Towards efficient adjoint-based mesh optimization for predictive Large Eddy Simulation

Li, X.

DOI
10.4233/uuid:3ec425ae-2055-42e2-b911-e00ccf035a16

Publication date
2023

Document Version
Final published version

Citation (APA)
Li, X. (2023). *Towards efficient adjoint-based mesh optimization for predictive Large Eddy Simulation*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:3ec425ae-2055-42e2-b911-e00ccf035a16

TOWARDS EFFICIENT ADJOINT-BASED MESH OPTIMIZATION FOR PREDICTIVE LARGE EDDY SIMULATION

TOWARDS EFFICIENT ADJOINT-BASED MESH OPTIMIZATION FOR PREDICTIVE LARGE EDDY SIMULATION

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the board for Doctorates,
to be defended publicly on
Monday 30 October 2023 at 15:00 o'clock

by

Xiaodong LI

Master of Engineering in Fluid Mechanics,
Northwestern Polytechnical University, Xi'an, China,
born in Weifang, China.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Prof. dr. ir. S. Hickel,	Delft University of Technology, promotor
Dr. S.J. Hulshoff,	Delft University of Technology, copromotor
Independent members:	
Prof. dr. K.J. Fidkowski	University of Michigan, USA
Prof. dr. J. Larsson	University of Maryland, USA
Prof. dr. R. Pecnik	Delft University of Technology
Dr. R. Dwight	Delft University of Technology
Dr. M. Möller	Delft University of Technology
Reserve member:	
Prof. dr. ir. C. Vuik	Delft University of Technology

Keywords: Large Eddy Simulation, Adjoint Method, Order Reduction, Incremental Singular Value Decomposition, A Posteriori Error Estimation, Adaptive Mesh Refinement

Front & Back: Cover designed by Dr. Y. Shang, using an image from `https://stablediffusionweb.com`.

ISBN 978-94-6366-753-1
Printed by Rijnja Repro, Delft

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

To improve is to change;
to be perfect is to change often.

Winston Churchill

CONTENTS

SUMMARY

This thesis contributes to the effective and efficient application of unsteady adjoint methods to Adaptive Mesh Refinement (AMR) for Large Eddy Simulation (LES). Three aspects, i.e. subgrid-scale model error, storage cost of high-dimensional data, and stability of the adjoint problem for turbulent flows, were studied to make adjoint-based mesh adaptation feasible for time-dependent high-fidelity simulations.

The effectiveness of adjoint-based error estimation is initially demonstrated using linear advection-diffusion problems. An adjoint-based AMR strategy is further developed and analysed for unsteady 1D Burgers problems with a multi-frequency forcing term. Then we introduce a Reduced-Order Representation (ROR), which uses the Proper Orthogonal Decomposition (POD) to replace full-order primal solutions when solving the adjoint problem backward in time. Numerical results demonstrate the effectiveness of using RORs for adjoint-based AMR.

An enhanced online algorithm for POD analysis is proposed to deal with high-dimensional LES data, resulting from the nonlinearity and unsteadiness that require us to store a time history of primal states for solving the adjoint problem. The enhanced algorithm is based on the incremental Singular Value Decomposition, but exploits the decomposition of full-order solutions into reconstructed and truncated solutions. Two lower-bound estimators are proposed to equip the enhanced algorithm with a posteriori error analyses. Numerical experiments demonstrate that the algorithm can significantly improve the efficiency of online POD analysis while computational accuracy is maintained with an appropriate number for the truncation of POD modes. Furthermore, the enhanced algorithm scales well in parallel and the improvement of computing efficiency is independent of the number of processors.

The unsteady adjoint problem is investigated for 2D and 3D cylinder flows. Using RORs significantly reduces the memory requirement for storing primal flow solutions for both 2D and 3D cylinder flow. Dynamic features of the adjoint field are well presented with using RORs, although there are differences in regions around and upstream of the cylinder using a small number of POD modes. Error distributions can be well predicted with POD-based RORs, especially in regions with large errors. The exponential growth of adjoint solutions in the 3D turbulent flow is found to be attenuated when using RORs for solving the adjoint problem.

NOMENCLATURE

Acronyms

2D	Two-dimensional
3D	Three-dimensional
AMR	Adaptative Mesh Refinement
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
DG	Discontinuous Galerkin
DNS	Direct Numerical Simulation
DoF	Degree of Freedom
FEM	Finete-Element Method
FVM	Finete-Volume Method
iSVD	Incremental Singular Value Decomposition
LES	Large Eddy Simulation
NS	Navier–Stokes
PDEs	Partial Differential Equations
POD	Proper Orthogonal Decomposition
QoI	Quantity of Interest
RANS	Reynolds-Averaged Navier–Stokes
ROM	Reduced-Order Model
ROR	Reduced-Order Representation
SVD	Singular Value Decomposition

Physical symbols

$\boldsymbol{\phi}$	POD modes
$\boldsymbol{u}$	Primal flow velocity vector

$\boldsymbol{u}_a$	Adjoint velocity vector
$\boldsymbol{V}$	Left singular vector
$\boldsymbol{v}_{M+1}$	The $M+1$-th left singular vector
$\boldsymbol{W}$	Right singular vector
Δt	The size of a time step
ΔV	The volume of a finite volume
ϕ	Any fields, i.e. pressure or velocity
θ_{sep}	Separation angles
C_D	Drag coefficient
C_L	Lift coefficient
L_d	Diameter of a circular cylinder
p	Primal pressure
q	Adjoint pressure
T_{vortex}	Time length of one vortex shedding period
u_i	Primal flow velocity
V	The space of a finite volume
v_i	Adjoint velocity

Subscripts/superscripts

$^{n+1}$	Parameters at $(n+1)$-th time
^{n}	Parameters at n-th time
$_{\infty}$	Free stream parameters
$_{\text{P}}$	Variables in resent cell
$_{\text{N}_i}$	Variables in i-th neighbour cell
$_{f_i}$	Variables at i-th cell face

1

Introduction

1

1.1. BACKGROUND AND MOTIVATION

Computational Fluid Dynamics (CFD) supplies an effective pathway to quantify the effect of turbulence for practical problems [1], such as enhancing the performance of transportation capability and lowering the noise level of aircraft [2]. Aerodynamic analyses in the aviation industry usually involve high Reynolds number (high-Re) turbulent flows that contain a vast range of time and length scales. Direct Numerical Simulation (DNS) is a method to directly solve the Navier–Stokes equations on a high-resolution mesh so that the whole range of scales is well resolved and thus the physics of turbulence is accurately captured. However, the computational cost of DNS for three-dimensional problems is too high for practical applications because of the significant increases in computing cost with increasing Re. Different numerical methods are needed to efficiently evaluate flow quantities, model turbulent flows, and capture complex physical phenomena.

The Reynolds-averaged Navier–Stokes (RANS) method is widely used [3] due to its efficiency and effectiveness. The RANS equations describe time-averaged flow characteristics using the Reynolds decomposition, in which Reynolds stress terms are modelled by a turbulence model that makes use of ensemble-averaged quantities. However, RANS methods are particularly unreliable for flows that are highly unsteady or involve mixing layers and large pressure gradients, such as the flow over airfoils at high angles of attack [4], supersonic engine inlets [5], flow separation or reattachment.

An alternative, known as Large Eddy Simulation (LES) [6], is to resolve the large and dynamically most important scales of the flow while representing only the effects of smaller and less significant scales using a model. LES has shown to be highly successful in situations where the computational mesh is fine enough to resolve an appropriate range of large scales. However, there are some prerequisites necessary to make the computation of LES affordable and precise for practical problems.

Firstly, for high-Re turbulent wall-bounded flows, LES needs very high grid resolutions, which is almost as high as that required for DNS within the inner part of boundary layers. Reducing the computing cost as much as possible without sacrificing accuracy is important for the application of LES. For this purpose, users need to generate a high-quality computing mesh, where the mesh size should be small enough to capture relevant flow structures in different physical regions but computationally feasible. However, the quality of manually-generated meshes strongly depends on the users' experience. Although highly experienced CFD engineers are possibly able to generate suitable meshes that ensure the required prediction accuracy at a tractable computational cost, the complexity and multiscale property of turbulent flow problems make it difficult for us to have a priori knowledge on where and how the grid needs to be refined [7]. Often many trial-and-error attempts are required to understand where to refine the grid, which impedes the employment of LES for practical designs. If the flow is heterogeneous and difficult to anticipate, standard meshes can provide inappropriate local levels of refinement [8, 9] and thus lead to either poor solutions or reduced efficiency. Engineering problems also involve complex geometries, making the generation of a high-quality mesh more demanding. While state-of-the-art computing power enables well-resolved highly-accurate LES for a wide range of practical engineering problems [1, 2], the difficulties mentioned often hamper application of LES to design optimization and critical

design decisions.

Adaptive mesh refinement (AMR) or mesh adaptation is the most effective approach to interaction-free high-accuracy simulations. AMR can provide numerical simulations with a high-quality mesh while keeping the computing cost tractable. Through generating the mesh with suitable levels of refinement, the calculation accuracy is improved automatically and the manual effort is avoided, leading to autonomous and reliable simulations [1]. However, the application of AMR in an LES framework is not straightforward. Classical a priori estimates based on truncation error are inappropriate for LES since a significant part of the solution must remain unresolved for efficient LES and this classical AMR method would lead to DNS-like meshes. Viable a priori estimates [10, 11] use physical models or multi-level approaches to detect regions where the unresolved scale model provides poor predictions. However, the final grids based on these a priori estimates are usually sub-optimal regarding computational cost because a priori methods treat all local flow features equally without accounting for their different contributions to a Quantity of Interest (QoI), such as lift and drag of an aircraft wing. This limitation can be overcome using adjoint-based error estimates, which use adjoint solutions of the Navier–Stokes (NS) equations to estimate the local contribution to the error of a QoI. This approach has been applied successfully to sensitivity analysis on shape optimization in steady flow problems [12, 13], the computation of turbulent flows [14], and mesh adaptation in steady and unsteady flows [15, 16].

The adjoint problem in unsteady NS fluid flows contains a time derivative term and needs to be marched backward in time whereas the primal flow problem advances forward in time, a salient difference from the steady case. Due to the nonlinearity of NS equations, the adjoint problem, a linear system, depends on primal flow states at each time step. Therefore, solving the unsteady adjoint problem requires access to all flow states over the time period necessary to reliably calculate statistics of a QoI, such as the mean drag and lift. This storage requirement is intractable for LES, particularly in practical problems, as we have not only a high number of degrees-of-freedom (DoF) for spatial fine-scale resolutions but we also need a small time step to resolve turbulence fine-scale dynamics and a long time period for the accurate estimate of statistics. This leads to intractable storage requirements for primal flow solution data. Developing techniques to overcome these difficulties is crucial for improving the efficiency of solving the adjoint problem and estimating the error of a QoI's approximation. The associated challenges and ways to handle them are discussed in Section 1.3 after the fundamentals and development of adjoint-based mesh adaptation are reviewed in Section 1.2.

