-edge.

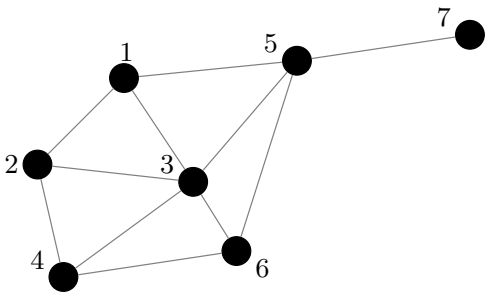
5.4.1 Networks

Networks, also called graphs in the literature, aim to find statistical properties among the components of a system, understand the manner by which these components interact, and eventually predict the behaviour of the system [145], through the identified interaction pattern. A network is a structure composed of a set of points, called *nodes* or *vertices*, linked together by a set of *edges*, graphically represented by lines. The edges of a network can be associated with weights, defining the importance of the connection between the nodes. In addition, the graph can take account of the *directivity* (or direction) of the edges, when the connection between two nodes is possible in one direction and not the opposite. A simple classification of networks can then be resumed in *weighted*, *directed* or *undirected* graphs. Directed graphs can also be *cyclic*, allowing self loops or edges from one node to itself, or *acyclic* when self loops are not allowed. Figure 5.3 shows the graphical representation of nodes and different types of edges which can exist in the network.

The structure of a network is mathematically described by its *adjacency matrix* $\mathbf{A}$, where for a network composed of N nodes, this matrix is $\mathbf{A} \in \mathbb{R}^{N\times N}$. In the case of an unweighted graph with N nodes and E edges the adjacency matrix is given by

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{5.10}$$

where the pair of indices (i, j) indicates a pair of nodes. When the graph is weighted and the weight of each edge is w_{ij}, with (i, j) the pair of nodes connected to each



(a)

A

0	1	1	0	1	0	0
1	0	1	1	0	0	0
1	1	0	1	1	1	0
0	1	1	0	0	1	0
1	0	1	0	0	1	1
0	0	1	1	1	0	0
0	0	0	0	1	0	0

(b)

Figure 5.4: Example of an undirected unweighted acyclic graph. (a) graph representation; (b) corresponding symmetric adjacency matrix.

other, the adjacency matrix is defined as

$$\mathbf{A}_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{5.11}$$

The *degree* of a vertex is defined as the summation of the number of edges connected to that vertex

$$k_i = \sum_{j=1}^{N} A_{ij}, \tag{5.12}$$

where A_{ij} is the element of an unweighted adjacency matrix. In the case where the adjacency matrix is weighted we talk about the *strength* of a vertex. For directed networks we can distinguish between *in-degree* and *out-degree*, respectively the number of ingoing and outgoing edges connected to a vertex

$$\begin{aligned} k_i^{in} &= \sum_{j=1}^{N} A_{ij}, \\ k_j^{out} &= \sum_{i=1}^{N} A_{ij}. \end{aligned} \tag{5.13}$$

The adjacency matrix of an undirected graph is symmetric. When self-connecting edges are allowed their contribution appears on the diagonal of the matrix, which is always empty for acyclic graphs. Figure 5.4 shows an example of an undirected unweighted and acyclic graph composed of $N = 7$ nodes. The edges are represented with the same width, meaning that they have the same weight. The adjacency matrix reported on the right has empty diagonal and it is symmetrical with respect to the diagonal. The elements highlighted in red represent the active connections between the nodes (edges). For more details regarding network construction, the reader is referred to the review papers [1, 59, 145] and books [12, 144, 58].

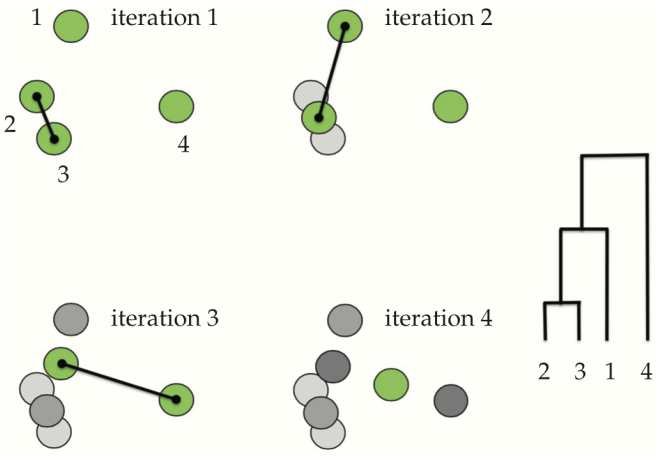


Figure 5.5: Example of an agglomerative hierarchical clustering scheme applied for four data points. The dendogram on the right shows how the process generates a summary of a hierarchical clustering. Image reproduced from [27]

Networks show community structure [51], where vertices belonging to the same community are represented with high density of connections. In general, nodes belonging to the same community have features in common. This property, eventually coupled with graph sparsification where less important edges are removed, thereby promoting connections amongst nodes of the same group, allows the use of network science for clustering purposes.

5.4.2 Community detection - Leicht-Newman network

The presence of community structures in networks provides additional knowledge on the system of interest, aiming to identify different groups of nodes showing similar properties and therefore highly linked to one another. The problem of community detection is quite challenging. Starting from an apparently random graph, the first approach would be to separate the graph into two densely connected communities of equal size. However in real complex networks the number or the size of the existing communities are not known a priori.

Techniques like hierarchical clustering can be employed to discover how a network breaks down into communities based on the existing connections between its vertices. These methods are based on the assumption that a random network does not exhibit community structures, and they are classified into *agglomerative* and *divisive* methods.

In agglomerative methods the procedure starts with a network with no edges but one connection between two vertices with high similarity, then edges are iteratively added between the other nodes of the network, creating a complete hierarchical graph that can be represented by a *dendrogram*, as shown in figure 5.5.

In divisive methods, on the other hand, the real network with its original connections is considered. Then, the edges between the least similar pairs of nodes are removed iteratively. Divisive methods for hierarchical clustering have been widely studied by Newman who initially used the concept of *betweenness* [70], where high values are attributed to edges between communities and low values to those lying inside the same community [79, 148]. Following this method, the betweenness score is first computed for all the edges, then the edge with the highest betweenness is removed from the network modifying the structure of the graph and consequentially requiring a new computation of the betweenness for the remaining edges. The algorithm is then repeated until a stopping criteria is satisfied. To impose a stopping criteria, a new measure of the quality of the division of a network called *modularity* is introduced [146, 148]. Modularity is defined as the number of edges connecting nodes belonging to the same community minus the expected number of edges in an equivalent network with random connections. For community detection, the value of the modularity has to be positive and ideally large. Large values of modularity indicates that the number of edges existing within the same community is higher than the expected number on the basis of chance, emphasising more important connections inside the same cluster of nodes. The community detection process can then be posed as a modularity maximization problem.

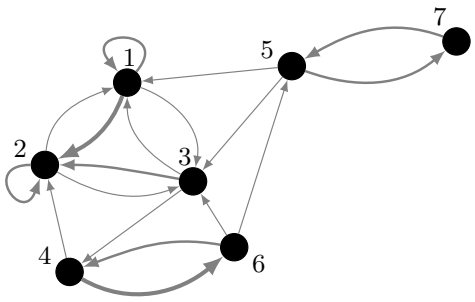
Consider a network containing n vertices with edges between node i and node j represented by the elements of the adjacency matrix A_{ij}. The network is supposed to be initially divided in two groups following a divisive procedure. The corresponding expected number of edges between the nodes, if the edges are randomly placed, is $k_i k_j / 2m$, where k is the degree of freedom of a node defined in (5.12) and $m = \sum_i k_i/2$ is the total number of edges in the network [147]. The *modularity matrix* can then be defined as

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{5.14}$$

The modularity is given by

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta_{c_i c_j}, \tag{5.15}$$

where δ_{ij} is the Kronecker delta and c_i is the community to which node i belongs. The modularity Q is then maximized over possible divisions of the network into communities, and its maximum value corresponds to the best estimate of the true communities. The maximization of the modularity can be done by using spectral methods [147], focusing on the eigenvectors of the modularity matrix $\mathbf{B}$. This procedure is similar to the spectral partitioning used in graph partitioning [164]. The spectral optimization method of Newman [147] has proven to give excellent results with low computational costs for applications to undirected networks. Before the introduction of the Leicht & Newman network, this method was employed indistinctly for undirected and directed networks, ignoring the directions of the edges and discarding information about the structure of the network.



(a)

A

w_{11}	w_{12}	w_{13}	0	0	0	0
w_{21}	w_{22}	w_{23}	0	0	0	0
w_{31}	w_{32}	0	w_{34}	0	0	0
0	w_{42}	0	0	0	w_{46}	0
w_{51}	0	w_{53}	0	0	0	w_{57}
0	0	w_{63}	w_{64}	w_{65}	0	0
0	0	0	0	w_{75}	0	0

(b)

Figure 5.6: Example of an directed weighted cyclic graph. (a) graph representation; (b) corresponding asymmetric adjacency matrix: the edges have different weights highlighted with different shades of red, the contribution of self-loops appears on the diagonal.

In [118], Leicht & Newman propose an extension of the spectral method to directed networks, now taking into account the directions of the edges. Figure 5.6 gives an example of a weighted directed network, with the corresponding adjacency matrix. In this case the degrees of the nodes involved in (5.14) are replaced by the in- and out-degrees defined in (5.13). The modularity matrix then becomes

$$B_{ij} = A_{ij} - \frac{k_i^{in} k_j^{out}}{m}, \tag{5.16}$$

and the modularity for the considered directed network is

$$Q = \frac{1}{m} \sum_{ij} \left(A_{ij} - \frac{k_i^{in} k_j^{out}}{2m} \right) \delta_{c_i c_j}, \tag{5.17}$$

here c_i denotes the communities and $m = \sum_{ij} A_{ij} = \sum_i k_i^{in} = \sum_j k_j^{out}$. Notice that for directed networks the modularity matrix $\mathbf{B}$ is not symmetrical as opposed to the adjacency matrix $\mathbf{A}$. This lack of symmetry affects the eigenvalues computed by the spectral method.

A directed network composed of n nodes is initially subdivided into two communities, where a vector $\boldsymbol{s}$ of labels is introduced. The labels are set to be $s_i = +1$ when i belongs to one group and $s_i = -1$ when it belongs to the other, then $\sum_i s_i^2 = n$, $\delta_{c_i c_j} = \frac{1}{2}(s_i s_j + 1)$ and (5.17) is recast as

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i^{in} k_j^{out}}{2m} \right) (s_i s_j + 1) \\ &= \frac{1}{2m} \sum_{ij} s_i B_{ij} s_j = \frac{1}{2m} \boldsymbol{s}^T \mathbf{B} \boldsymbol{s}. \end{aligned} \tag{5.18}$$

The first operation to be done is to restore the symmetry of the problem, adding the transpose of the modularity matrix $\mathbf{B}$ to equation (5.18), resulting in

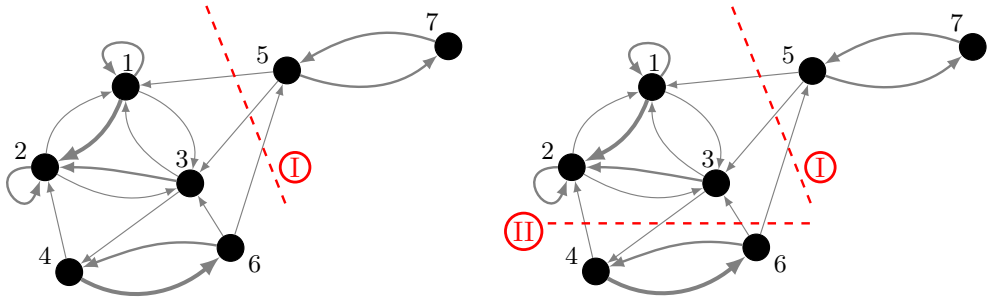
$$Q = \frac{1}{4m} \boldsymbol{s}^T \left(\mathbf{B} + \mathbf{B}^T \right) \boldsymbol{s}, \tag{5.19}$$

where $\mathbf{B} + \mathbf{B}^T$ is symmetric. Following the strategy used in [147], the label vector $\boldsymbol{s}$ can be rewritten as a linear combination of the eigenvectors of $\mathbf{B} + \mathbf{B}^T$, such that $\boldsymbol{s} = \sum_i a_i \mathbf{v}_i$, and $\mathbf{v}_i$'s are eigenvectors corresponding to the eigenvalues β_i, and $a_i = \mathbf{v}_i^T \cdot \boldsymbol{s}$. The modularity can then be reformulated as

$$\begin{aligned} Q &= \sum_i a_i \mathbf{v}_i^T \left(\mathbf{B} + \mathbf{B}^T \right) \sum_j a_j \mathbf{v}_j \\ &= \sum_i \beta_i \left(\mathbf{v}_i^T \cdot \boldsymbol{s} \right)^2 . \end{aligned} \tag{5.20}$$

Assuming that the eigenvalues are labeled in decreasing order: $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$, the maximum of Q obtained by the maximization of equation (5.20) is found when the vector $\boldsymbol{s}$ is parallel to the leading (largest) eigenvector $\mathbf{v}_1$. This solution is however unachievable, since the elements of $\boldsymbol{s}$ are restricted to be $s_i = \pm 1$. Therefore $\boldsymbol{s}$ is instead chosen to be as close as possible to the first eigenvector of $\mathbf{B} + \mathbf{B}^T$. The maximization problem (5.20) is reformulated as the maximization of the product $\mathbf{v}_1^T \cdot \boldsymbol{s}$. Since the eigenvectors are orthogonal to each other, the terms corresponding to the lower eigenvalues are equal to zero and they can be ignored. Considering $i = 1, \ldots, n$ elements of the first eigenvector $v_i^{(1)}$, if $v_i^{(1)} > 0$ then $s_i = +1$, and if $v_i^{(1)} < 0$ then $s_i = -1$. In the limit of $v_i^{(1)} = 0$ then $s_i = \pm 1$ is imposed. This process assigns a label to each node of the network, dividing all the nodes into the two communities closest to the truth. An additional stage of *fine tuning* is added at the end of this process by moving the vertices between communities to further increase the value of the modularity until a maximum is reached. Once the first two communities are detected, with $G = 2$ being the number of communities, a repeated bisection strategy is used. The algorithm is applied again for each of these communities in order to eventually divide them into two corresponding sub-communities and so forth until no further division increases the value of the total modularity of the system as represented in figure 5.7. Giving a label g to one of these two community, the subdivision within g will generate a change in the modularity defined as

$$\begin{aligned} \Delta Q &= \frac{1}{2m} \left[\sum_{i,j \in g} \left(B_{ij} + B_{ji} \right) \frac{s_i s_j + 1}{2} - \sum i, j \in g \left(B_{ij} + B_{ji} \right) \right] \\ &= \frac{1}{4m} \sum_{i,j \in g} \left[\left(B_{ij} + B_{ji} \right) - \delta_{ij} \sum_{k \in g} \left(B_{ik} + B_{ki} \right) \right] s_i s_j \\ &= \frac{1}{4m} \boldsymbol{s}^T \left(\mathbf{B}^{(g)} + \mathbf{B}^{(g)^T} \right) \boldsymbol{s}, \end{aligned} \tag{5.21}$$



(a) (b)

Figure 5.7: Repeated bisection following the Leicht-Newman network procedure (a) the network is initially divided in two communities (5.20) (bisection I); (b) the community on the left is in turn further divided in two communities using the generalized formulation of the modularity for multiple groups (5.21) (bisection II).

where $\mathbf{B}^{(g)}$ is the submatrix of the original modularity matrix $\mathbf{B}$ for the community g. The method consists then in maximizing ΔQ by applying the spectral partitioning method described earlier to all the existing communities G, and adding a fine-tuning step at the end of each subdivision until $\Delta Q \leq 0$. The community detection procedure is summarized in algorithm 4, and an example of the repeated bisection procedure is illustrated in figure 5.7.

For further details on community identification methods and in particular on the modularity maximization via spectral method, the reader is referred to the work of Newman *et al.* [146, 79, 147, 118].

As mentioned previously, the numerical procedure for dynamical identification is done in two stages. The dominant dynamics are first identified in the full domain using sparse linear regression, described in section §5.3. Then, the points are clustered in different communities by using the Leicht-Newman network. In this section, in order to validate the Leicht-Newman network for community detection on its own, a stochastic model that does not involve the linear regression step is considered. The full procedure, combining linear regression and community identification, is then analysed in the following section.

5.4.3 Stochastic model of turbulent axisymmetric wakes behind a bluff body

To validate the Leicht-Newman network for community detection, we consider the stochastic model of a turbulent wake generated by the flow around an axisymmetric bluff body at high Reynolds numbers, proposed by Rigas *et al.* [175, 174], where large-scale structures responsible for symmetry breaking in a wake at high Reynolds numbers ($Re \sim 2 \times 10^5$) are first investigated experimentally (figure 5.8a illustrates the experimental setup used in [175]). The results show that the magnitude of the

Algorithm 4 Modularity optimization for community detection in directed Leicht-Newman networks

Input Adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$

Output Communities labels $c_i \in \{1,\dots,G\}$ for each node of the network $i = 1,\dots,n$

```
Compute modularity matrix: B (5.17)
Restore symmetry by adding B + B^T
Find most positive eigenvalue β_1 and the corresponding eigenvector v_1 of B+B^T
if v_i^(1) > 0 then
    s_i = +1
else if v_i^(1) < 0 then
    s_i = −1
end if
Fine-tune: move vertices between assigned communities
for g = 1,...,G do
    while ΔQ > 0 do
        Subdivide community g by maximizing (5.21)
        Fine-tune within g
    end while
end for
```

asymmetry of the wake in the turbulent regime can be quantified by the position of the centre-of-pressure (CoP) at the base of the bluff body. The time evolution of the position of the CoP is calculated from pressure measurements from 64 static pressure taps at the base of the bluff body, space averaged on the base surface A

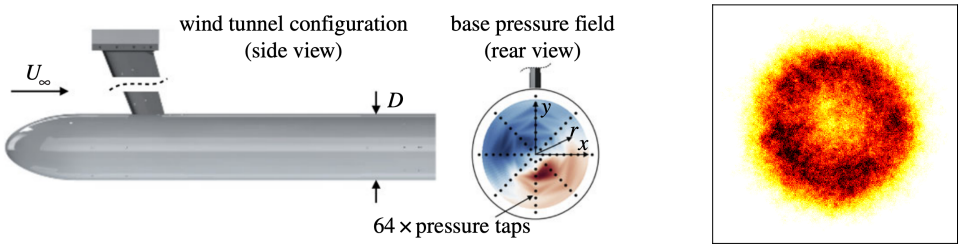
$$\boldsymbol{R}(t) = \frac{1}{D\int p(t)\mathrm{d}A}\int p(t)\boldsymbol{x}\mathrm{d}A. \tag{5.22}$$

Here, $\boldsymbol{x}$ represents the cartesian coordinate of the CoP and D is the bluff body diameter. When CoP lies at the center of the surface A, the flow is symmetric, while departure from this value produces an asymmetry in the wake. As reported in figure 5.8b, the probability density function (PDF) of the CoP position on A shows that the CoP spends most of the time at a non-zero radial location, while preserving a uniform distribution on the angle. Therefore the wake is asymmetric, but by long-time averaging an axisymmetric pressure field is obtained.

The dynamics of the turbulent wake can be modelled by a nonlinear two-dimensional Langevin equation [174], extending the weakly nonlinear models of laminar and linearly stable axisymmetric wakes, which have been proven to lose spatial symmetry in the azimuthal direction due to a supercritical pitchfork bifurcation [62, 139]

$$\dot{\boldsymbol{x}} = \alpha\boldsymbol{x} + \lambda\boldsymbol{x}|\boldsymbol{x}|^2, \tag{5.23}$$

when $\alpha > 0$. In the turbulent adaptation of the laminar model (5.23), the effect of the turbulent background fluctuations is taken into account as stochastic forcing ξ



(a) (b)

Figure 5.8: Experimental configuration for the axisymmetric wake used in [175]. (a) the bluff body is mounted from the wind tunnel ceiling. (b) Center of Pressure probability distribution.

and added to the deterministic system as independent white noise

$$\begin{cases} \dot{x} = \alpha x + \lambda x \left(x^2 + y^2\right) + \sigma \xi_x(t), \\ \dot{y} = \alpha y + \lambda y \left(x^2 + y^2\right) + \sigma \xi_y(t), \end{cases} \tag{5.24}$$

where σ^2 is the variance of the random forcing, taken to be rotationally symmetric $\sigma \equiv \sigma_x \equiv \sigma_y$. In polar coordinates (r, ϕ), with $r = \sqrt{x_2 + y^2}$ being the amplitude and $\phi = \tan^{-1}(y/x)$ the phase, the Langevin system (5.24) becomes independent of the phase,

$$\begin{cases} \dot{r} = \alpha r + \lambda r^3 + \dfrac{\sigma^2}{2r} + \sigma \xi_r, \\ \dot{\phi} = \dfrac{\sigma}{r} \xi_\phi. \end{cases} \tag{5.25}$$

Consider $\boldsymbol{x}$ as the coordinates of the CoP, the polar coordinate (r, ϕ) now represents the distance from the center of the round surface of the bluff body and the corresponding angle. The temporal evolution of the location of the CoP can then be described by (5.25).
Starting from this model, we aim to use Leicht-Newman clustering algorithm to reconstruct the PDF of the evolution of the CoP. As already mentioned, the PDF resulting from the polar Langevin in equation (5.25) is symmetric with respect to the phase, making this case unfavourable for clustering purposes.