1.2. FUNDAMENTALS OF ADJOINT-BASED MESH ADAPTATION

1.2.1. GENERAL MESH ADAPTATION STRATEGIES

Systematic mesh adaptation requires (I) *error estimates* that determine where to refine the mesh and (II) *mesh adaptation strategies* that define how to change the mesh. In the early development of AMR, feature-based methods were applied to provide error estimates that can effectively reduce the discretization error [17, 18]. For instance, Roy [19] examined four types of approaches, i.e., solution gradients, solution curvature, discretization error and truncation error, and found that the adaptation based on trun-

cation errors provided superior results in his study. Other physical-based criteria or heuristic indicators, such as wall distance, vorticity, shock sensors, small-scale kinetic energy [20], and sub-scale error estimators [21], have been applied to drive mesh adaptation. While the sub-domains to be enriched can be recognized, the expert-user knowledge is required to determine which kind of features should be considered for driving the AMR, which indeed limits their efficient utilization. Feature-based error estimates are less efficient for identifying the regions necessary for the accurate computation of a specific QoI. Feature-based adaptive schemes can fail to self-terminate or can produce erroneous values for the predicted QoI at termination [22]. For instance, one study reported that for the ONERA M6 wing at $Ma = 0.84$ with the angle of attack of 3 deg, feature-based adaptation does not focus adaptation near the trailing edge where the grid has a great influence on the drag and lift prediction [23].

Mesh adaptation is categorized into r-adaptation, p-adaptation, and h-adaptation. r-adaptation [24] is a mesh movement method where the positions of mesh points are changed while maintaining the number of mesh cells. In an AMR content, this can induce chaotic behaviours [25]. Alternatively, the approximation order of the numerical scheme can be changed locally via p-adaptation. Finally, the number of mesh cells can be altered, known as h-adaptation (mesh refinement). Isotropic or anisotropic mesh adaptation can be considered for p-adaptation and h-adaptation. Oliver [26] reported that h-adaptation allows the easy generation of anisotropic meshes, especially for boundary layer elements in high-*Re* flows, while p-adaptation is known to be more efficient for sufficiently regular solutions.

The main objective of mesh adaptation is to keep a good balance between computing cost and solution accuracy. The parameter designed for determining where the mesh should be enriched is the *error indicator*, η_n, which is also referred to as *adaptation intensity* [23] or *error intensity* [27]. The error indicator is often defined based on the error estimate ϵ_n of each cell using its absolute value, viz. $\eta_n = |\epsilon_n|$. The cells to be refined are chosen, for example, by using a fixed fraction of the largest error indicators. Such fractions must be chosen to prevent an excessive number of iterations or over-refinement [9]. The most common refinement strategies are given below.

1. Refinement of a prescribed percentage of the mesh (5% [28, 29], 10% [14, 30, 31]) with the largest error indicators.
2. Refinement based on a local parameter defined as $\eta_n > \lambda \frac{e_{tot}}{N}$, with $\lambda = 1$ [22, 32], a decreasing λ starting at $\lambda > 1$ [33], or a scaled λ defined as $\frac{e_{tot}}{\sum_{i=1}^{N} \eta_i}$ [23].
3. Refinement based on a normalized error indicator [27], i.e. $\eta_n > \eta_{lim} = \mu + c_{rel} * \sigma$, where μ and σ denote the mean value and standard derivation of η_n and c_{rel} is a relaxation factor usually larger than 0.5.

1.2.2. ADJOINT-BASED MESH ADAPTATION STRATEGIES

According to the U.S. National Research Council [34], the best practice in verification and validation is to identify simulation outputs (i.e. QoIs) and then assess the accuracy of these outputs. Output-driven mesh adaptation based on the adjoint method follows this strategy by solving a second set of equations for a dual solution that represents the sensitivity of the QoIs. It is also referred to as goal-oriented [35] or adjoint-weighted residual

method [15]. The adjoint problem itself is also referred to as the *dual problem* [36] while the original flow problem is referred to as the *primal problem*.

There are continuous and discrete adjoint methods. Continuous adjoint methods are firstly derived based on the continuous primal equation and then discretized afterwards. In contrast, discrete adjoint methods are formulated from the discretized primal problem. Duraisamy et al. [37] compared the application of discrete and continuous adjoint methods for error estimation and mesh adaptation for supersonic inviscid flows with strong shocks. For their test cases and discretization schemes, they found that the discrete adjoint methods are capable of estimating the relative error between the fine and coarse meshes more accurately whereas continuous adjoint methods appear to be marginally better at estimating the true error in the limit of well-converged flow and adjoint solutions.

The adjoint variables act as a "bridge" between the gradient (or perturbation) of the QoI and the derivative (or perturbation) of the flow equations, both in steady [36, 38] and unsteady problems [39]. The adjoint connects local error sources to the QoI. Thus the mesh regions that have a large influence on the calculation of the QoI can be determined based on the flow residual and associated adjoint solutions.

Adjoint-based error estimation has been widely studied for mesh adaptation in different frameworks, including finite-element methods (FEM) [15], finite-volume methods (FVM) [23, 35, 40] and discontinuous Galerkin (DG) methods [41]. There are several review papers on adjoint-based mesh adaptation for steady linear and nonlinear flows [9, 15, 42, 43]. These consider both discrete and continuous formulations as well as mesh adaptation strategies for laminar and turbulent aerodynamic applications.

The general procedure [14, 44, 45] for adjoint-based mesh adaptation is summarized in Algorithm 1. The user-specified total tolerance (e_{tot}) controls the terminating condition of the mesh adaptation, although other criteria, such as relative tolerance and the number of adaptations, can also be used. η_n^k is the adjoint-based *error indicator* and denotes the local error contribution of the n-th cell in a k-th adaptive mesh, which is used to determine the regions that will be refined [19, 22, 32, 46–49]. The *error indicator* relies on the error estimation from adjoint solutions. Adjoint-based error estimation

Algorithm 1 A general procedure for mesh adaptation

1. Given a prescribed total error tolerance (e_{tot}), start from an initial coarse mesh $\mathscr{T}_h^k$, with $k = 0$.
2. Compute the discrete flow solution $u_H \in V_H$, on the current mesh $\mathscr{T}_H^k$.
3. Compute the approximation of the adjoint solution $v_H \in V_H$ on the same mesh.
4. Interpolate the flow and adjoint solutions onto a fine space.
5. Evaluate the error indicator η_n^k based on the solutions in a fine space, where n denotes the index of mesh cells. If $\sum_n |\eta_n^k| < e_{tot}$, then terminate mesh adaptation.
6. Flag the mesh cells in $\mathscr{T}_H^k$ with a selected criterion, such as a fixed fraction of elements/cells with large error indicators.
7. Enrich the flagged cells to generate a new mesh $\mathscr{T}_H^{k+1}$ and go to step 2.

requires adjoint solutions on a fine space, of which the computation is expensive. Various techniques have been used to deal with this issue so as to make the error estimation computable as follows.

1. Reconstructing high-order solutions based on coarse solutions
   - using an embedded finer mesh [23],
   - using quadratic [33, 50] or cubic [51] polynomials,
   - using least-squares operator penalizing first derivative differences [22] or patch recovery post-processing [45].
2. Solving the adjoint problem on a chosen richer space
   - by increasing the approximation order [46, 52],
   - by using several fine-mesh iterations with the projection of coarse mesh solution as an initial value [26].
3. Evaluating the error of the adjoint solution using its derivatives [53, 54].

Reconstruction is an efficient way to estimate the error, although this method does not incorporate the physics of the problem. Error evaluation using derivatives [53, 54] gives an upper bound for error estimation and this leads to overestimating the error. The computation of the adjoint approximation in a finer space is accurate but expensive for three-dimensional problems [45].

1.2.3. ADJOINT-BASED MESH ADAPTATION FOR STEADY FLOWS

Mesh adaptation based on adjoint methods is a mature technique for steady-state simulations (including RANS). Pierce and Giles [43, 51] systematically presented the construction of adjoint methods for linear and nonlinear problems, and applied them to error analyses of steady 1D/2D Poisson and quasi-1D Euler solutions. Such error estimations have been deployed to facilitate mesh adaptation in subsonic and transonic airfoil flow by Hartmann et al. [46] and Müller et al. [55], who found that mesh adaptation based on adjoint sensors is superior to that based on featured-based sensors. Venditti et al. [40] explored a similar adjoint-based adaptive mesh for 2D inviscid compressible flows [22, 56] after the development and verification of the approach on quasi-1D flows [40, 57]. The advantages of adjoint-based AMR relative to gradient-based adaption are demonstrated by their studies. Park [23] employed the adjoint method to drive unstructured mesh adaptation in three-dimensional applications, extending work by Venditti et al. [22] to 3D wing-body and high-lift configurations.

Nemec et al. [33] applied the adjoint-based technique to cut-cell FVM on Cartesian grids with applications to 3D inviscid compressible flows with sonic booms and around launch-vehicle configurations. Dwight [35, 58] related the added dissipation used in FVM to the accuracy of the function error via an adjoint method and applied this method to inviscid flows past a 2D NACA0012 airfoil and a 3D ONERA M6 wing. Other application examples are adjoint-based error indicators and mesh adaption for RANS simulations of a four-element airfoil [27] or a turbine distributor [59]. In these applications, computing cost was reduced drastically using the adjoint-based mesh adaptation comparing uniform mesh refinement. Shi et al. [30] verified the effectiveness of an adjoint-based adaptation approach using a high-order correction via reconstruction formulation.

In the FEM framework, adjoint-based error estimation and mesh adaptation have been studied for different problems. Becker et al. [15] summarized the development of a posteriori error analyses of elliptic, parabolic and hyperbolic problems, and then applied adjoint-based error estimation for practical applications. Becker et al. [16] analyzed adjoint-based mesh adaptation on a 2D incompressible circular cylinder flow. Through their work, the adjoint-based method was shown to provide a basis for the construction of economical meshes with considerable computing accuracy. Hartmann et al. [46] developed adjoint-based error estimation for a DG method and applied it to define a suitable mesh for the computation of the desired output quantities. They validated the procedure on transonic nozzle flows, as well as subsonic/supersonic airfoil flows. Kouhi et al. [32] utilized adjoint-based mesh adaptation with a finite calculus scheme for a Galerkin FEM, which added streamlined and transverse stabilization terms to the discretization. They demonstrated the capability of this adaptive approach in subsonic, transonic and supersonic regimes. Output-based mesh adaptation was demonstrated in high-order DG framework by Ceze et al. both for 2D [41] and 3D [60] turbulent flows.

In fluid flow problems, it is quite common to encounter anisotropic physical phenomena such as boundary layers, shock waves, and wakes, for which uniform mesh enrichment is not optimal. Some research works [47, 61, 62] consider these properties and use anisotropic mesh adaptation to efficiently obtain the solution. Alauzet et al. [48] reviewed the development and progress of anisotropic mesh adaptation based on adjoint variables in inviscid flows, relying on anisotropic ratios and quotients from the optimal goal-oriented metric. This is also an active field in mesh adaptation research.

1.2.4. ADJOINT-BASED MESH ADAPTATION FOR UNSTEADY FLOWS

The application of adjoint-based mesh adaptation to unsteady problems remains an active research field, although the basic approach has demonstrated its effectiveness for both incompressible and compressible turbulence flows as listed in Table 1.1. Linear time-dependent model problems including heat and acoustic wave equations were studied by Becker et al. [15] with DG framework. Barth [44] extended adjoint-based mesh adaptation with the DG method to 2D compressible NS flows. Hoffman et al. [54] combined stabilized Galerkin FEM with the adjoint method to adjust the mesh in incompressible flow problems. They extended this method to high-Re turbulent flows by using a wall shear stress model [14, 28] as well as to inviscid compressible flows with least squares stabilization [31] and residual-based artificial viscosity [29]. In the first method, the stabilization functions as a subgrid model in LES. The temporal and spatial adaptations were separately studied and controlled by the local error indicators in turbulent flows by Fidkowski [63] and Besier et al. [64]. Adjoint-based unsteady wake problems were studied by Krakos et al. [65] for application to anisotropic mesh adaptation in the DG method. However, the unsteady adjoint methods have received less attention than their steady counterparts because of their associated difficulties.

1.3. CHALLENGES IN ADJOINT-BASED AMR FOR LES

The application of adjoint-based mesh adaptation for LES presents new challenges. The unsteadiness and nonlinearity, as mentioned above, produce high-dimensional data in

Table 1.1: A summary of adjoint-based mesh adaptation in unsteady flow problems

Flow types	Incompressible	Compressible
Inviscid		1. High-*Re* wing-body flow [14] 2. Standard 3D high-lift Model [28] 3. 2D wedge/cylinder [31] & 3D sphere supersonic flow [29]
Viscous	1. Bluff body [53, 54] 2. Couette flow [49] 3. Laminar cylinder flow [64] 4. Sphere flow [66]	1. 2D scalar advection-diffusion [63]-reaction [67] problem 2. 2D cylinder subsonic flow [44] 3. Subsonic [63, 65] & transonic [67] airfoil gust encounter 4. Pitching and plunging airfoil [63] 5. Impulsively-started airfoil [65, 67]

unsteady flow problems, which requires a good trade-off between data storage, computational cost and solution accuracy. Furthermore, model error and discretization error are both influenced by the grid spacing or filter width. Stability is another crucial aspect for solving the unsteady adjoint problem in turbulent flows [39, 68]. These challenges in the application of the unsteady adjoint method to LES are described in detail below, leading to interesting research questions that we will explore and answer in this thesis.

1.3.1. HIGH-DIMENSIONAL DATA FOR TIME-DEPENDENT PROBLEMS

Whether the adjoint problem is solved efficiently and reliably determines the accuracy and effectiveness of the a posteriori error estimation. Considering time-dependent problems, the primal flow states, as coefficients for the adjoint problem, are needed in every time step when the adjoint problem is solved backward in time. It is apparent to consider storing the complete time series of primal solutions so as to provide primal flow information. However, their storage in memory is prohibitively expensive for large-scale turbulence problems. In order to handle this difficulty, some researchers choose to store the solution over a whole time interval on hard disk [64, 67, 69, 70], although the cost of data communication between hard disk and solver is much higher than directly accessing memory. Others choose to store snapshot solutions with a certain frequency and interpolate intermediate values in time [14, 28, 54]. An alternative is to make use of the checkpointing technique [71], which stores the primal solutions at selected states (so-called "checkpoints") and then recomputes them locally as needed. This technique has been employed for sensitivity analysis of statistical quantities in dynamical systems [72] and cylinder flows [39]. The disadvantage of this technique is the computational cost of repeated computations of the states between checkpoints.

Efficient techniques to solve the adjoint problem [44] are still required for error estimation in space-time problems. Order-reduction techniques show great potential for reducing the memory requirement of the primal flow states. A Reduced-Order Model (ROM) is a low-dimensional mathematical model that approximates a nonlinear system and thus allows for rapid evaluation of the specific output of a system. This characteristic

of ROMs makes them popular for problems in fluid dynamics [73–77]. ROMs are often built based on a Proper Orthogonal Decomposition (POD) [78–80], which is a well-known method to identify fluid structures based on their energy content. POD utilizes a small set of bases (POD modes) and their corresponding coefficients to express the flow field. These modes are extracted from the information of flow field snapshots (i.e., $u_r, r = 1,2,\cdots,m$). Through the eigendecomposition of the correlation matrix of those snapshots (u_r), POD modes (i.e., $\phi_i, i = 1,2,\cdots,M$) are selected as the sufficient number of eigenvectors whose eigenvalues account for the most kinetic energy of the correlation matrix. M denotes the truncation number of POD modes and is usually much less than the spatial dimensional of primal solutions. The flow solution is approximated as a linear combination of POD modes, amounting to

$$\tilde{u}(x,t) = \sum_{i=1}^{M} c_i(t)\phi_i(x), \tag{1.1}$$

where c_i is the parameter related to flow states and $\tilde{u}(x,t)$ denotes the low-order reconstruction. A ROM to determine the $c_i(t)$ can be obtained by a Galerkin projection employing the POD modes, $\phi_i(x)$. However, there are several aspects that need to be rigorously considered to take advantage of the substantial computational reduction of ROMs. Firstly, a limited number of modes should suffice to represent the system. This can be verified by whether the decay of eigenvalues is sufficiently rapid or not. Secondly, the sampling of snapshots for the POD should include dynamic features that are important to describe flow characteristics. The greedy algorithm [81] or the evaluation of the information captured by POD can be used to meet this requirement. Besides, a ROM should consider essential dynamical structures of the flow problem and thus be capable of representing the flow physics through these selected modes. The flow manifold should be a local branch of the nonsingular solutions in parameter space for the construction of a ROM.

The computation of $c_i(t)$ using Galerkin projection has been applied to steady [81] and unsteady [82] viscous problems. However, the truncation of POD modes results in the neglect of part of the underlying characteristics of a dynamical system. As a result, some Galerkin ROMs lack long-time stability [81, 83]. Another approach is to compute the coefficients, $c_i(t)$, directly from snapshot solutions. This can be achieved by means of the least-square method and interpolated for other times via radial basis functions [84] or neural networks [85], for instance. To distinguish this approach from the Galerkin projection, the latter approach is called a reduced-order representation (ROR) in this thesis since there is no model to be solved. Given that there is no need to predict the primal solutions in the future when we solve the unsteady adjoint problem, the $c_i(t)$ of a ROR can be reproduced from the primal flow solutions previously solved from the governing equations, avoiding the re-computation of primal solutions.

The computation cost of standard batch SVD (or offline POD method) is proportional to $\mathcal{O}(nm^2)$, where n denotes the DoF and m represents the number of snapshots. This becomes computationally intractable for LES since both of n and m are large values. The issue of high-dimensional data can be handled by performing online POD analysis using the incremental Singular Value Decomposition (SVD) [86, 87]. Incremental SVD is an alternative way to achieve POD analysis without recording flow solutions. It was

applied to unsteady flow simulations in Ref. [88]. Baker et al. [87] noted that the cumulative cost of the incremental SVD, $\mathcal{O}(nm^3)$, is higher than the batch SVD, $\mathcal{O}(nm^2)$, which leads to a high computational cost for long-time statistical simulations. Selection of snapshots [89, 90] and an error estimator [91] were proposed to reduce the matrix size required for the incremental SVD and thus constrained the increase of the associated computational cost. But this kind of improvement is obtained by sacrificing part of the snapshot solutions. In contrast, the truncation of POD modes has been considered for the incremental SVD and used for adjoint-based optimization in unsteady flows [92]. However, it remains vague how to choose the truncation for a practical problem. This is a key point for the application to LES.

1.3.2. MODEL ERROR

When the adjoint method is used in mesh adaptation for steady RANS simulations, the numerical error is controlled by the grid spacing. As a result, the error solely depends on the turbulence model when a sufficiently fine mesh is used. In that sense, the mesh adaptation is only used to reduce the discretization error in RANS. However, mesh adaptation in LES is different from RANS since the grid size in LES will affect not only the truncation error of the numerical discretization but also the model error [93]. If the model error is assumed to be smaller than the numerical error or proportional to the numerical error, then the numerical error could be estimated with confidence [7]. However, this is not always the case for LES. Because of the competition between the model error and numerical error, some researchers maintain a fixed filter width to verify the mesh independence of LES [94]. If the filter width is modified, grid-converged solutions would be changed as well. In implicit LES, the model error is linked to the discretization error which depends on the computational mesh. Therefore, these two factors should be carefully treated in error estimation for LES.

The convergence of a QoI has a relation with the general convergence of the solution but can be different from the latter. Certainly, the convergence of the flow solution leads to obtaining a QoI with absolute confidence. However, LES can give the well-converged QoI, such as lift or drag, before a DNS-like refinement is reached. For filter-based or implicit LES, the mesh convergence of a QoI is defined as the convergence with respect to the computing mesh size, which can be determined by the mesh adaptation process.

Adjoint-based error estimation requires the computation of the adjoint solution on a fine space, which can be expensive for large-scale problems. This can be avoided using reconstruction on a finer mesh [7, 93]. Generally, there is no physical support for interpolated or reconstructed error estimators and thus one must have less confidence in these error estimations [67]. We noted that the sub-grid model in LES describes the unresolved/fine-scale solution and represents the influence on the resolved scales in LES. Sub-grid models can thus also be used for the computation of the solution in a fine space.

Considering the multiscale features of turbulent flows, the Variational Multiscale Method (VMM) [95–97] is a good choice for modelling turbulent flows since VMM solves the flow problem at different scales. Specifically, the VMM divides the flow variables into resolved and unresolved scales, with the unresolved scales being modelled. This approach serves as an alternative numerical algorithm for LES. The unresolved scale

is typically modelled using the residual of resolved-scale flow solutions to circumvent the complexity of finding unresolved-scale solutions in non-linear multi-dimensional problems. Bazilevs et al. [98] presented the LES-type VMM for incompressible turbulent flows and validated it on forced homogeneous isotropic turbulence and turbulent channel flows. Akkerman et al. [99] examined the impact of basis functions continuity in VMM using non-uniform rational B-splines (NURBS) and found that the NURBS basis functions outperformed standard finite elements in turbulent flow calculations while the impact of continuity becomes more significant for higher Re cases. Hulshoff et al. [100] developed two methods to deal with the wall-stress boundary conditions in VMM for simulating turbulent flows and found that the method using wall jump values was more suitable for practical application. Chen et al. [101] employed residual-based VMM to solve the flow past a cylinder mounted above a flat plate and found that the calculation of hydrodynamic forces agreed well with the traditional LES approaches. The unresolved scale can be regarded as the computational error in the multiscale LES and thus it can be utilized to estimate the error in the QoI. Granzow et al. [102] illustrated the VMM-based error estimation and mesh adaptation in 2D linear advection-diffusion problems. However, adjoint-driven mesh adaptation in a VMM framework for unsteady nonlinear problems remains rare.

1.3.3. CHAOTIC NATURE OF TURBULENCE

Turbulent flows are inherently chaotic, which means small variations in design parameters can cause significant changes in the flow field. Consequently, this sensitivity to variations can lead to a substantial impact on the value of a QoI, and it becomes increasingly difficult to compute the gradient accurately. The adjoint problem is typically employed to compute the gradient of a QoI with respect to design parameters. However, due to the chaotic nature of turbulent flows, the adjoint field can undergo significant changes or even diverge, posing challenges in solving the adjoint problem. Anti-diffusion is also present in the unsteady adjoint equations, increasing the difficulty of computing bounded solutions by time marching methods. The resulting stability issues when solving the adjoint problem for turbulent flows have received considerable attention. Lea et al. [68] reported that the sensitivity of a time-averaged QoI can grow exponentially for long computational times in the Lorenz (1963) system that retains essential chaotic behaviour of turbulent flows. An ensemble-adjoint method was developed to compute the averaged sensitivity from different initial conditions at an intermediate time scale. Using Direct Numerical Simulation (DNS), Wang et al. [39] showed the divergence of drag-based adjoint solutions for the turbulent flow around a cylinder at a Reynolds number of 500. Numerous remedies have been proposed, including ensemble sensitivity [103], least-squares shadowing (LSS) [104], space-split sensitivity [105], and additional artificial viscosity [106]. Key methods for controlling the stability of adjoint problems include modifications of the averaged-time length or finding a substitute for the chaotic primal solution, for example, one defined on a shadowing trajectory in phase space.