The original Langevin system is then modified by adding a nonlinear term in the evolution equation for the phase. Notice that this modification has no physical meaning and it does not reflect the real dynamics of the CoP. The purpose here is to promote a bimodal behaviour that will be finally identified by the clustering algorithm.

$$\begin{cases} \dot{r} = \alpha r + \lambda r^3 + \dfrac{\sigma^2}{2r} + \sigma \xi_r, \\ \dot{\phi} = \dfrac{\sigma}{r} \xi_\phi + \gamma \sin^2(\phi). \end{cases} \tag{5.26}$$

The coefficients of the Langevin equation are obtained from the experimental data [174] and $\gamma = 10$. The system of equations (5.26) is solved for $n = 3 \times 10^6$

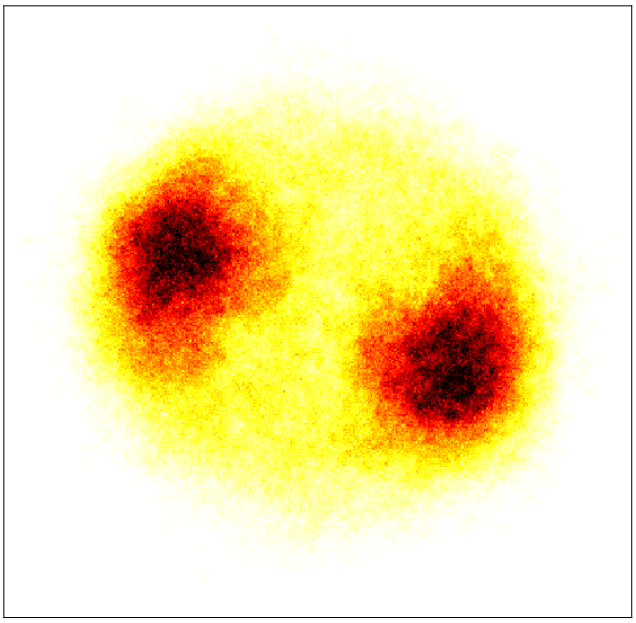


Figure 5.9: Probability distribution for the evolution of the Center of Pressure obtained by the modified system (5.26).

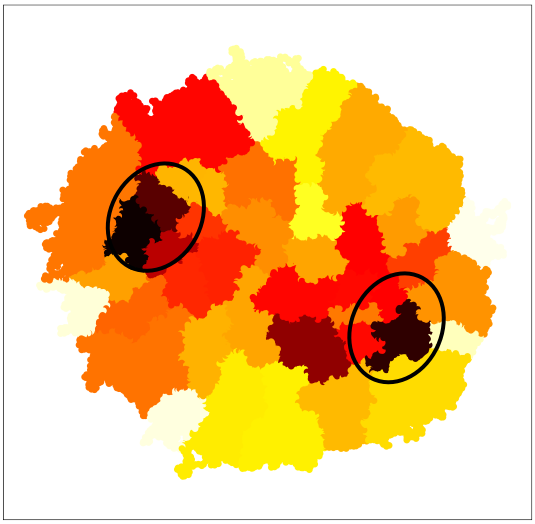
iterations, the PDF for the evolution of the CoP coordinates is shown in figure 5.9 and presents two well defined high density regions.

The spatial coordinates (r, ϕ) are stored for each time step and they are fed into a k-means algorithm to create a first set of $N = 200$ small clusters using euclidean distance on the surface. The purpose of using the Leicht-Newman network is to identify communities among these clusters in order to consider the trajectory followed during the motion and understand how and how often the CoP moves from one region to another.

The clusters are then selected to be the nodes of a graph, the edges being the transitions of the trajectory from one cluster to the other. These crossings can happen multiple times and in opposite directions, self-edges are allowed and used when the CoP does not move to a different cluster. The graph obtained is weighted, directed, and cyclic, and the weights of the edges take into account the number of transitions between the initial subdivisions of the domain.

The corresponding weighted asymmetric adjacency matrix is fed into the Leicht-Newman algorithm to discover communities among the nodes. The detected communities are showed and compared to the original histogram in figure 5.10. The algorithm is able to find high density regions that are slightly different than the ones highlighted in the histogram. This is due to the additional information on the trajectory of the motion described by the stochastic model, showing not only how many times the CoP falls into a certain cluster but also giving a statistical insight on what cluster it is more likely to move towards.

(a)



(b)

Figure 5.10: Community identification. (a) Initialization of 200 clusters with the *k-means* algorithm; (b) Communities identified by the Leicht-Newman network.

5.5 Combining regression and clustering procedures for dynamical identification

In this section the regression algorithm (based on SINDy) presented in section §5.3 is combined with the clustering algorithm presented above, resulting in a dynamical identification procedure, suing the underlying Navier-Stokes equation.

To assess the algorithm, the same case as considered in section §5.3: the two dimensional flow around a cylinder at $Re = 200$ is analysed. The vorticity field of the flow is shown in figure 5.11, and the domain is defined for $-2 \leq x \leq 30$ and $-9 \leq y \leq 9$.

We aim to discover the different dynamics existing in the flow using the procedure introduced in section §5.3. The discrete state is collected for $n_t = 340$ time iterations, covering one full shedding period. The grid is staggered and refined around the cylinder and its wake, and the surface of the cylinder is defined by Lagrangian points. In staggered grids the pressure is defined at the center of the cells, while the velocity components are defined on its boundaries. To be able to define all the variables at the same location, the velocity field is interpolated at the center of each cell before selecting the points. A subset $\mathcal{I}$ of $N_p = 3000$ points is selected inside the domain. The cartesian points inside the cylinder surface and the immersed boundary points are excluded from the procedure. The relative distance between the points in the subset is twice the minimum cell size, inducing a higher concentration of points in the regions where the grid is refined.

When the points are selected, the temporal spatial derivatives are numerically approximated and the library of candidate function is created. In particular, a library of candidate function is defined for each point in the subset.

$$\begin{aligned}\boldsymbol{\Theta}(\boldsymbol{q}^{(j)}) = \Big[\quad & p_x^{(j)}, p_y^{(j)}, \\ & \frac{1}{Re}u_{xx}^{(j)}, \frac{1}{Re}u_{yy}^{(j)}, \frac{1}{Re}v_{xx}^{(j)}, \frac{1}{Re}v_{yy}^{(j)}, \\ & uu_x^{(j)}, vu_y^{(j)}, uv_x^{(j)} + vv_y^{(j)} \quad \Big].\end{aligned} \tag{5.27}$$

For simplicity, the subscript indicates the partial derivative, for $j = 1, \ldots, N_p$ points. The temporal evolution of the flow at each point can be rewritten as

$$\dot{\boldsymbol{q}}^{(j)} = \boldsymbol{\Theta}(\boldsymbol{q}^{(j)})\boldsymbol{\Xi}^{(j)}, \tag{5.28}$$

where $\boldsymbol{\Xi}^{(j)}$ are the coefficients of the polynomial combination for the generic point $j \in \mathcal{I}$. The identification of the active terms of the polynomial combination is performed by using a Sequential Threshold Ridge Regression (STRidge) introduced in [181] and adapted for group sparsity. In this algorithm, the classic Ridge regression

$$\boldsymbol{\xi} = \underset{\xi'}{\operatorname{argmin}} \|\boldsymbol{\Theta}\boldsymbol{\xi}' - \dot{\boldsymbol{q}}\|_2 + \lambda\|\boldsymbol{\xi}'\|_2, \tag{5.29}$$

is followed by a recursive penalization in ℓ_0, removing coefficients smaller than a certain threshold to avoid overfitting.

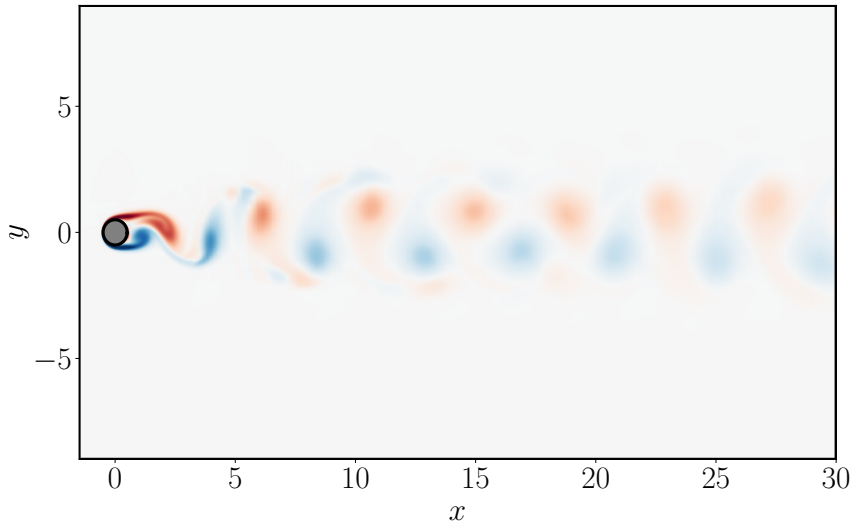


Figure 5.11: Vorticity field for a flow around a cylinder, $Re = 200$.

After the regression, each point is represented in the coefficient subspace by its coordinates $\boldsymbol{\Xi}^{(j)}$, and is set to be a node of the Leicht-Newman network. The edges are defined according to their relative distance in the coefficient subspace. The edges are automatically removed when the distance between a pair of points is higher than a certain value. Here, this value is set to the 10% of the maximum distance among all the points included in the procedure.
The communities identified by the Leicht-Newman network are shown in figure 5.12a. Three main clusters emerge from the community detection, defining a free stream region (blue-green) where the flow is not perturbed by the presence of the cylinder, a central region behind the cylinder (violet) where the von Karman street is evolving, and an intermediate layer (orange) between the two. As we move away from the cylinder along the positive direction of the x-axis, the central and the intermediate clusters start to merge, due to the reduced intensity of the vorticity field. The rearranged adjacency matrix of the final clusters is shown in figure 5.12b. Here, we detect the three clusters, each highlighted in its corresponding color. The adjacency matrix shows correlations between clusters (gray areas), meaning that the dynamics governing each cluster can affect the other and that the clusters are not strictly independent. As an example, in figure 5.12 we show the active coefficients recovered by the linear regression for three points belonging to the different clusters. To generate the figure, each set of coefficients has been normalized with respect to its maximum value. In the wake, right behind the cylinder, all the terms on the right hand side of the Navier-Stokes equation are active, while moving away from the wake, the viscous forces reduce their intensity until they deactivate. In the free stream the pressure and the inertial forces are active, but their values are much smaller than their corresponding values in the other clusters. For comparison, the numerical values of the coefficients for these points are reported in table 5.1.