Different techniques have been employed to reduce the computing cost and large storage requirements of LSS. The Multiple Shooting Shadowing (MSS) method was proposed in Ref. [107], which considers the norm of the distance between reference and

shadowing trajectories at checkpoints rather than all time steps. Shawki et al. [108] improved the computing efficiency of the MSS algorithm using a block diagonal matrix-free preconditioner. A non-intrusive LSS method was developed in Ref. [109] to reduce the computing cost of LSS using a linear combination of one inhomogeneous tangent solution and several homogeneous tangent solutions. The resulting computing cost is proportional to the number of unstable/positive Lyapunov Exponents. Lasagna et al. [110] formulated a periodic shadowing method for flows with periodic boundary conditions in time. This method can bound adjoint solutions for an infinite orbit length and the sensitivities can be computed exactly for every orbit. Conversely, Hoffman et al. [14, 111] observed stable adjoint solutions at high Reynolds numbers computed with wall-modelled boundary conditions. The stability of the adjoint method is still a vital topic for practical applications. How to define stable and efficient adjoint methods for turbulent flow problems remains an important but open question.

1.4. THESIS OBJECTIVE

This thesis will address the above challenges and pave the way to intervention-free LES for practical engineering analyses, assessment and optimization. To make adjoint-based error estimation tractable and accurate for LES, the issue of excessive storage requirements for high-dimensional snapshot data will be addressed using reduced-order representations based on incremental SVD. The resulting methodology is presented and investigated in different flow problems. The issue of determining appropriate truncation levels will also be tackled. The research questions below are considered.

- What kind of factors are important for reliable adjoint-based error estimation in a VMM framework for the computation of a QoI? How do they influence mesh adaptation in unsteady problems?
- How can an appropriate truncation number for online POD analysis be chosen when using incremental SVD for unsteady problems? What are the effects on accuracy and parallel performance for high-dimensional data sets?
- How does using a ROR impact the efficiency of adjoint-based error estimation and the dynamics of adjoint problems for unsteady flows? How does using a ROR affect the growth of adjoint solutions for turbulent flows?

1.5. THESIS OUTLINE

The remainder of this dissertation consists of the following sections:

- Chapter 2 studies error estimation using VMM-driven adjoint solutions for steady linear and nonlinear model problems.
- Chapter 3 investigates how to make the ROR feasible for high-dimensional problems using an enhanced parallel incremental SVD.
- Chapter 4 demonstrates the applicability of adjoint-based AMR and the effects of the ROR choice on AMR in one-dimensional forced Burgers problems.
- Chapter 5 presents the impacts of using RORs on the adjoint problem and adjoint-based error estimation for multi-dimensional transient flow problems, including

the dynamics of the adjoint system, the accuracy and efficiency, as well as the stability in 3D turbulent flows.

- Chapter 6 summarizes the main findings of the present work and provides recommendations for future developments.

2

Adjoint-based error estimation in VMM for steady problems

The modelling error in LES plays an important role in the error estimation based on the adjoint method. We consider the Variational Multiscale Method (VMM) to evaluate the unresolved-scaled solution and then approximate the actual flow solution in error estimation. The associated adjoint equation is introduced to achieve output-based error analysis. Computational results for advection-dominant problems are presented to verify and validate the efficiency and accuracy of adjoint-based error estimates compared with mesh-refined calculations.

2.1. INTRODUCTION

The accurate prediction of integral statistical quantities, such as lift, drag, thrust and engine efficiency [112], is of primary interest in design, optimization and characterization analyses in engineering aerodynamic applications. The adjoint method can be used to connect the local error of flow solutions with the error in a Quantity of Interest (QoI) [9]. Thus adjoint-based error estimation is utilized to achieve mesh adaptation in FEM [14, 31, 51, 102] and FVM [22, 40]. In the literature, the continuous adjoint is widely used in FEM whereas the discrete adjoint is typically applied in FVM, with differences in the formulation and the implementation of the adjoint problem. Adjoint-based error estimates were reported to improve the prediction of QoI values without extensively adding extra computing cost compared to results from uniformly refined meshes in literature.

Turbulent flow simulations require representing the effects of different flow scales, such as from *Kolmogorov scale* to macro-scale (e.g. the chord length of a 2D airfoil). LES based on the VMM was proposed to efficiently represent such multiscale phenomena [95]. The VMM can be used to formulate subgrid models used in LES by dividing the flow solution into resolved and unresolved scales. The effects of the unresolved scales are here modelled based on the residual of resolved flow solutions in order to circumvent the complexity of finding unresolved-scale solutions in non-linear multi-dimensional problems. Actually, the unresolved scale represents the discretization error in the VMM and thus can be employed to calculate the estimated error in the QoI. Through the combination of the adjoint solution and VMM, the accuracy of error estimation can be improved significantly, as will be shown in Section 2.5.

This chapter is organized as follows. The related adjoint problem is explained for linear problems in Section 2.2, where the adjoint identity is emphasized in continuous and discrete spaces, respectively. In Section 2.3, we describe the framework of VMM-based LES and the associated implementation for flow and adjoint problems. General adjoint-based error estimation is discussed in Section 2.4. The conventional estimation used in classical FEM is introduced and four variants of a VMM-driven estimator are proposed. The validation and evaluation of these error estimations are presented in numerical results of linear advection-diffusion problems in Section 2.5. We summarize the application of the adjoint method to error estimation in VMM.

2.2. ADJOINT PROBLEM

2.2.1. MODEL DESCRIPTION

Let $\Omega \subset \mathbb{R}^{n_{\text{dim}}}$ be an open bounded computing domain in n_{dim} dimensions. Its boundary is represented by $\partial\Omega$. For fluid problems with homogeneous Dirichlet boundary condi-

tions, $\mathcal{V}$ is the solution space $\mathcal{V} = \{u | u \in H(\Omega), u|_{\partial\Omega} = 0\}$, where $H(\Omega)$ is a Hilbert space defined in Ω. The corresponding space for adjoint solutions is $\mathcal{V}^*$. The abstract model problem for flow solutions, $u \in \mathcal{V}$, is described as follows

$$\begin{cases} Lu = f, x \in \Omega \\ u = 0, x \in \partial\Omega, \end{cases} \tag{2.1}$$

where L is a linear second-order differential operator, such as the one in Poisson and advection-diffusion problems, and f is a given function in $\mathbb{R}$. The equivalent weak form of this linear problem is demonstrated as finding $u \in \mathcal{V}$ such that

$$(\omega, Lu) = (\omega, f), \forall \omega \in \mathcal{V}, \tag{2.2}$$

where $(\cdot, \cdot)$ denotes the inner product in $\mathcal{V}$ with the definition of

$$(p, q) = \int_\Omega pq \, \mathrm{d}\Omega, \, p \in \mathcal{V}, q \in \mathcal{V}. \tag{2.3}$$

2.2.2. RELATED ADJOINT PROBLEM

The adjoint problem is introduced to connect the local residual to the QoI and the adjoint solutions are used to estimate the approximation error of the QoI. The adjoint equation is determined by the flow problem and the QoI. Let us consider a physical QoI, $J(u) \in \mathbb{R}$, as a volume integral output [51, 102] with the definition

$$J(u) = (g, u), \tag{2.4}$$

where $g \in \mathcal{V}^*$.

The *adjoint identity* is the basis for the introduction of the adjoint problem. It is derived by integration by part as follows

$$(v, Lu) = (L^* v, u), \tag{2.5}$$

where $v \in \mathcal{V}^*$, $u \in \mathcal{V}$, and L^* is the *adjoint operator*.

Piecewise basis functions are widely utilized in FEM to discretize Partial Differential Equations (PDEs). The computing domain is normally discretized into non-overlapping cells Ω_e in the discrete space $\mathcal{V}_h$, with cell boundary $\partial\Omega_e, e = 1, 2, \ldots, N_{\text{cells}}$. The subscript h denotes the average cell size of a computational mesh. For the demonstration of the next parts, two union notations are presented here: the union of cell interiors Ω' and the union of cell interfaces and boundaries $\partial\Omega'$, viz.

$$\Omega' = \bigcup_{e=1}^{N_{\text{cells}}} \Omega_e, \quad \partial\Omega' = \bigcup_{e=1}^{N_{\text{cells}}} \partial\Omega_e, \tag{2.6}$$

where N_{cells} denotes the number of mesh cells.

For the *adjoint identity* in discrete space, the right-hand term $(L^* v, u)$ in Equation (2.5) consists of a *Dirac distribution* in the domain Ω owing to the slope discontinuity of shape functions. This relation is elaborated as follows,

$$(v, Lu) = (L^* v, u)_{\Omega'} - \big((\llbracket B^* v \rrbracket, u) + (v, \llbracket Bu \rrbracket)\big)_{\partial\Omega'} = (L^* v, u)_\Omega, \tag{2.7}$$

where B and B^* are the *boundary operators* corresponding to L and L^*, respectively. $[\![\cdot]\!]$ represents the *jump operator* at cell interfaces and is defined as

$$[\![\cdot]\!] = (\cdot)_{\text{interface right}} - (\cdot)_{\text{interface left}} = (\cdot)_+ - (\cdot)_- . \tag{2.8}$$

Equation (2.7) gives a union expression and L^* includes the boundary integration over the entire computing domain Ω. Taking the linear advection-diffusion problem in one dimension as an example, we explain the *adjoint identity* as the following

$$\begin{aligned}
(v, Lu) &= \int_\Omega v(\lambda \nabla u - \nu \Delta u)\, \mathrm{d}\Omega \\
&= \sum_{e=1}^{N_{\text{cells}}} \int_{\Omega_e} v(\lambda \nabla u - \nu \Delta u)\, \mathrm{d}\Omega_e \\
&= \sum_{e=1}^{N_{\text{cells}}} \left[\int_{\Omega_e} (-\lambda \nabla v - \nu \Delta v) u\, \mathrm{d}\Omega_e + \left(\lambda u v - \nu v \frac{\mathrm{d}u}{\mathrm{d}x} + \nu u \frac{\mathrm{d}v}{\mathrm{d}x} \right) \Big|_{\partial \Omega_e} \right] \\
&= (L^* v, u)_{\Omega'} + (\lambda u v - \nu v \frac{\mathrm{d}u}{\mathrm{d}x} + \nu u \frac{\mathrm{d}v}{\mathrm{d}x}) \Big|_{\partial \Omega'} \\
&= (L^* v, u)_\Omega .
\end{aligned} \tag{2.9}$$

The adjoint problem is then defined as

$$\begin{cases} L^* v = g, x \in \Omega \\ \quad v = 0, x \in \partial\Omega . \end{cases} \tag{2.10}$$

with an equivalent weak form

$$(\omega, L^* v) = (\omega, g),\ \forall \omega \in \mathcal{V}^* . \tag{2.11}$$

2.3. FORMULATION OF THE VMM

A brief description of the VMM technique in LES is stated in this section, and the process of solving the primal and adjoint equations using this approach is presented.

2.3.1. THE FRAMEWORK OF VMM

In VMM, the flow solution is decomposed into resolved scales $\overline{u}$ and unresolved scales u', i.e. $u = \overline{u} + u'$, where $\overline{u} \in \overline{\mathcal{V}}, u' \in \mathcal{V}'$, $\overline{\mathcal{V}}$ and $\mathcal{V}'$ are subsets of $\mathcal{V}$, and $\overline{\mathcal{V}} \oplus \mathcal{V}' = \mathcal{V}$. The $\overline{u}$ and u' are also referred to as computable *coarse-scale* and *fine-scale* solutions. By applying this decomposition to the weak form Equation (2.2), the model problem is re-expressed as

$$(\omega, L(\overline{u} + u')) = (\omega, f),\ \forall \omega \in \mathcal{V} . \tag{2.12}$$

By considering test functions ω in the same space as u, ω can be similarly decomposed as $\overline{\omega}$ and ω'. Equation (2.12) is then divided into two subproblems

$$(\overline{\omega}, L\overline{u}) + (\overline{\omega}, Lu') = (\overline{\omega}, f),\ \forall \overline{\omega} \in \overline{\mathcal{V}} , \tag{2.13}$$

$$(\omega', L\overline{u}) + (\omega', Lu') = (\omega', f), \forall \omega' \in \mathcal{V}' . \tag{2.14}$$

The key process of VMM is to identify an approximation of the unresolved scales u' in terms of $\overline{u}$ so that the governing equation of the resolved scales can be solved independently. To this end, the unresolved-scale Green's function $g'(x;y)$ [95, 113] is employed with the following definition

$$\begin{cases} L^* g'(x,y) = \delta(x-y) & \text{in} \quad \Omega \\ g'(x,y) = 0 & \text{on} \quad \partial\Omega\,. \end{cases} \tag{2.15}$$

Replacing the ω' in Equation (2.14) with $g'(x,y)$, we can obtain the analytical expression for u', i.e.

$$u'(y) = -\int_\Omega g'(x,y)(L\overline{u} - f)\,\mathrm{d}\Omega = -\int_\Omega g'(x,y) R(\overline{u})\,\mathrm{d}\Omega\,, \tag{2.16}$$

where $R(\cdot) = L(\cdot) - f$ is the residual operator of the resolved scales. It can also be written using an integral operator M' as

$$u'(y) = M' R(\overline{u})\,, \tag{2.17}$$

where $M'(\cdot) = -\int_\Omega g'(x,y)(\cdot)d\Omega$. Substituting the expression of u' into Equation (2.13), the resolved-scale problem is expressed as

$$(\overline{\omega}, L\overline{u}) + (L^*\overline{\omega}, M' R(\overline{u})) = (\overline{\omega}, f),\ \forall \overline{\omega} \in \overline{\mathcal{V}}\,. \tag{2.18}$$

This equation is referred to as the *general formulation of VMM*. This model is precise for the resolved scales and can be used in the spectral FEM directly. However, the computation is costly because of the double integration in the second term in Equation (2.18), especially for multi-dimensional engineering problems. Thus, u' is approximated to provide an equivalent contribution to the resolved-scale problem. This is discussed in the following section.

2.3.2. DISCRETE FORMULATION OF VMM

The FEM is normally used to solve Equation (2.18). When piecewise shape functions are used, $\overline{u}$ and $\overline{\omega}$ are smooth over element interiors, but their slopes are discontinuous across element interfaces. This gives rise to additional jump terms for the integral over the domain using integration by parts. Here, $(L^*\overline{\omega}, M' R(\overline{u}))$ is thus regarded as a representation for the expression including the jump across elements. L^* can be viewed as a *Dirac distribution* [95] in the entire computing domain Ω.

In the discrete setting, $u_h, \omega_h \in \mathcal{V}_h$ and $\omega'_h, u'_h \in \mathcal{V}'_h$ denote the resolved and unresolved solutions, where $\mathcal{V}_h$ and $\mathcal{V}'_h$ are the resolved and unresolved spaces, respectively. The *algebraic operator* τ is introduced as

$$\tau = -M'\,. \tag{2.19}$$

Substituting it into Equation (2.18), the resolved-scale problem can be expressed as

$$(\omega_h, Lu_h) + \left(L^*\omega_h, -\tau R(u_h)\right) = (\omega_h, f),\ \forall \omega_h \in \mathcal{V}_h\,, \tag{2.20}$$

which requires finding an explicit formulation for τ. An element Green's function $g_e^{'}(x,y)$ is introduced to approximate the exact one in Equation (2.15),

$$\begin{cases} L^* g_e^{'}(x,y) = \delta(x-y) & \text{in} \quad \Omega_e \\ g_e^{'}(x,y) = 0 & \text{on} \quad \partial\Omega_e \,, \end{cases} \tag{2.21}$$

we define a computable form of τ by using the average value on the element interior,

$$\tau \approx \tau_e = \frac{1}{\text{meas}(\Omega_e)} \int_{\Omega_e} \int_{\Omega_e} g_e^{'}(x,y) \mathrm{d}\Omega_{e,x} \mathrm{d}\Omega_{e,y} \,, \tag{2.22}$$

where $\text{meas}(\Omega_e)$ is the measured size of mesh cells. For the one-dimensional linear advection-diffusion problem, for example,

$$\lambda \nabla u - \nu \Delta u = f \,, \tag{2.23}$$

an analytic element Green's function[113] can be found by solving problem 2.21, allowing τ_e to be approximated by

$$\tau_e = \frac{1}{h_e} \int_{\Omega_e} \int_{\Omega_e} g_e^{'}(x,y) d\Omega_{e,x} \, \mathrm{d}\Omega_{e,y} = \frac{h_e}{2\lambda} (\coth \alpha - \frac{1}{\alpha}) \,, \tag{2.24}$$

where $\alpha = \lambda h_e / (2\nu)$ and h_e is the element length. Using the residual operator R, the discrete VMM in Equation (2.20) is then equivalent to

$$\left(L^* \omega_h, -\tau R(u_h)\right) = -(\omega_h, R(u_h)), \, \forall \omega_h \in \mathcal{V}_h \,. \tag{2.25}$$

2.3.3. VMM FORMULATION FOR ADJOINT PROBLEM

The adjoint problem is solved using the same VMM used for the primal problem. The domain space $\mathcal{V}^*$ is chosen as equal to $\mathcal{V}$. Thus the adjoint solution is decomposed into the resolved-scale solution $\overline{v}$ and unresolved-scale solution v', i.e. $v = \overline{v} + v'$, where $\overline{v} \in \overline{\mathcal{V}}$, $v' \in \mathcal{V}'$. After introducing the adjoint fine-scale Green's function $g'_*(x,y)$ and the integral operator M'_*, the *general formulation of VMM* for the adjoint problem is

$$(\overline{\omega}, L^* \overline{v}) + (L\overline{\omega}, M'_* R_*(\overline{v})) = (\overline{\omega}, g), \, \forall \overline{\omega} \in \overline{\mathcal{V}} \,, \tag{2.26}$$

where $R_*(\cdot) = L^*(\cdot) - g$, is the adjoint residual operator of the resolved scales, and the unresolved scales in the adjoint problem are represented as

$$v'(y) = M'_* R_*(\overline{v}) \,, \tag{2.27}$$

where $M'_*(\cdot) = -\int_\Omega g'_*(x,y)(\cdot)\, \mathrm{d}\Omega$.

Here, the discrete space for the primal problem is also used to solve the adjoint problem, such that $\mathcal{V}_d = \mathcal{V}_h$. The corresponding discrete formulation of the adjoint problem is obtained using an element-averaged Green's function τ_e^*, expressed as

$$\tau_e^* = \frac{1}{\text{meas}(\Omega_e)} \int_{\Omega_e} \int_{\Omega_e} g_{*,e}^{'}(x,y) \, \mathrm{d}\Omega_{e,x} \mathrm{d}\Omega_{e,y} \,. \tag{2.28}$$

Consequently, the discrete adjoint problem is

$$(\omega_h, L^* v_h) + \left(L\omega_h, -\tau^* R_*(v_h)\right) = (\omega_h, g), \forall \omega_h \in \mathcal{V}_h \,. \tag{2.29}$$

2.4. ADJOINT-BASED ERROR ESTIMATION

Truncation error is introduced when the discrete solution u_h is used to evaluate the QoI. In order to get precise high-fidelity computation, we need to identify this error. In this section, we derive error approximations for the general formulation and for classical FEM and VMM respectively.

2.4.1. ERROR ESTIMATION FORMULA

Here, we present a general framework for adjoint-based error estimation. Let $\tilde{u}$ be an obtainable approximation value, and the corresponding deviation $\tilde{u}' = u - \tilde{u}$ from the exact u. When a linear QoI is defined as $J(u) = (g, u)$, the approximation error can be expressed by

$$\epsilon = J(u) - J(\tilde{u}) = (g, u) - (g, \tilde{u}) = (g, u - \tilde{u}) = (g, \tilde{u}')\,. \tag{2.30}$$

With consideration of the *adjoint operator* and the following definitions,

$$\begin{aligned} L\tilde{u} &= \tilde{f} \\ L^*\tilde{v} &= \tilde{g}\,, \end{aligned} \tag{2.31}$$

we re-write Equation (2.30) as

$$\begin{aligned} \epsilon &= J(u) - J(\tilde{u}) \\ &= (g, u - \tilde{u}) \\ &= (\tilde{g}, u - \tilde{u}) + (g - \tilde{g}, u - \tilde{u}) \\ &= (L^*\tilde{v}, u - \tilde{u}) + (g - \tilde{g}, u - \tilde{u}) && \text{by definition} \\ &= (\tilde{v}, L(u - \tilde{u})) + (g - \tilde{g}, u - \tilde{u}) && \text{adjoint opeator} \\ &= (\tilde{v}, f - \tilde{f}) + (g - \tilde{g}, u - \tilde{u}) && \text{by linearity and definition} \\ &= - \underbrace{(\tilde{v}, R(\tilde{u}))}_{\text{Adjoint Eorrection}} + \underbrace{(R_*(\tilde{v}), \tilde{u} - u)}_{\text{Remaining Error}}\,. && \text{by residual operator} \end{aligned} \tag{2.32}$$

The first term is referred to as *adjoint correction* and is computable from the adjoint solution and the residual of the primal problem. The second term is named the *remaining error*. This is unobtainable since the exact solution (u) of the primal problem usually is unknown.

2.4.2. ERROR REPRESENTATION IN CLASSICAL FEM

In the classical FEM, an error estimation is obtained by replacing $\tilde{u}$ and $\tilde{v}$ with u_h and v_h, respectively. The adjoint-based error estimation is then re-expressed in terms of these discrete solutions as follows

$$\begin{aligned} \epsilon &= (g, u) - (g, u_h) \\ &= (g_h, u - u_h) + (g - g_h, u - u_h) \\ &= -(v_h, f_h - f) + (g_h - g, u_h - u)\,, \end{aligned} \tag{2.33}$$

where $f_h = Lu_h$ and $g_h = L^* v_h$ are the source terms approximated using discrete solutions.

2.4.3. CONTINUOUS VMM ERROR EXPRESSION

We now derive the error estimation for the *general VMM formulation* (Equation (2.18)). $\overline{u}$ and $\overline{v}$ are used to replace aforementioned $\tilde{u}$ and $\tilde{v}$ in Equation (2.32), where the unresolved scales are u' and v'. The below relation can then be written

$$\epsilon_{vmm} = J(u) - J(\overline{u}) = (g, u') = (g, -\tau R(\overline{u}))\,. \tag{2.34}$$

This is a direct way to calculate the error estimation, which does not need the solution of the adjoint problem. When the adjoint solution is used, the error estimation is re-stated as

$$\begin{aligned}\epsilon_{vmm,1} &= (\overline{g}, u - \overline{u}) + (g - \overline{g}, u - \overline{u}) \\ &= (L^*\overline{v}, u') + (L^* v - L^*\overline{v}, u') \\ &= -(L^*\overline{v}, \tau R(\overline{u})) + (R^*(\overline{v}), \tau R(\overline{u}))\,,\end{aligned} \tag{2.35}$$

where, $\overline{g} = L^*\overline{v}$. Note, from Equation (2.35), that both terms in the first expression are dependent on the adjoint solution. Using the *adjoint identity* gives us the following relation

$$(L^*\overline{v}, -\tau R(\overline{u})) = (L^*\overline{v}, u') = (\overline{v}, Lu') = (\overline{v}, -R(\overline{u})) \tag{2.36}$$

$$(R^*\overline{v}, \tau R(\overline{u})) = (L^* v - L^*\overline{v}, u') = (v', Lu') = (\tau^* R^*(\overline{v}), R(\overline{u}))\,. \tag{2.37}$$

Combining these two expressions, four categories of adjoint-based error estimation can be generated for the VMM. The first is given in the last line of Equation (2.35). The other three are as follows

$$\epsilon_{vmm,2} = -(\overline{v}, R(\overline{u})) \quad + (R^*\overline{v}, \tau R(\overline{u})) \tag{2.38}$$

$$\epsilon_{vmm,3} = -(\overline{v}, R(\overline{u})) \quad + (\tau^* R^*(\overline{v}), R(\overline{u})) \tag{2.39}$$

$$\epsilon_{vmm,4} = -(L^*\overline{v}, \tau R(\overline{u})) + (\tau^* R^*(\overline{v}), R(\overline{u})) \tag{2.40}$$

The last expression is the same as the one suggested by Granzow et al. [102].