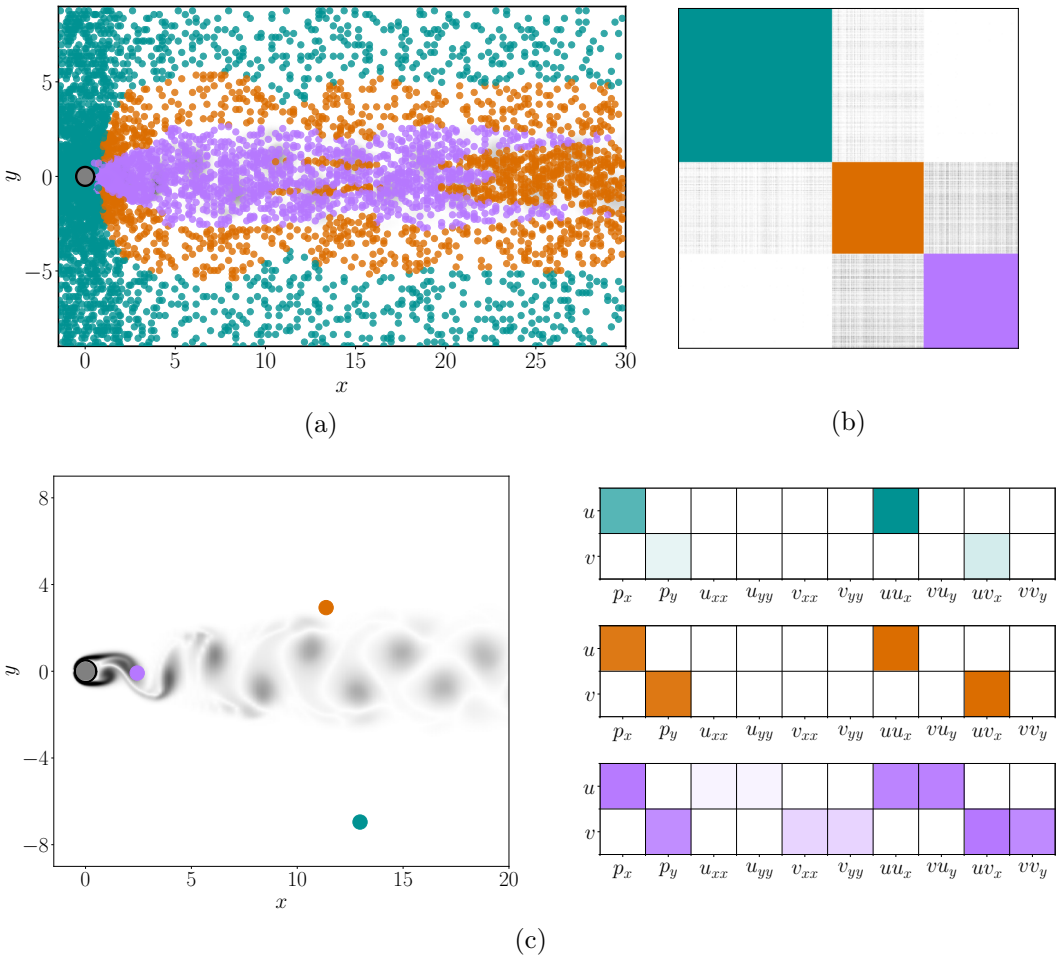


Figure 5.12: Dynamics identification on a two dimensional flow around a cylinder at $Re = 200$. (a) the selected points are divided in three main communities sharing same dynamics; (b) rearranged adjacency matrix showing correlations between communities; (c) example of active coefficients identified for points belonging to different communities, in the figure the values are normalized for each point.

	Cluster 1		Cluster 2		Cluster 3	
	u	v	u	v	u	v
p_x	0.013		0.89		1.015	
p_y		0.002		0.89		0.879
$\frac{1}{Re}u_{xx}$					0.135	
$\frac{1}{Re}u_{yy}$					0.135	
$\frac{1}{Re}v_{xx}$						0.361
$\frac{1}{Re}v_{yy}$						0.361
uu_x	0.02		0.97		0.946	
vu_y					0.946	
uv_x		0.004		0.97		1.035
vv_y						0.899

Table 5.1: Numerical values of the identified coefficients for points belonging to different communities.

5.6 Conclusions

In this chapter we proposed a procedure to identify the dominant dynamics existing in a flow. The proposed procedure combines sparse linear regression for system identification and network partitioning to cluster the identified regimes. Information regarding the dynamics of the flow has been included in the identification process by the introduction of a costumized library of candidate functions. In this way, the dynamics at each point of the domain was well defined by the parameters recovered with the regression step. These dynamics were then regrouped using network science, and in particular community detection with the Leicht-Newman network. The advantage in using the Leicht-Newman network to identify different groups of dynamics is that the number or the size of the communities must not be known a priori.

The procedure has been validated on a two dimensional flow around a cylinder, showing promising preliminary results. When applied on more complex systems, this procedure could improve the quality of the description of the flow, by allowing the employment of different models and approximations for different regimes existing in the flow. For example, the equations describing the system can be dimensionally reduced by employing different ROMs when the dynamics affecting the full flow are spatially well identified. On the other hand, in control applications the information on the interactions between different dynamics can have a significant role in the selection of the design parameters, allowing a more detailed understanding of the effect of the control and then reducing the complexity of the problem.

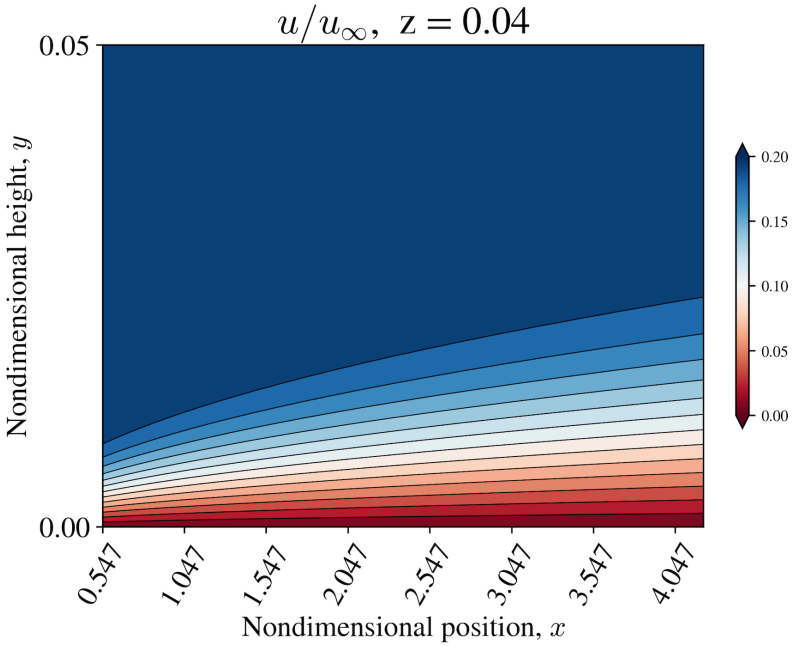


Figure 5.13: Two dimensional slice of a three dimensional boundary layer flow, $Re = 1 \times 10^5$, $Ma = 0.2$.

Applications on more complex flows are being evaluated. In the next stage, the algorithm will be performed on an unperturbed boundary layer at steady state for $Re = 1 \times 10^5$ and $Ma = 0.2$. Figure 5.13 represents the horizontal velocity field on a two dimensional slice. This data was generated using a three-dimensional compressible Navier-Stokes solver and kindly provided by Athanasios T. Margaritis. More complex flow scenarios including transitional and turbulent boundary layers at higher Mach numbers are envisioned as the following steps.

Chapter 6

Conclusions and perspectives

This thesis studies and proposes different strategies to combine CFD with state-of-the-art optimization methods, overcoming the bottleneck given by the high computational time associated to the optimization procedure. The proposed algorithms can be employed both on derivative-free and gradient-based optimization, and they can eventually be used to combine the two methodologies, leading to hybrid and multi-fidelity optimization strategies. The formalism leading to the derivation of the optimality problem for gradient-based algorithms is presented and applied in chapter 2 using adjoint techniques. The computational performance of this algorithm is investigated and its resolution is accelerated by the implementation of a parallel-in-time algorithm relying on Krylov-based exponential time integrator in chapter 3. Additionally, the thesis contains two data-driven methods that can be employed to ease the optimization problem. Chapter 4 presents projection-based ROMs, considering the problems related to the stabilization of Galerkin projection methods and the adaptivity required for parameters variation. Finally, in chapter 5 we introduced a data-driven algorithm used to identify dominant dynamics in a flow, combining system identification with clustering algorithms. The numerical simulations used throughout the thesis are done using a two-dimensional incompressible Navier-Stokes solver with immersed boundaries [52].

Concerning gradient-based optimization, and in particular adjoint-based algorithms, to extract the gradient, the computational cost associated with the optimization procedure can be reduced employing ROMs for the forward and backward problems. The use of ROMs on unsteady and nonlinear dynamical systems implies neglecting some information about the flow, and leads to the extraction of an approximated gradient. As an alternative, to allow the use of adjoint equations with high-fidelity simulations, in chapter 3 we propose a time parallelization strategy, able to reduce the time to solution. The algorithm proposed here is the extension of the hybrid direct-serial-adjoint-parallel algorithm introduced by Skene *et al.* [193] to ADEs, such as incompressible Navier-Stokes equations.

The performance of the parallel algorithm and the reduction of the overall time to solution is evaluated on adjoint-based optimization problems in chapter 2, involving passive (steady) and active (unsteady) control. In the case of steady control for drag reduction, we were able to almost completely eliminate the time needed for the gradient extraction. Increasing the number of processors involved in the problem, the gradient is extracted almost at the same time as the cost function, cutting the total time to solution for one direct-adjoint loop almost

by half. When unsteady control is imposed, i.e. the design variables are time dependent, the algorithm becomes less efficient, but still the gain in compute time is still appreciable. The reduced gain obtained with unsteady control is due to the time dependency of the operators involved in the optimality system and it can not be improved further.

Both derivative-free and gradient-based optimization algorithms improve their solution iteratively, by moving the design variable towards the optimum. This involves multiple function evaluations, in our case represented by a full, computationally expensive CFD unsteady simulation. While it is desirable to optimise a design using only the high-fidelity model, in cases where high-fidelity simulations prove to be too expensive, lower-fidelity models may provide valuable information that can accelerate the optimisation process. Chapter 4 investigates ROMs based on Galerkin and Petrov-Galerkin projection, to approximate the high-fidelity simulation with a low-fidelity counterpart. However, these methods proved to lack robustness when considering changes in the parameter space. We built a ROM for a two dimensional incompressible flow around a cylinder at variable Reynolds numbers using the GNAT algorithm proposed by Carlberg *et al.* [41]. The results obtained show an accurate reproduction of the flow with a dimensional reduction of 93%. In addition an interpolation on the tangent space to the Grassmann manifold was implemented to make the ROM adaptive to parameter changes.