2.4.4. DISCRETE VMM ERROR EXPRESSION

To use the VMM for error estimation, we need to derive a formulation in the discrete space. In fact, two different paths can be utilized to represent the error estimation in a discrete space: starting from the universal error formula or starting from the previous continuous expression. The latter is used here to compute the adjoint-based error estimation in a discrete space. The resolved-scale solution $\overline{u}$, from here, is replaced by u_h in discrete space; at the same time, u'_h is the substitute of u', and $\hat{u} = u - u_h - u'_h$. Substituting these relations into Equation (2.34), we obtain

$$\epsilon_{vmmd} = J(u) - J(u_h) = (g, u - u_h) = (g, u'_h + \hat{u}) = (g, -\tau R(u_h)) + (g, \hat{u})\,, \tag{2.41}$$

where the residual operator is $R(u_h) = f - Lu_h$, and $u'_h = -\tau R(u_h)$. The representations derived by the adjoint solution and *adjoint identity* are given as follows:

$$\epsilon_{vmmd,1} = -(L^* v_h, \tau R(u_h)) + (R^*(v_h), \tau R(u_h)) + (L^* v, \hat{u}) \tag{2.42}$$

$$\epsilon_{vmmd,2} = -(v_h, R(u_h)) \quad + (R^*(v_h), \tau R(u_h)) + (-R^*(v_h), \hat{u}) \tag{2.43}$$

$$\epsilon_{vmmd,3} = -(v_h, R(u_h)) \quad + (\tau^* R^*(v_h), R(u_h)) + (\hat{v}, -R(u_h)) \tag{2.44}$$

$$\epsilon_{vmmd,4} = -(L^* v_h, \tau R(u_h)) + (\tau^* R^*(v_h), R(u_h)) + (\hat{v}, -R(u_h)) + (L^*(v_h), \hat{u}) \tag{2.45}$$

It is obvious that the terms involving representation errors ($\hat{u}$ and $\hat{v}$) make supplemented contributions to the error estimation, which are distinct from the continuous one. However, these terms are beyond the computation of the discrete VMM because of unknown variables $\hat{u}$ and $\hat{v}$. Thus we neglect the terms with those factors so that the simplified expressions are

$$\widetilde{\epsilon}_{vmmd,1} = -(L^* v_h, \tau R(u_h)) + (R^*(v_h), \tau R(u_h)) \tag{2.46}$$

$$\widetilde{\epsilon}_{vmmd,2} = -(v_h, R(u_h)) \quad + (R^*(v_h), \tau R(u_h)) \tag{2.47}$$

$$\widetilde{\epsilon}_{vmmd,3} = -(v_h, R(u_h)) \quad + (\tau^* R^*(v_h), R(u_h)) \tag{2.48}$$

$$\widetilde{\epsilon}_{vmmd,4} = -(L^* v_h, \tau R(u_h)) + (\tau^* R^*(v_h), R(u_h)) \tag{2.49}$$

2.5. NUMERICAL RESULTS

In this section, we investigate the error estimation of a QoI in one-dimensional linear problems and discuss the effectiveness and accuracy of various error estimations. Adjoint-based error estimations are from both classical FEM and VMM.

2.5.1. VERIFICATION USING A POISSON PROBLEM

The Poisson problem is first considered to demonstrate the process of error estimation and to evaluate its effectiveness. The governing equation of the primal problem is

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = f, \tag{2.50}$$

where the source term is $f = x^3(1-x)^3$ and $x \in [0,1]$, with the homogeneous boundary condition $u(0) = u(1) = 0$. The analytic solution to this Poisson problem is

$$u(x) = \frac{1}{56}x(1-x^7) - \frac{1}{14}x(1-x^6) + \frac{1}{10}x(1-x^5) - \frac{1}{20}x(1-x^4). \tag{2.51}$$

The QoI is defined by $J(u) = (g, u)$, where $g = \sin(\pi x)$. The associated adjoint equation is

$$\frac{\mathrm{d}^2 v}{\mathrm{d}x^2} = g, \tag{2.52}$$

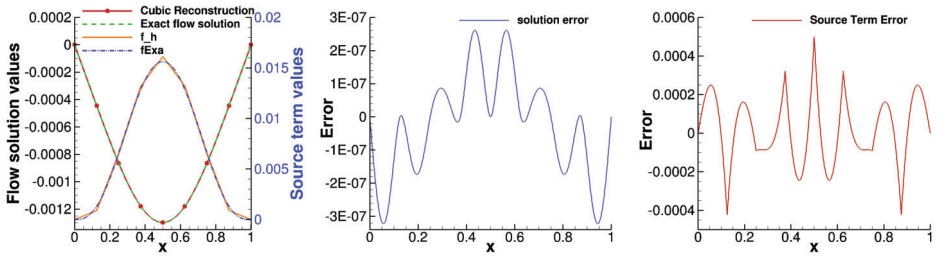
subject to homogeneous boundary conditions as well.

We use the classical FEM to solve the flow and adjoint equations on the same mesh. Linear basis functions are used to obtain the primal and adjoint solutions at grid nodes. Cubic reconstruction is employed to estimate the error of the approximated QoI.

THE VERIFICATION OF FLOW AND ADJOINT SOLVER

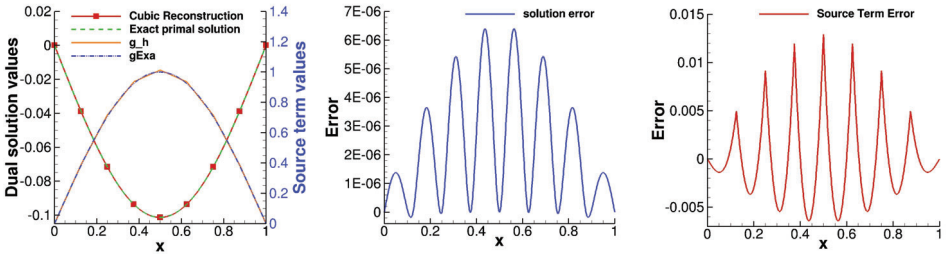
The accuracy of the flow solver is illustrated by the comparison between discrete and analytical solutions. Figure 2.1(a) shows flow solutions solved using 8 elements compared with the analytical solution. Good agreement is observed but there are still truncation errors in numerical solutions as shown in Figure 2.1(b). Figure 2.1(c) shows the approximation error of the source term reconstructed by a cubic spline.

Similar results for the adjoint problem are shown in Figure 2.2. In both cases the errors are small. Consequently, the current FEM solver is verified based on these results and we can employ it to investigate error estimation based on the adjoint method.



(a) Flow solutions & source term (b) Error of flow solutions (c) Error of source term

Figure 2.1: Flow solutions, associated error distribution, and approximated error of source term in the flow problem.



(a) Adjoint solutions & source term (b) Error of adjoint solutions (c) Error of source term

Figure 2.2: Adjoint solutions, approximated source term, and corresponding error distribution in the adjoint problem.

ADJOINT-BASED ERROR ESTIMATION FOR THE POISSON PROBLEM

The adjoint correction is computed using two different approximations. As shown in Table 2.1, the first ("Adjoint Correction") uses adjoint solutions while the second ("Correction without adjoint") is the direct calculation without adjoint solutions. The two values compare well with each other, which validates the error estimation using adjoint solutions.

Table 2.1: Comparison of error estimations in a Possion problem solved by 8 elements.

Exact error	Approximated error	Adjoint Correction	Correction without adjoint
-2.14168e-08	-2.14171e-08	2.13345e-08	2.13342e-08

Figure 2.3 shows how the computation of the QoI converges with the increase of the number of elements. Obviously, the accuracy of the approximation is drastically improved when the adjoint correction is added to the QoI estimation. This improvement is also noted from the comparison of approximated error in Figure 2.4, where the error with

adjoint correction is referred to as "errWithCorr" whereas the one without correction is referred to as "errNonCorr".

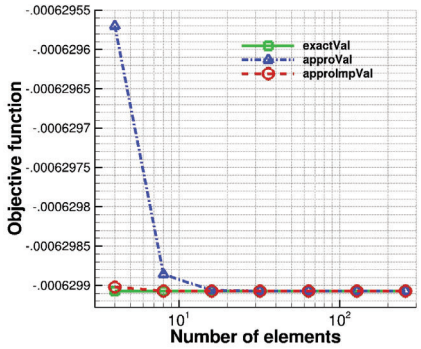


Figure 2.3: The approximation of the QoI.

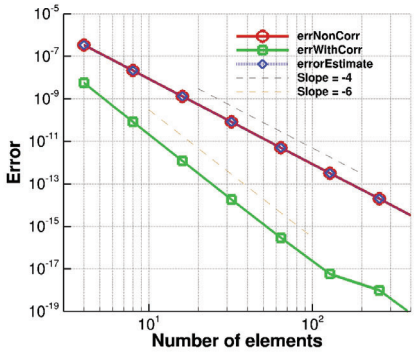


Figure 2.4: Exact and estimated errors of the QoI.

The rate of convergence of the approximation error is -4 since the cubic spline can capture the high-order term in the flow equation whose solutions are polynomial, albeit the equations are solved by linear basis functions. The corrected error is superconvergent, with a slope of -6. Actually, this arises from the computation of the associated adjoint correction term and remaining error, which are presented in Figure 2.5 with the slope of -4 and -6, respectively.

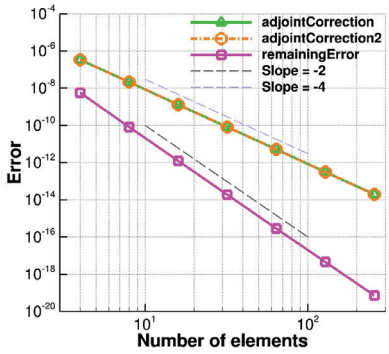


Figure 2.5: The convergence of adjoint correction and remaining error.

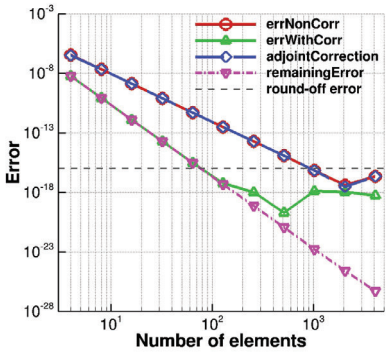


Figure 2.6: Comparison of error estimations with the influence of round-off errors.

The influence of round-off errors on the error estimation can be seen in Figure 2.6. After the remaining error is below the round-off error, there exists a slight difference between the estimated error and exact approximation error, whereas the remaining error still decreases with the same rate of convergence. In this case, the exact solution is used to estimate the remaining error in a finite space as well as to evaluate the accuracy of

the error estimation. However, it is impossible to know the exact solution in practical problems. Instead, it is common to use high-resolution flow solutions to represent the exact values, as discussed in the next section.

2.5.2. ADJOINT-BASED ERROR ESTIMATION IN CLASSICAL FEM

The steady linear advection-diffusion problem considered here is governed by

$$\lambda \nabla u - \nu \Delta u = f\,, \tag{2.53}$$

where $\lambda = 1, \nu = 0.1, x \in [0,1]$. A manufactured solution, $u(x) = \sin(\pi x)$, is used in this problem so that we can analyze the accuracy of error estimation. The source term of the primal flow problem is given as follows

$$f = \lambda\pi\cos(\pi x) + \nu\pi^2 \sin(\pi x)\,. \tag{2.54}$$

Using the *adjoint identity*, $(v, Lu) = (L^* v, u)$, the corresponding adjoint problem is

$$-\lambda \nabla v - \nu \Delta v = g\,, \tag{2.55}$$

where $g = \sin(\pi x)$ is selected in this case, and the homogeneous boundary conditions are applied. The classical FEM, which uses linear basis functions, is employed to solve the primal and adjoint problems in this case. We verified the computing accuracy of the approximation of a QoI, adjoint correction and remaining error using the analytic solution as the mesh is uniformly refined. The adjoint-based error estimation was discussed afterward.

APPROXIMATION OF THE QOI

Discrete solutions are used to make an estimate of the QoI, which can be improved using the adjoint correction. Figure 2.7 shows the approximated QoI compared to the corrected one. The approximation without correction is over-estimated compared to the exact one but becomes convergent after using 64 elements. In contrast, the adjoint-based corrected approximation is underestimated and converges much more efficiently (with 8 elements) than the one without correction.

ANALYTICAL DEMONSTRATION OF THE RATE OF CONVERGENCE

The rate of convergence is determined by the error of the primal and adjoint solutions. The rate of convergence of adjoint correction, $(v_h, f_h - f)$, satisfies the following theorem.

Theorem 1. *If the primal residual error is $O(h^p)$, which means $f_h - f = O(h^p)$, the rate of convergence of the adjoint correction term is $-p$.*

Proof. The adjoint correction is expressed as follows:

$$(v_h, f_h - f) = \int_\Omega v_h O(h^p)\, d\Omega \approx \text{Const} * O(h^p)\,. \tag{2.56}$$

Therefore, when the mesh size is halved, the adjoint correction term is

$$(v_{h/2}, f_{h/2} - f) \approx \text{Const} * \left(\frac{h}{2}\right)^p\,. \tag{2.57}$$

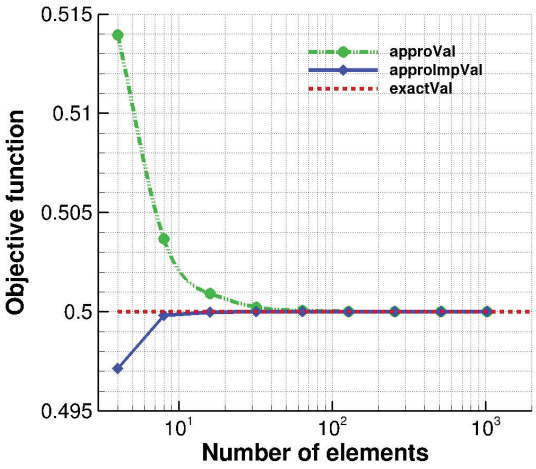


Figure 2.7: Approximation of a QoI in a linear advection-diffusion problem as the mesh is uniformly refined. "approxVal" denotes approximated value while "approxImpVal" represents the corrected approximation.

Considering these two expressions, the rate of convergence of the adjoint correction is obtained as

$$\begin{aligned} \text{Rate of convergence} &= \frac{\log(v_{h/2}, f_{h/2} - f) - \log(v_h, f_h - f)}{lg(2N) - lg(N)} \\ &\approx \frac{\log\left(\text{Const} * (\frac{h}{2})^p\right) - \log\left(\text{Const} * (h^p)\right)}{\log 2}) \\ &= -p\,, \end{aligned} \tag{2.58}$$

where N and $2N$ are the number of mesh elements before and after mesh refinement. □

The same rule can be used to analyze the remaining error, $(g_h - g, u_h - u)$. If the source term g_h involves up to m-order derivatives of v_h, the residual of the source term in the adjoint problem is m degree less than the residual of the adjoint solution. This is to say that $g_h - g = O(h^{p-m})$. Therefore, the remaining error is $O(h^{2p-m})$ and the corresponding rate of convergence in log-log coordinates is $-(2p-m)$, which is super-convergent when the adjoint correction is added to the approximated QoI.

For this advection-diffusion problem ($m = 2$), linear basis functions are employed to solve the primal and adjoint problem so that $u_h - u = O(h^2), v_h - v = O(h^2)$, i.e. $p = 2$. The rate of convergence of the *adjoint correction* and *remaining error* is thus -2 and -4. The numerical results, shown in Figure 2.8, agree with this analysis. Figure 2.9 shows the convergence history of the exact and remaining errors. The superimposed dashed lines have slopes of -2 and -4, respectively. It is observed that the exact error is second order while the remaining error is fourth order. A similar trend has been highlighted by Pierce et al. [51].

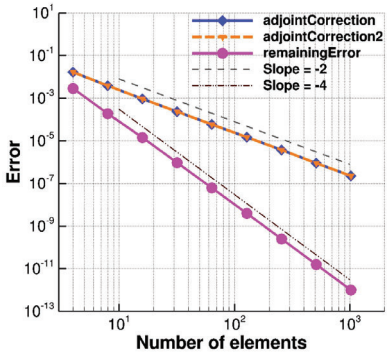


Figure 2.8: Convergence history of adjoint correction and remaining error.

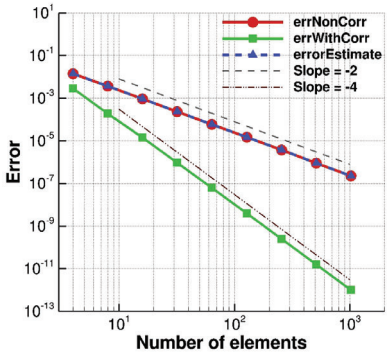


Figure 2.9: Comparison between exact and remaining errors of a QoI.

DISCUSSION ON ADJOINT-BASED ERROR ESTIMATION

The error estimate computed by the adjoint problem is compared with results obtained by mesh doubling. Figure 2.10 shows the variation of QoI's errors computed using uniformly-refined meshes as well as using adjoint solutions. The values obtained using adjoint solutions are more accurate than the ones from the refined mesh. Taking the mesh with 16 elements for example, the accuracy of doubling the number of elements is improved by about 0.6 orders of the original value, whereas the improvement using adjoint solutions is around two orders. Therefore adjoint-based error estimation is much more efficient and accurate (approximately 1.5 orders higher) than using a uniformly-refined mesh in this case. For multidimensional problems, the advantage is more distinct as the cost of flow solutions rises significantly. On the other hand, this benefit becomes more significant with the increase of mesh cells.

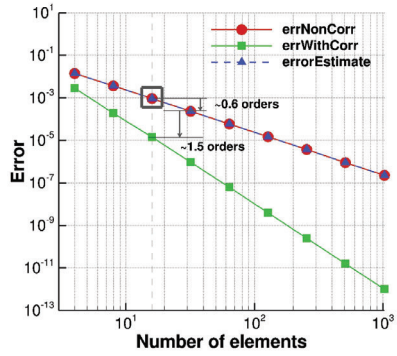


Figure 2.10: Adjoint-based error estimation with comparison to uniform mesh refinement.

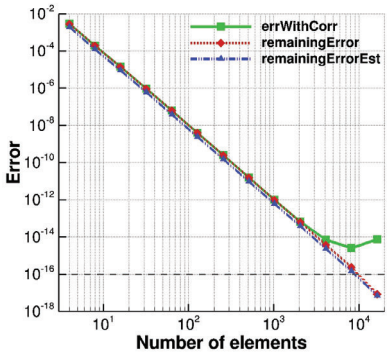


Figure 2.11: Remaining error estimation by solutions with different resolutions.

For this linear advection-diffusion problem, the analytical solution of u is known and

used to compute the reaming error precisely. Generally, we do not know the exact values of flow solution u. Thus the remaining error needs to be estimated, such as by using a higher-order interpolation based on the discrete solution u_h. We estimate this value using linear and cubic flow solutions (see Figure 2.11). In these results, the prediction of remaining error by solutions with various resolutions captures a trend similar to the actual one. Table 2.2 gives the corresponding numerical data. In this case, 60% – 80% of the actual remaining error is acquired by the estimating method. In summary, the main part of the remaining error is obtained by this approach.

Table 2.2: Comparison between the actual and estimated remaining error on uniformly-refined meshes.

Number of elements	Actual remaining Error	Estimated remaining Error	Percentage
4	2.842120E-03	1.943790E-03	68.39%
8	1.891390E-04	1.232690E-04	65.17%
16	1.386140E-05	8.720330E-06	62.91%
32	9.465190E-07	5.857840E-07	61.89%
64	6.201620E-08	3.808310E-08	61.41%
128	3.971920E-09	2.429910E-09	61.18%
256	2.513540E-10	1.534870E-10	61.06%
512	1.580860E-11	9.644550E-12	61.01%
1024	9.758380E-13	6.044150E-13	61.94%
2048	6.203980E-14	3.782700E-14	60.97%
4096	3.873140E-15	2.363390E-15	61.02%
8192	2.349410E-16	1.454740E-16	61.92%
16384	8.524250E-18	6.775300E-18	79.48%

2.5.3. VMM-DRIVEN ERROR ESTIMATION

The aforementioned results were obtained with the classical FEM. Here, four types of error estimation based on the VMM are studied in a convection-dominant case and their results are compared with that of the classical FEM. The parameters of this linear advection-diffusion problem are selected as $\lambda = 1, \nu = 0.025, f = 1, x \in [0,1]$ with homogeneous boundary conditions. The Péclét number is $\mathrm{Pe} = \lambda l/\nu = 40$, where $l = 1$ is the reference length. The analytical solution is

$$u(x) = \frac{1}{\lambda}\left(x - \frac{1 - e^{\mathrm{Pe}\,x}}{1 - e^{\mathrm{Pe}}}\right) \tag{2.59}$$

The QoI is given by $J(u) = (g, u)$, where $g = \sin(\pi x)$.

ERROR ESTIMATION USING THE CLASSICAL FEM

The adjoint-based error estimation is evaluated using primal and adjoint solutions solved by the classical FEM. Figure 2.12 illustrates the adjoint correction and remaining error term in this framework. These terms give a good prediction of the error of the approximated QoI with or without correction, respectively. However, the error estimation ex-

hibits different characteristics when the number of elements is less than 16. The reason is that the advection-dominant problem produces a sharp layer on the right domain boundary and the discrete flow solutions are not accurate over this region (see Figure 2.13). As a result, the adjoint correction term cannot improve the QoI value as shown

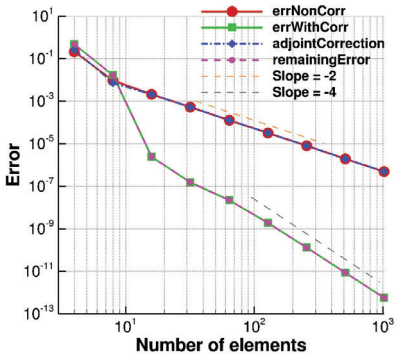


Figure 2.12: Adjoint correction and remaining error in classical FEM.