Using ROMs in optimization algorithms, derivative-free or gradient-based, drastically reduces the cost of evaluating the function or the gradient, accelerating the optimization procedure. The solution to a robust multi-fidelity optimization algorithm, applicable to unsteady multi-physics flow problems, lies in the accuracy of the low-fidelity model. The more accurate the low-fidelity model, the less need for high-fidelity function calls.

In the first chapters of the thesis we proposed methodologies directly applicable to the optimization problem, however another study was conducted to discover the underlying dynamics governing a fluid flow. As repeatedly stated throughout the thesis, in many regimes, the dynamics of the flow are governed by few dominant dynamical features. The capability to identify and locate the different dynamics present in a complex system can help improve the optimization procedure. In chapter 5 we developed an algorithm capable to identify the active terms in the Navier-Stokes equation governing the behaviour of the flow. The algorithm combines sparse linear regression for system identification with clustering algorithm. As a result, we were able to identify the dynamics existing in a two dimensional flow and the interaction occurring between them, while more complex flow applications are currently being explored. In control applications, additional insight on the mechanism governing a system and eventually their interactions can have a significant role in the selection of the design parameters, the placement of the sensors, and the possibility to replace the high-fidelity simulation with ROMs, allowing a more detailed understanding of the effect of the control and then

reducing the complexity of the problem.

Whilst this thesis has met the objective of proposing strategies to reduce the computational cost associated with state-of-the-art optimisation techniques, further studies can be conducted. The algorithms presented here showed promising results on the applications proposed, however in order to apply them to real large-scale problems, we need to enable the extension of these methods to three dimensional complex flows.

To exploit the computational power, the parallel-in-time algorithm could be applied in combination with spatial parallelization, making the most of every processor available. Additionally, this method could be employed in hybrid optimization algorithms, combining gradient-based and stochastic derivative-free algorithms, to accelerate the extraction of the gradient when needed as proposed by Quiros *et al.* [167].

The ROMs investigated in chapter 4 can be extended to multiple parameters variation and included in the optimization problem. In particular, for adjoint-based algorithm, building the ROM for the inverse problem is not trivial, while the use of a ROM for the direct problem is straightforward. The application of ROMs to stochastic derivative-free optimization is more evident, however the interpolation of the basis on the tangent space to the Grassmann manifold could be improved using an adaptive strategy. When the function on a new operating point has to be evaluated, and this evaluation is done by using a high-fidelity simulation, this point can be added to the interpolation dataset to incrementally improve the quality of the interpolated ROM.

Additional investigations have to be done for the dynamics identification algorithm, to validate it on more complex flows, and increasing its robustness with respect to complex dynamics, different regimes, turbulence and the effect of compressibility.

Further research should be concentrated on how to include all the presented methods in a hybrid/multi-fidelity dynamic optimization algorithm, capable of enabling and disabling these tools when possible.

Bibliography

[1] Albert, R., and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics 74*, 1 (2002), 47. (Cited on page 90.)

[2] Alexandersen, J., and Andreasen, C. S. A review of topology optimisation for fluid-based problems. *Fluids 5*, 1 (2020), 29. (Cited on page 2.)

[3] Alexandrov, N., Lewis, R., Gumbert, C., Green, L., and Newman, P. Approximation and model management in aerodynamic optimization with variable-fidelity models. *AIAA J 38*, 6 (2001), 1093–1101. (Cited on page 10.)

[4] Amsallem, D. *Interpolation on manifolds of CFD-based fluid and finite element-based structural reduced-order models for on-line aeroelastic predictions*. Stanford University, 2010. (Cited on pages 13, 56 and 62.)

[5] Amsallem, D., and Farhat, C. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA journal 46*, 7 (2008), 1803–1813. (Cited on pages 13, 56 and 61.)

[6] Antoulas, A. Approximation of large-scale dynamical systems. *SIAM* (2005). (Cited on page 10.)

[7] Argaud, J.-P., Bouriquet, B., De Caso, F., Gong, H., Maday, Y., and Mula, O. Sensor placement in nuclear reactors based on the generalized empirical interpolation method. *Journal of Computational Physics 363* (2018), 354–370. (Cited on page 56.)

[8] Arnoldi, W. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics 9* (1951), 17–29. (Cited on page 44.)

[9] Aubry, N., Holmes, P., Lumley, J. L., and Stone, E. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of fluid Mechanics 192* (1988), 115–173. (Cited on pages 12 and 67.)

[10] Baglama, J., Calvetti, D., and Reichel, L. Fast leja points. *Electron. Trans. Numer. Anal 7*, 124-140 (1998), 119–120. (Cited on page 45.)

[11] Bakewell, H. P., and Lumley, J. L. Viscous sublayer and adjacent wall region in turbulent pipe flow. *Phys. Fluids 10*, 9 (1967), 1880–1889. (Cited on pages 11 and 55.)

[12] Barabási, A. *Network Science*. Cambridge University press, 2016. (Cited on page 90.)

[13] Barone, M. F., Kalashnikova, I., Segalman, D. J., and Thornquist, H. K. Stable galerkin reduced order models for linearized compressible flow. *Journal of Computational Physics 228*, 6 (2009), 1932–1946. (Cited on page 12.)

[14] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. An 'empirical interpolation'method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique 339*, 9 (2004), 667–672. (Cited on pages 11, 13, 56 and 67.)

[15] Benner, P., Gugercin, S., and Willcox, K. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review 57*, 4 (2015), 483–531. (Cited on page 57.)

[16] Bergamaschi, L., Caliari, M., Martinez, A., and Vianello, M. Comparing leja and krylov approximations of large scale matrix exponentials. In *International Conference on Computational Science* (2006), Springer, pp. 685–692. (Cited on page 45.)

[17] Bergmann, M., Bruneau, C.-H., and Iollo, A. Enablers for robust pod models. *Journal of Computational Physics 228*, 2 (2009), 516–538. (Cited on pages 12 and 67.)

[18] Bergmann, M., Cordier, L., and Brancher, J.-P. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. *Physics of fluids 17*, 9 (2005), 097101. (Cited on pages 12 and 67.)

[19] Berkooz, G., Holmes, P., and Lumley, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics 25*, 1 (1993), 539–575. (Cited on page 58.)

[20] Bewley, T., Moin, P., and Temam, R. DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *J. Fluid Mech. 447* (2001), 179–225. (Cited on page 22.)

[21] Binev, P., Cohen, A., Mula, O., and Nichols, J. Greedy algorithms for optimal measurements selection in state estimation using reduced models. *SIAM/ASA Journal on Uncertainty Quantification 6*, 3 (2018), 1101–1126. (Cited on pages 11, 13, 56 and 67.)

[22] Björkman, M., and Holmström, K. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering 1*, 4 (2000), 373–397. (Cited on page 10.)

[23] Blanchard, M., Schuller, T., Sipp, D., and Schmid, P. Response analysis of a laminar premixed M-flame to flow perturbations using a linearized compressible Navier-Stokes solver. *Phys. Fluids 27*, 4 (2015), 043602. (Cited on pages 4, 17 and 23.)

[24] Boninsegna, L., Nüske, F., and Clementi, C. Sparse learning of stochastic dynamical equations. *The Journal of chemical physics 148*, 24 (2018), 241723. (Cited on page 86.)

[25] Braman, K., Oliver, T., and Raman, V. Adjoint-based sensitivity analysis of flames. *Comb. Theor. Modelling 19*, 1 (2015), 29–56. (Cited on pages 4 and 17.)

[26] Bramer, M. *Clustering*. Springer, 2007. (Cited on page 88.)

[27] Brunton, S. L., and Kutz, J. N. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022. (Cited on pages vii, x, 7, 8, 88 and 91.)

[28] Brunton, S. L., Noack, B. R., and Koumoutsakos, P. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics 52* (2020), 477–508. (Cited on page 88.)

[29] Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences 113*, 15 (2016), 3932–3937. (Cited on pages 13 and 14.)

[30] Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS 113*, 15 (2016), 3932–3937. (Cited on pages 83 and 84.)

[31] Brunton, S. L., Proctor, J. L., and Kutz, J. N. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine 49*, 18 (2016), 710–715. (Cited on pages 14 and 86.)

[32] Bui-Thanh, T., Damodaran, M., and Willcox, K. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA journal 42*, 8 (2004), 1505–1516. (Cited on page 56.)

[33] Caliari, M. Accurate evaluation of divided differences for polynomial interpolation of exponential propagators. *Computing 80*, 2 (2007), 189–201. (Cited on page 45.)

[34] Caliari, M., Vianello, M., and Bergamaschi, L. The lem exponential integrator for advection–diffusion–reaction equations. *Journal of computational and applied mathematics 210*, 1-2 (2007), 56–63. (Cited on page 45.)

[35] Callaham, J. L., Koch, J. V., Brunton, B. W., Kutz, J. N., and Brunton, S. L. Learning dominant physical processes with data-driven balance models. *Nature communications 12*, 1 (2021), 1–10. (Cited on pages 14, 84 and 86.)

[36] Cantwell, B. J. Organized motion in turbulent flow. *Annual review of fluid mechanics 13*, 1 (1981), 457–515. (Cited on page 58.)

[37] Cao, Y., Li, S., Petzold, L., and Serban, R. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM J. Sci. Comput. 24*, 3 (2003), 1076–1089. (Cited on page 19.)

[38] Capecelatro, J., Bodony, D., and Freund, J. Adjoint-based sensitivity analysis of ignition in a turbulent reactive shear layer. *AIAA Sci. Tech. Forum* (2017). (Cited on pages 4 and 17.)

[39] Carlberg, K., Barone, M., and Antil, H. Galerkin v. least-squares petrov–galerkin projection in nonlinear model reduction. *Journal of Computational Physics 330* (2017), 693–734. (Cited on pages 67 and 81.)

[40] Carlberg, K., Bou-Mosleh, C., and Farhat, C. 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 (2011), 155–181. (Cited on pages 67, 68, 69 and 70.)

[41] Carlberg, K., Farhat, C., Cortial, J., and Amsallem, D. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *J Comp. Phys. 242* (2013), 623–647. (Cited on pages 12, 13, 56, 67, 68, 70, 81 and 108.)

[42] Cattafesta III, L. N., and Sheplak, M. Actuators for active flow control. *Annual Review of Fluid Mechanics 43* (2011), 247–272. (Cited on page 3.)

[43] Champion, K., Lusch, B., Kutz, J. N., and Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences 116*, 45 (2019), 22445–22451. (Cited on page 14.)

[44] Chang, W., Giraldo, F., and Perot, B. Analysis of an exact fractional step method. *Journal of Computational Physics 180*, 1 (2002), 183–199. (Cited on page 25.)

[45] Chaturantabut, S., and Sorensen, D. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM J Sci. Comput. 32*, 5 (2010), 2737–2764. (Cited on pages 11, 13, 56 and 67.)

[46] Chinesta, F., Huerta, A., Rozza, G., and Willcox, K. Model order reduction. *Encyclopedia of Computational Mechanics* (2016). (Cited on page 57.)

[47] Chorin, A. J. Numerical solution of the navier-stokes equations. *Mathematics of computation 22*, 104 (1968), 745–762. (Cited on pages 25 and 46.)

[48] Conn, A. R., and Toint, P. L. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In *Nonlinear optimization and applications*. Springer, 1996, pp. 27–47. (Cited on page 10.)

[49] Cossu, C. An Introduction to Optimal Control: Lecture notes from the FLOW-NORDITA Summer School on Advanced Instability Methods for Complex Flows. *Appl. Mech. Rev. 66*, 2 (2014), 021001. (Cited on page 18.)

[50] Daniels, B. C., and Nemenman, I. Automated adaptive inference of phenomenological dynamical models. *Nature communications 6*, 1 (2015), 1–8. (Cited on page 14.)

[51] Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment 2005*, 09 (2005), P09008. (Cited on page 91.)

[52] de Pando, M. F. IBMOS: Immersed boundary method optimization and stability, 2020. (Cited on pages 18, 23 and 107.)

[53] de Pando, M. F., Sipp, D., and Schmid, P. Efficient evaluation of the direct and adjoint linearized dynamics from compressible flow solvers. *J. Comp. Phys. 231*, 23 (2012), 7739–7755. (Cited on page 23.)

[54] Deane, A., Kevrekidis, I., Karniadakis, G. E., and Orszag, S. Low-dimensional models for complex geometry flows: application to grooved channels and circular cylinders. *Physics of Fluids A: Fluid Dynamics 3*, 10 (1991), 2337–2354. (Cited on page 60.)

[55] Dede, L. Optimal flow control for navier–stokes equations: drag minimization. *International Journal for Numerical Methods in Fluids 55*, 4 (2007), 347–366. (Cited on page 2.)

[56] Denton, J. D. *Loss mechanisms in turbomachines*, vol. 78897. American Society of Mechanical Engineers, 1993. (Cited on page 27.)

[57] Docquier, N., and Candel, S. Combustion control and sensors: a review. *Progress in energy and combustion science 28*, 2 (2002), 107–150. (Cited on page 1.)

[58] Dorogovtsev, S. *Complex networks*. Oxford University Press, Oxford, UK, 2010. (Cited on page 90.)

[59] Dorogovtsev, S. N., and Mendes, J. F. Evolution of networks. *Advances in physics 51*, 4 (2002), 1079–1187. (Cited on page 90.)

[60] Duraisamy, K., and Alonso, J. Adjoint-based techniques for uncertainty quantification in turbulent flows with combustion. *42nd AIAA Fluid Dynamics Conference and Exhibit* (2012), 25–28. (Cited on pages 4 and 17.)

[61] Everson, R., and Sirovich, L. Karhunen–loeve procedure for gappy data. *JOSA A 12*, 8 (1995), 1657–1664. (Cited on pages 67, 68 and 69.)

[62] Fabre, D., Auguste, F., and Magnaudet, J. Bifurcations and symmetry breaking in the wake of axisymmetric bodies. *Physics of Fluids 20*, 5 (2008), 051702. (Cited on page 96.)

[63] Fan, D., Yang, L., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. Reinforcement learning for bluff body active flow control in experiments and simulations. *Proceedings of the National Academy of Sciences 117*, 42 (2020), 26091–26098. (Cited on page 3.)

[64] Fang, F., Pain, C. C., Navon, I., Elsheikh, A. H., Du, J., and Xiao, D. Non-linear petrov–galerkin methods for reduced order hyperbolic equations and discontinuous finite element methods. *Journal of Computational Physics 234* (2013), 540–559. (Cited on page 12.)

[65] Farhat, C., Cortial, J., Dastillung, C., and Bavestrello, H. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. *Int. J. Numer. Meth. Engng. 67* (2006), 697–724. (Cited on page 38.)

[66] Fasel, U., Kaiser, E., Kutz, J. N., Brunton, B. W., and Brunton, S. L. Sindy with control: A tutorial. *arXiv preprint arXiv:2108.13404* (2021). (Cited on pages 14 and 86.)

[67] Fikl, A., V. Le Chenadec, and Sayadi, T. Control and optimization of interfacial flows using adjoint-based techniques. *Fluids 5* (2020), 3. (Cited on pages 4 and 17.)

[68] Fish, F., and Lauder, G. V. Passive and active flow control by swimming fishes and mammals. *Annu. Rev. Fluid Mech. 38* (2006), 193–224. (Cited on page 3.)

[69] Foures, D., Caulfield, C., and Schmid, P. Optimal mixing in two-dimensional plane Poiseuille flow at finite Peclet number. *J. Fluid Mech. 748* (2014), 241–277. (Cited on pages 4 and 17.)

[70] Freeman, L. C. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41. (Cited on page 92.)

[71] Gad-el Hak, M. Flow control and the energy crisis. *International Journal of Flow Control* (2009). (Cited on page 1.)

[72] Galletti, B., Bruneau, C., Zannetti, L., and Iollo, A. Low-order modelling of laminar flow regimes past a confined square cylinder. *Journal of Fluid Mechanics 503* (2004), 161–170. (Cited on page 12.)

[73] Gander, M., and Güttel, S. A parallel integrator for linear initial-value problems. *SIAM J Scientific Computing 35* (2013), 123–142. (Cited on pages viii, 6, 36, 38 and 39.)

[74] Gander, M., Güttel, S., and Petcu, M. A nonlinear ParaExp algorithm. *Int. Conference on Domain Decomposition Methods, Domain decomposition methods in science and engineering XXIV* (2017), 161–270. (Cited on pages 6 and 38.)

[75] Gander, M. J. 50 years of time parallel time integration. In *Multiple shooting and time domain decomposition methods*. Springer, 2015, pp. 69–113. (Cited on pages vii, 6 and 36.)

[76] Ghaemi, S. Passive and active control of turbulent flows, 2020. (Cited on page 3.)

[77] Giannakis, D., and Majda, A. J. Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences 109*, 7 (2012), 2222–2227. (Cited on page 14.)

[78] Giles, M. B., and Pierce, N. A. An introduction to the adjoint approach to design. *Flow, turbulence and combustion 65*, 3 (2000), 393–415. (Cited on page 23.)

[79] Girvan, M., and Newman, M. E. Community structure in social and biological networks. *Proceedings of the national academy of sciences 99*, 12 (2002), 7821–7826. (Cited on pages 92 and 95.)

[80] Golub, G. H., and Van Loan, C. F. *Matrix computations*. JHU press, 2013. (Cited on page 58.)

[81] González-García, R., Rico-Martìnez, R., and Kevrekidis, I. G. Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering 22* (1998), S965–S968. (Cited on page 14.)

[82] Griewank, A., and Walther, A. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS) 26*, 1 (2000), 19–45. (Cited on page 35.)

[83] Gunzburger, M. D. *Perspectives in flow control and optimization*. SIAM, 2003. (Cited on page 18.)

[84] Gutmann, H.-M. A radial basis function method for global optimization. *Journal of global optimization 19*, 3 (2001), 201–227. (Cited on page 10.)

[85] Hansen, N., Müller, S. D., and Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation 11*, 1 (2003), 1–18. (Cited on page 9.)

[86] Hassan, A., Sayadi, T., LeChenadec, V., and Attili, A. Sensitivity analysis of an unsteady char particle combustion. *Fuel 287* (2021). (Cited on pages 4 and 17.)

[87] Hassan, A., Sayadi, T., LeChenadec, V., Pitsch, H., and Attili, A. Adjoint-based sensitivity analysis of steady char burnout. *Comb. Theor. Modeling* (2020). (Cited on pages 4 and 17.)

[88] Hervé, A., Sipp, D., Schmid, P., and Samuelides, M. A physics-based approach to flow control using system identification. *J Fluid Mech. 702* (2017), 26–58. (Cited on page 84.)

[89] Hinton, G. E., and Roweis, S. Stochastic neighbor embedding. *Advances in neural information processing systems 15* (2002). (Cited on page 9.)

[90] Hochbruck, M., and Lubich, C. On krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis 34*, 5 (1997), 1911–1925. (Cited on page 45.)

[91] Hochbruck, M., and Ostermann, A. Exponential integrators. *Acta Numerica 19* (2010), 209–286. (Cited on page 43.)

[92] Holmes, P., Lumley, J. L., and Berkooz, G. Turbulence, coherent structures, dynamical systems and symmetry. *In Cambridge Monographs on Mechanics. Cambridge Univ. Press* (1996). (Cited on pages 11, 12, 55 and 67.)

[93] Holmes, P., Lumley, J. L., Berkooz, G., and Rowley, C. W. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012. (Cited on page 67.)

[94] Huang, S., and Kim, J. Control and system identification of a separated flow. *J Fluid Mech. 20*, 10 (2008), 101509. (Cited on page 84.)

[95] Hussain, A. K. M. F., and Reynolds, W. C. The mechanics of an organized wave in turbulent shear flow. *Journal of Fluid Mechanics 41*, 2 (1970), 241–258. (Cited on page 29.)

[96] Iollo, A., Dervieux, A., Désidéri, J.-A., and Lanteri, S. Two stable pod-based approximations to the navier–stokes equations. *Computing and visualization in science 3*, 1 (2000), 61–66. (Cited on pages 12 and 67.)

[97] Iollo, A., Lanteri, S., and Désidéri, J.-A. Stability properties of pod–galerkin approximations for the compressible navier–stokes equations. *Theoretical and Computational Fluid Dynamics 13*, 6 (2000), 377–396. (Cited on pages 12 and 67.)

[98] Ito, K., and Ravindran, S. A reduced-order method for simulation and control of fluid flows. *J Comp. Phys. 143*, 2 (1998), 403–425. (Cited on page 56.)

[99] Jacobi, I., and McKeon, B. Dynamic roughness perturbation of a turbulent boundary layer. *Journal of Fluid Mechanics 688* (2011), 258–296. (Cited on page 29.)

[100] Jameson, A. Aerodynamic design via control theory. *J Sci. Comp. 3*, 3 (1988), 233–260. (Cited on pages 4 and 17.)

[101] Jameson, A. Optimum aerodynamic design using cfd and control theory. In *12th computational fluid dynamics conference* (1995), p. 1729. (Cited on pages 2 and 17.)

[102] Jameson, A. Aerodynamic Shape Optimization Using the Adjoint Method. *Lectures at the Von Karman Institute* (2003). (Cited on page 22.)

[103] Jameson, A., Martinelli, L., and Pierce, N. Optimum aerodynamic design using the Navier-Stokes equations. *Theor. Comp. Fluid Dyn. 10*, 1 (1998), 213–237. (Cited on pages 2, 4 and 17.)

[104] Jones, D. A taxonomy of global optimization methods based on response surfaces. *J Global Optim 21* (2001), 345–383. (Cited on page 10.)

[105] Juniper, M. Triggering in the horizontal Rijke tube: non-normality, transient growth and bypass transition. *J. Fluid Mech. 667* (2010), 272–308. (Cited on pages 2, 4 and 17.)

[106] Kaheman, K., Kutz, J. N., and Brunton, S. L. Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A 476*, 2242 (2020), 20200279. (Cited on page 86.)

[107] Kaiser, E., Kutz, J. N., and Brunton, S. L. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A 474*, 2219 (2018), 20180335. (Cited on pages 14 and 86.)

[108] Kalashnikova, I., and Barone, M. F. On the stability and convergence of a galerkin reduced order model (rom) of compressible flow with solid wall and far-field boundary treatment. *International journal for numerical methods in engineering 83*, 10 (2010), 1345–1375. (Cited on page 12.)

[109] Karagozian, A. R. The jet in crossflow. *Physics of Fluids 26*, 10 (2014), 1–47. (Cited on pages vii, 1 and 3.)

[110] Kevrekidis, I. G., Gear, C. W., Hyman, J. M., Kevrekidis, P. G., Runborg, O., Theodoropoulos, C., et al. Equation-free, coarse-grained

multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Sci 1*, 4 (2003), 715–762. (Cited on page 14.)

[111] King, R., Seibold, M., Lehmann, O., Noack, B., Morzyński, M., Tadmor, G., et al. Nonlinear flow control based on a low dimensional model of fluid flow. In *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*. Springer, 2005, pp. 369–386. (Cited on page 2.)

[112] Kochenderfer, M. J., and Wheeler, T. A. *Algorithms for optimization*. Mit Press, 2019. (Cited on pages 4, 8 and 9.)

[113] Kooij, G. L., Botchev, M. A., and Geurts, B. J. A block krylov subspace implementation of the time-parallel paraexp method and its extension for nonlinear partial differential equations. *Journal of computational and applied mathematics 316* (2017), 229–246. (Cited on pages 6 and 39.)

[114] Kooij, G. L., Botchev, M. A., and Geurts, B. J. An exponential time integrator for the incompressible navier–stokes equation. *SIAM journal on scientific computing 40*, 3 (2018), B684–B705. (Cited on pages 7 and 45.)

[115] Koumoutsakos, P., and Mezic, I. *Control of fluid flow*, vol. 330. Springer Science & Business Media, 2006. (Cited on page 9.)

[116] Larson, J., Menickelly, M., and Wild, S. M. Derivative-free optimization methods. *Acta Numerica 28* (2019), 287–404. (Cited on page 8.)

[117] Lassila, T., Manzoni, A., Quarteroni, A., and Rozza, G. Model order reduction in fluid dynamics: challenges and perspectives. *Reduced Order Methods for modeling and computational reduction* (2014), 235–273. (Cited on page 2.)

[118] Leicht, E. A., and Newman, M. E. Community structure in directed networks. *Physical review letters 100*, 11 (2008), 118703. (Cited on pages 93 and 95.)

[119] Lemke, M., Cai, L., Reiss, J., Pitsch, H., and Sesterhenn, J. Adjoint-based sensitivity analysis of quantities of interest of complex combustion models. *Comb. Theo. Modelling 23*, 1 (2019), 180–196. (Cited on pages 4 and 17.)

[120] Lemke, M., Reiss, J., and Sesterhenn, J. Adjoint-based analysis of thermoacoustic coupling. *ICNAAM* (2013), 2163–2166. (Cited on pages 4 and 17.)

[121] Loiseau, J.-C., Brunton, S., and Noack, B. From the pod-galerkin method to sparse manifold models, 2021. (Cited on pages 11 and 86.)

[122] Loiseau, J.-C., and Brunton, S. L. Constrained sparse galerkin regression. *Journal of Fluid Mechanics 838* (2018), 42–67. (Cited on pages 11, 84 and 86.)

[123] Loiseau, J.-C., Noack, B. R., and Brunton, S. L. Sparse reduced-order modelling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics 844* (2018), 459–490. (Cited on pages 11, 13, 84 and 86.)

[124] Lumley, J. L. Coherent structures in turbulence. In *Transition and turbulence*. Elsevier, 1981, pp. 215–242. (Cited on page 58.)

[125] Ma, X., and Karniadakis, G. E. A low-dimensional model for simulating three-dimensional cylinder flow. *Journal of Fluid Mechanics 458* (2002), 181–190. (Cited on page 60.)

[126] Maday, Y. Reduced basis method for the rapid and reliable solution of partial differential equations. (Cited on pages 11 and 56.)

[127] Maday, Y., Anthony, T., Penn, J. D., and Yano, M. Pbdw state estimation: Noisy observations; configuration-adaptive background spaces; physical interpretations. *ESAIM: Proceedings and Surveys 50* (2015), 144–168. (Cited on page 56.)

[128] Maday, Y., and Mula, O. A generalized empirical interpolation method: application of reduced basis techniques to data assimilation. In *Analysis and numerics of partial differential equations*. Springer, 2013, pp. 221–235. (Cited on pages 11, 13, 56 and 67.)

[129] Maday, Y., Mula, O., Patera, A. T., and Yano, M. The generalized empirical interpolation method: stability theory on hilbert spaces with an application to the stokes equation. *Computer Methods in Applied Mechanics and Engineering 287* (2015), 310–334. (Cited on pages 11, 13, 56 and 67.)

[130] Maday, Y., Patera, A. T., and Turinici, G. A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. *Journal of Scientific Computing 17*, 1 (2002), 437–446. (Cited on pages 11 and 56.)

[131] Maday, Y., Riahi, M.-K., and Salomon, J. Parareal in time intermediate targets methods for optimal control problems. In *Control and optimization with PDE constraints*. Springer, 2013, pp. 79–92. (Cited on page 36.)

[132] Majda, A. J., Franzke, C., and Crommelin, D. Normal forms for reduced stochastic climate models. *Proceedings of the National Academy of Sciences 106*, 10 (2009), 3649–3653. (Cited on page 14.)

[133] March, A., Willcox, A., and Wang, Q. Gradient-based multifidelity optimisation for aircraft design using bayesian model calibration. *Aeronautical J 115*, 1174 (2011), 729–738. (Cited on page 10.)

[134] Marion, M., and Temam, R. Nonlinear galerkin methods. *SIAM Journal on numerical analysis 26*, 5 (1989), 1139–1157. (Cited on page 12.)

[135] Marsden, A. L., Feinstein, J. A., and Taylor, C. A. A computational framework for derivative-free optimization of cardiovascular geometries. *Computer Methods in Applied Mechanics and Engineering 197*, 21-24 (2008), 1890–1905. (Cited on page 9.)

[136] Marsden, A. L., Wang, M., Dennis, J., and Moin, P. Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation. *Journal of Fluid Mechanics 572* (2007), 13–36. (Cited on page 9.)

[137] Marsden, A. L., Wang, M., Dennis Jr, J. E., and Moin, P. Suppression of vortex-shedding noise via derivative-free shape optimization. *Physics of fluids 16*, 10 (2004), L83–L86. (Cited on page 9.)

[138] McKeon, B. A model for 'dynamic'roughness in turbulent channel flow. In *Proceedings of the Summer Program* (2008), pp. 399–410. (Cited on page 29.)

[139] Meliga, P., Chomaz, J.-M., and Sipp, D. Global mode interaction and pattern selection in the wake of a disk: a weakly nonlinear expansion. *Journal of Fluid Mechanics 633* (2009), 159–189. (Cited on page 96.)

[140] Minchev, B. V., and Wright, W. A review of exponential integrators for first order semi-linear problems. (Cited on page 43.)

[141] Mohammadi, B., and Pironneau, O. Shape optimization in fluid mechanics. *Annu. Rev. Fluid Mech. 36* (2004), 255–279. (Cited on page 2.)

[142] Murray, N. E., and Ukeiley, L. S. An application of gappy pod. *Experiments in Fluids 42*, 1 (2007), 79–91. (Cited on page 56.)

[143] Nemili, A., Özkaya, E., Gauger, N. R., Kramer, F., and Thiele., F. Discrete adjoint based optimal active control of separation on a realistic high-lift configuration. *New Results in Numerical and Experimental Fluid Mechanics X: Contributions to the 19th STAB/DGLR Symposium Munich, Germany, 2014. Springer International Publishing* (2016), 237–246. (Cited on page 4.)

[144] Newman, M. *Networks*. Oxford university press, 2018. (Cited on page 90.)

[145] Newman, M. E. The structure and function of complex networks. *SIAM review 45*, 2 (2003), 167–256. (Cited on pages 89 and 90.)

[146] Newman, M. E. Fast algorithm for detecting community structure in networks. *Physical review E 69*, 6 (2004), 066133. (Cited on pages 92 and 95.)

[147] Newman, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences 103*, 23 (2006), 8577–8582. (Cited on pages 92, 94 and 95.)

[148] Newman, M. E., and Girvan, M. Finding and evaluating community structure in networks. *Physical review E 69*, 2 (2004), 026113. (Cited on page 92.)

[149] Nguyen, N. C., Patera, A. T., and Peraire, J. A 'best points' interpolation method for efficient approximation of parametrized functions. *International journal for numerical methods in engineering 73*, 4 (2008), 521–543. (Cited on pages 13 and 67.)

[150] Nguyen, N. C., and Peraire, J. An efficient reduced-order modeling approach for non-linear parametrized partial differential equations. *International Journal for Numerical Methods in Engineering 76*, 1 (2008), 27–55. (Cited on pages 13 and 67.)

[151] Nievergelt, J. Parallel methods for integrating ordinary differential equations. *Communications of the ACM 7*, 12 (1964), 731–733. (Cited on page 36.)

[152] Noack, B. R., Afanasiev, K., MORZYŃSKI, M., Tadmor, G., and Thiele, F. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics 497* (2003), 335–363. (Cited on page 60.)

[153] Noack, B. R., and Eckelmann, H. A low-dimensional galerkin method for the three-dimensional flow around a circular cylinder. *Physics of Fluids 6*, 1 (1994), 124–143. (Cited on page 60.)

[154] Noack, B. R., Morzynski, M., and Tadmor, G. *Reduced-order modelling for flow control*, vol. 528. Springer Science & Business Media, 2011. (Cited on page 67.)

[155] Noack, B. R., Papas, P., and Monkewitz, P. A. The need for a pressure-term representation in empirical galerkin models of incompressible shear flows. *Journal of Fluid Mechanics 523* (2005), 339–365. (Cited on pages 12 and 67.)

[156] Ohlberger, M., and Rave, S. Reduced basis methods: Success, limitations and future challenges. *Proceedings of the Conference Algoritmy* (2016), 1–12. (Cited on page 57.)

[157] Ong, B. W., and Schroder, J. B. Applications of time parallelization. *Computing and Visualization in Science 23*, 1 (2020), 1–15. (Cited on page 36.)

[158] Pan, W., Yuan, Y., Gonçalves, J., and Stan, G.-B. A sparse bayesian approach to the identification of nonlinear state-space systems. *IEEE Transactions on Automatic Control 61*, 1 (2015), 182–187. (Cited on page 86.)

[159] Peherstorfer, B., Drmac, Z., and Gugercin, S. Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points. *SIAM Journal on Scientific Computing 42*, 5 (2020), A2837–A2864. (Cited on pages 11, 13, 56 and 67.)

[160] Perot, J. B. An analysis of the fractional step method. *Journal of Computational Physics 108*, 1 (1993), 51–58. (Cited on page 25.)

[161] Pierret, S., Coelho, R. F., and Kato, H. Multidisciplinary and multiple operating points shape optimization of three-dimensional compressor blades. *Structural and Multidisciplinary Optimization 33*, 1 (2007), 61–70. (Cited on page 9.)

[162] Pironneau, O. On optimal design in fluid mechanics. *J. Fluid Mech. 64*, 1 (1974), 97–110. (Cited on pages 4 and 17.)

[163] Podvin, B., and Lumley, J. A low-dimensional approach for the minimal flow unit. *Journal of Fluid Mechanics 362* (1998), 121–155. (Cited on pages 12 and 67.)

[164] Pothen, A., Simon, H. D., and Liou, K.-P. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications 11*, 3 (1990), 430–452. (Cited on page 92.)

[165] Powell, M. On the use of quadratic models in unconstrained minimization without derivatives. *Optimization Methods and Software 19*, 3-4 (2004), 399–411. (Cited on page 10.)

[166] Powell, M. J. Uobyqa: unconstrained optimization by quadratic approximation. *Mathematical Programming 92*, 3 (2002), 555–582. (Cited on page 10.)

[167] Quirós Rodríguez, A., Fosas de Pando, M., and Sayadi, T. Gradient-enhanced stochastic optimization of high-fidelity simulations. Available at SSRN: https://ssrn.com/abstract=4035229 or http://dx.doi.org/10.2139/ssrn.4035229. (Cited on page 109.)

[168] Rabin, S., Caulfield, C., and Kerswell, R. Designing a more nonlinearly stable laminar flow via boundary manipulation. *J. Fluid Mech. 738 R1* (2014), 1–12. (Cited on pages 4 and 17.)

[169] Reddy, C. K., and Vinzamuri, B. A survey of partitional and hierarchical clustering algorithms. *Data clustering: Algorithms and applications 87* (2013). (Cited on page 88.)

[170] Regis, R. G., and Shoemaker, C. A. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing 19*, 4 (2007), 497–509. (Cited on page 10.)

[171] Reichel, L. Newton interpolation at leja points. *BIT 30*, 2 (1990), 332–346. (Cited on page 45.)

[172] Rempfer, D. On low-dimensional galerkin models for fluid flow. *Theoretical and Computational Fluid Dynamics 14*, 2 (2000), 75–88. (Cited on page 12.)

[173] Reuther, J., Jameson, A., Alonso, J., Rimlinger, M., and Saunders, D. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers: Part 2. *J. Aircraft 36*, 1 (1999), 61–74. (Cited on pages 4 and 17.)

[174] Rigas, G., Morgans, A. S., Brackston, R., and Morrison, J. F. Diffusive dynamics and stochastic models of turbulent axisymmetric wakes. *Journal of Fluid Mechanics 778* (2015). (Cited on pages 95, 96 and 97.)

[175] Rigas, G., Oxlade, A., Morgans, A., and Morrison, J. Low-dimensional dynamics of a turbulent axisymmetric wake. *Journal of Fluid Mechanics 755* (2014). (Cited on pages x, 95 and 97.)

[176] Rios, L. M., and Sahinidis, N. V. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization 56*, 3 (2013), 1247–1293. (Cited on page 8.)

[177] Roberts, A. J. *Model emergent dynamics in complex systems*, vol. 20. SIAM, 2014. (Cited on page 14.)

[178] Roma, A. M., Peskin, C. S., and Berger, M. J. An adaptive version of the immersed boundary method. *Journal of computational physics 153*, 2 (1999), 509–534. (Cited on page 26.)

[179] Rowley, C. W., Colonius, T., and Murray, R. M. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena 189*, 1-2 (2004), 115–129. (Cited on page 12.)

[180] Rudy, S., Alla, A., Brunton, S. L., and Kutz, J. N. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems 18*, 2 (2019), 643–660. (Cited on pages 14 and 86.)

[181] Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. Data-driven discovery of partial differential equations. *Science advances 3*, 4 (2017), e1602614. (Cited on pages 14, 86 and 100.)

[182] Saad, Y. Analysis of some krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis 29*, 1 (1992), 209–228. (Cited on pages 44 and 45.)

[183] Saad, Y. *Iterative methods for sparse linear systems*. SIAM, 2003. (Cited on page 44.)

[184] Safiran, N., Lotz, J., and Naumann, U. Algorithmic Differentiation of Numerical Methods: Second- order Adjoint Solvers for Parameterized Systems of Nonlinear Equations. *Procedia Computer Science 80* (2016), 2231–2235. (Cited on page 23.)

[185] Schmid, P. Dynamic mode decomposition of numerical and experimental data. *J. Fluid. Mech. 656* (2010), 5–28. (Cited on pages 11 and 55.)

[186] Schmidt, S., Ilic, C., Schulz, V., and Gauger, N. R. Three-dimensional large-scale aerodynamic shape optimization based on shape calculus. *AIAA J. 51*, 11 (2013), 2615–2627. (Cited on pages 4 and 17.)

[187] Schuller, T., T, P., and Candel, S. Dynamics and control of premixed combustion systems based on flame transfer and describing functions. *J Fluid Mech. 894* (2020). (Cited on page 4.)

[188] Shen, J. Long time stability and convergence for fully discrete nonlinear galerkin methods. *Applicable Analysis 38*, 4 (1990), 201–229. (Cited on page 12.)

[189] Simpson, T., Peplinksi, J., Koch, P., and Allen, J. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. with Computers 17* (2001), 129–150. (Cited on page 10.)

[190] Sirisup, S., and Karniadakis, G. E. A spectral viscosity method for correcting the long-term behavior of pod models. *Journal of Computational Physics 194*, 1 (2004), 92–116. (Cited on pages 12 and 67.)

[191] Sirkes, Z., and Tziperman, E. Finite difference of adjoint or adjoint of finite difference? *Monthly weather review 125*, 12 (1997), 3373–3378. (Cited on page 22.)

[192] Sirovich, L. Turbulence and the dynamics of coherent structures. *Q. Appl. Math. 45* (1987), 561–590. (Cited on pages 11, 55 and 58.)

[193] Skene, C. S., Eggl, M. F., and Schmid, P. J. A parallel-in-time approach for accelerating direct-adjoint studies. *Journal of Computational Physics 429* (2021), 110033. (Cited on pages vii, 5, 6, 15, 36, 38, 41, 54 and 107.)

[194] Taira, K., Brunton, S. L., Dawson, S. T., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V., and Ukeiley, L. S. Modal analysis of fluid flows: An overview. *Aiaa Journal 55*, 12 (2017), 4013–4041. (Cited on page 11.)

[195] Taira, K., and Colonius, T. The immersed boundary method: A projection approach. *J. Comp. Phys. 225* (2007), 2118–2137. (Cited on pages 23 and 25.)

[196] Taira, K., Hemati, M. S., Brunton, S. L., Sun, Y., Duraisamy, K., Bagheri, S., Dawson, S. T., and Yeh, C.-A. Modal analysis of fluid flows: Applications and outlook. *AIAA journal 58*, 3 (2020), 998–1022. (Cited on pages vii, 11 and 12.)

[197] Tang, K., Graham, W., and Peraire, J. Active flow control using a reduced order model and optimum control. In *Fluid Dynamics Conference* (1996), p. 1946. (Cited on page 2.)

[198] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288. (Cited on page 85.)

[199] Volkwein, S. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. 1025* (2011). (Cited on page 58.)

[200] Voss, H. U., Kolodner, P., Abel, M., and Kurths, J. Amplitude equations from spatiotemporal binary-fluid convection data. *Physical review letters 83*, 17 (1999), 3422. (Cited on page 14.)

[201] Wang, Q., Hu, R., and Blonigan, P. Least Squares Shadowing sensitivity analysis of chaotic limit cycle oscillations. *J. Comp. Phys. 267* (2014), 210–224. (Cited on page 5.)

[202] Wang, Q., Moin, P., and Iaccarino, G. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2549–2567. (Cited on page 22.)

[203] Wang, Z., Akhtar, I., Borggaard, J., and Iliescu, T. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering 237* (2012), 10–26. (Cited on page 12.)

[204] Webster, J. G., and Ljung, L. system identification. *In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (Ed.)* (2017). (Cited on page 84.)

[205] Wei, M., and Freund, J. A noise-controlled free shear flow. *J. Fluid Mech. 546* (2005), 123. (Cited on page 22.)

[206] Wieder, N. Cybernetics or control and communication in the animal and the machine, 1st ed. *MIT Press* (1948). (Cited on page 84.)

[207] Willcox, K. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & fluids 35*, 2 (2006), 208–226. (Cited on page 56.)

[208] Xiao, D., Fang, F., Du, J., Pain, C., Navon, I., Buchan, A., Elsheikh, A. H., and Hu, G. Non-linear petrov–galerkin methods for reduced order modelling of the navier–stokes equations using a mixed finite element pair. *Computer Methods In Applied Mechanics and Engineering 255* (2013), 147–157. (Cited on page 12.)

[209] Ye, H., Beamish, R. J., Glaser, S. M., Grant, S. C., Hsieh, C.-h., Richards, L. J., Schnute, J. T., and Sugihara, G. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences 112*, 13 (2015), E1569–E1576. (Cited on page 14.)

[210] Yeh, C.-A., and Taira, K. Resolvent-analysis-based design of airfoil separation control. *Journal of Fluid Mechanics 867* (2019), 572–610. (Cited on page 3.)

[211] Zhang, L., and Schaeffer, H. On the convergence of the sindy algorithm. *Multiscale Modeling & Simulation 17*, 3 (2019), 948–972. (Cited on page 86.)

[212] Zheng, P., Askham, T., Brunton, S. L., Kutz, J. N., and Aravkin, A. Y. A unified framework for sparse relaxed regularized regression: Sr3. *IEEE Access 7* (2018), 1404–1423. (Cited on page 86.)

[213] Zhonghua, H., Chenzhou, X., Zhang, L., Zhang, Y., Zhang, K., and Wenping, S. Efficient aerodynamic shape optimization using variable-fidelity surrogate models and multilevel computational grids. *Chinese Journal of Aeronautics 33*, 1 (2020), 31–47. (Cited on page 10.)