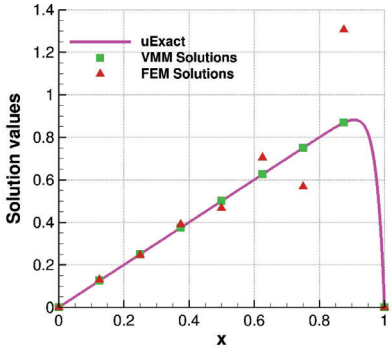
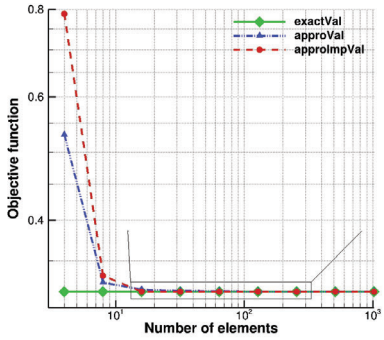
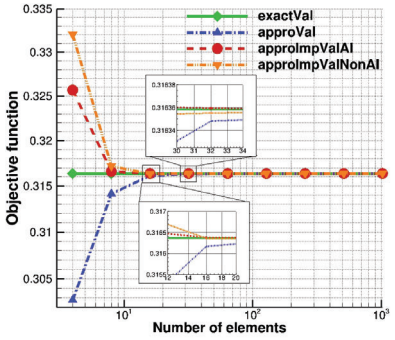


Figure 2.13: Flow solutions solved by 8 elements in FEM and VMM.

in Figure 2.14(a). Conversely, the adjoint solution can improve the approximation of the QoI when the deviation of flow solutions is not severe over a computing domain.



(a) Classical FEM



(b) VMM

Figure 2.14: Approximation of the QoI in the linear advection-diffusion problem with sharp flow gradients solved by classical FEM and VMM.

ERROR ESTIMATION USING IN VMM

Figure 2.14(b) shows that the QoI's approximation was improved significantly using the VMM compared with the computations by classical FEM. The corrected approximation is more efficient than the non-corrected one in the VMM. When the adjoint identity is applied to compute the adjoint correction (the red-circle dashed line), the prediction

of the QoI becomes more accurate than the result without this operation (the orange dashed line).

Four error estimations developed in the VMM are compared in Figure 2.15. $\widetilde{\epsilon}_{vmmd,1}$ and $\widetilde{\epsilon}_{vmmd,4}$ present a similar trend for the estimated error while $\widetilde{\epsilon}_{vmmd,2}$ and $\widetilde{\epsilon}_{vmmd,3}$ agree well with the each other. $\widetilde{\epsilon}_{vmmd,2}$ and $\widetilde{\epsilon}_{vmmd,3}$ capture the changes of the actual error ("errNonCorr") when the resolution of the discrete solutions is high, whereas the other two underestimate the error. In fact, the main reason for this difference is the calculation of the adjoint correction term which is the primary component of the error estimation. The adjoint correction before applying the adjoint identity underestimates the value since there is a remaining component involving $\hat{u}$ which is neglected. This missing part is included after employing the adjoint identity to the adjoint correction. Figure 2.16(a) depicts this improvement. Furthermore, the estimations of remaining

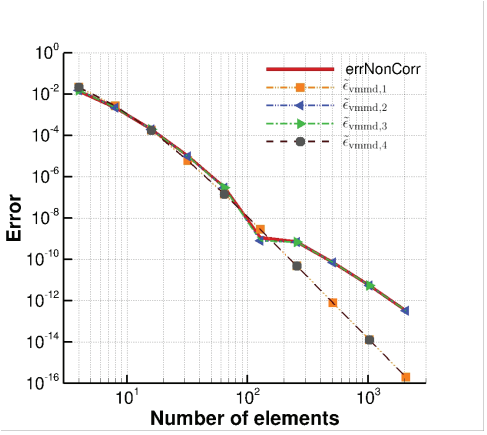
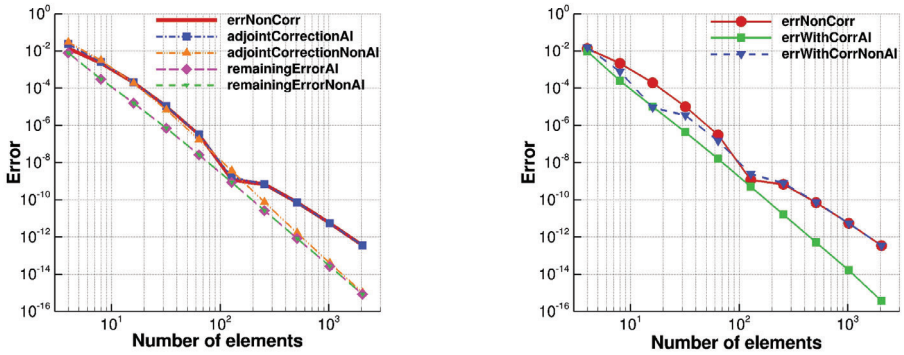


Figure 2.15: Comparison of four error estimations using the VMM for the linear advection-diffusion problem with sharp flow gradients.

error before and after applying adjoint identity have a good agreement for this one-dimensional linear case. When we examine the corresponding error as shown in Figure 2.16(b), the error without using adjoint identity ("errWithCorrNonAI") is higher than the one using adjoint identity ("errWithCorrAI"). The finer the computational mesh, the larger the difference between them.

The rates of convergences using classical FEM and VMM are compared side-by-side in Figure 2.17. Here, $\widetilde{\epsilon}_{vmmd,2}$ is used to predict the error estimate in the VMM. When the adjoint correction is not considered to improve the approximation of the QoI, the VMM-driven error estimate ("VMM errNonCorr") converges at the slope of -4 with the uniform mesh refinement while the one using classical FEM ("Classical FEM errNonCorr") shows a slope of -2. In contrast, if the adjoint correction is used for improving the QoI's computation, the error estimate in classical FEM reaches a slope of -4 and the slope of error estimation in VMM is ever higher, particularly on fine meshes. Overall, using the adjoint method can substantially improve the accuracy of approximating the QoI both in the VMM and classical FEM. The VMM-based error estimation is more efficient than the one from classical FEM, both for the results with and without adjoint correction. Fur-



(a) Adjoint correction and remaining error

(b) Error estimations

Figure 2.16: Computations of adjoint correction, remaining error, and error estimations with and without adjoint identity in the VMM for the linear advection-diffusion problem with sharp flow gradients.

thermore, the VMM-based error estimation is super-convergent compared to that from the classical FEM.

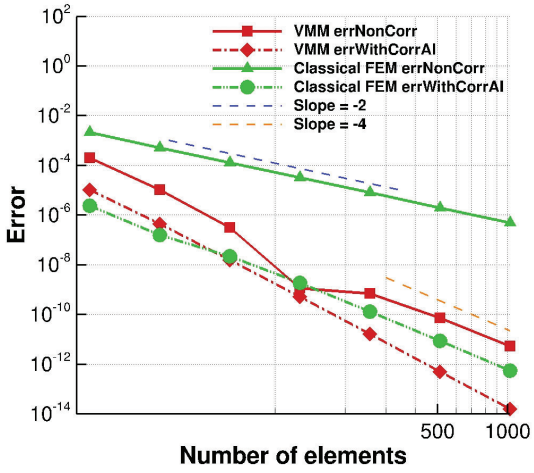


Figure 2.17: The comparison of error estimations between classical FEM and VMM for the linear advection-diffusion problem with sharp flow gradients.

2.6. SUMMARY

This chapter presents the study of adjoint-based error estimation for linear advection-diffusion problems with classical FEM and VMM. Through this study, we draw several

conclusions that will be useful for the subsequent study as follows.

- The adjoint solution can provide a correction term to improve the approximated QoI both in classical FEM and VMM. Flow solutions with low and high resolutions can capture the main part of the remaining error.
- Computing a QoI with adjoint-based correction is more accurate and efficient than the one with uniform mesh refinement under similar computing cost. It is anticipated that this benefit will be more obvious in multi-dimensional problems since the increase of computation on solving NS equations is non-linear and this cost increases dramatically in two/three-dimensional problems.
- The *adjoint identity* introduces the boundary integral to error estimation when we reformulate the relations by partial integration. A crucial aspect of adjoint-based error estimation is its need to match the continuous requirement of first derivatives across element interfaces if the boundary jump term is unknown in FEM.
- The rate of convergence of the adjoint correction and the remaining error in a classical FEM framework are $-p$ and $-(2p-m)$, if the source term f_h and g_h involve up to m-order derivative of u_h and v_h, respectively, and the flow and adjoint solutions are obtained by a p-order scheme, i.e. $u_h - u = O(h^p)$ and $v_h - v = O(h^p)$. On the other hand, the adjoint correction and remaining error are super-convergent when using the VMM.
- Four types of adjoint-based error estimation are proposed for the simulations based on the VMM. The error estimates after applying the adjoint identity, $\widetilde{\epsilon}_{vmmd,2}, \widetilde{\epsilon}_{vmmd,3}$, are effective for computing the approximation error of the QoI and these VMM-based error estimates are more efficient than the traditional technique obtained with classical FEM.

3

PARALLEL ENHANCED ONLINE ALGORITHM FOR BUILDING RORS

Parts of this chapter have been published in Computers and Mathematics with Applications **126**, (2022) [114].

High-fidelity simulations produce high-dimensional data, which is especially challenging to store for solving the adjoint problem backwards in time. To this end, we introduce a primal solution order-reduction technique. An enhanced online algorithm is proposed to build a Reduced-Order Representation (ROR) of the full-order primal solution, which enables the flow solution to be accessed efficiently when the adjoint problem is solved. This algorithm is based on the incremental Singular Value Decomposition developed for POD analysis on the fly. We examine the serial and parallel performance of the algorithm for large-scale data sets here.

3

3.1. INTRODUCTION

Modal analysis has been an essential tool for understanding complex physical features and unsteady nonlinear phenomena governed by partial differential equations (PDEs). It is especially useful for large-scale systems since it can reveal underlying mechanisms by extracting dominant flow structures [78]. One modal analysis technique is the Proper Orthogonal Decomposition (POD) [79, 80], also known as the Karhunen-Loeve procedure or Principal Component Analysis (PCA). It has been widely used to find an optimal combination of basis functions that effectively represent the original full-order system. Such bases can be used to build a reduced-order model (ROM) for high-dimensional full-order data sets [115], which is useful for many applications, such as fluid-structure interactions [73, 74], optimization [75, 116], uncertainty quantification [76, 117], and optimal control [77, 118, 119]. With the development of supercomputers and the improvement of experimental techniques, we can now obtain high-fidelity solutions for complicated flow phenomena and potentially provide insights into unrevealed mechanisms. But these high-dimensional solutions pose a challenge for the computation of a POD.

Common approaches to compute the POD for a given dataset ($\boldsymbol{X} \in \mathbb{R}^{n\times m}$) have been discussed in Refs. [78, 120], which includes eigenvalue decomposition, the method of snapshots, and singular value decomposition (SVD). The POD was introduced to fluid dynamics by Lumey [121] using an eigenvalue decomposition of the covariance matrix (i.e. $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^{\top}$). When the number of degrees of freedom (DoF) (viz. the number of rows) is larger than the number of snapshots (the number of columns), the method of snapshots [122] can be used to solve the eigenvalue decomposition on an easy-solved matrix $\boldsymbol{C} = \boldsymbol{X}^{\top}\boldsymbol{X}$ and then POD modes can be computed with the snapshot solutions (see Section 3.2.2). Due to the relation of the POD with the eigenvalue decomposition, SVD [123] can also be applied on $\boldsymbol{X}$ directly to find the POD modes, which has been shown to be more robust against round-off error [78]. These different approaches need the complete snapshot matrix before performing the POD. The chosen approach depends on the different structure of data sets, as shown in Figure 3.1. Typical high-fidelity data, such as that from large-eddy simulations, are of high dimension in terms of DoF and snapshots, making it prohibitively expensive for these offline approaches (see Figure 3.1).

In order to address this issue, researchers have developed alternative methods, such as recursive POD [124, 125], randomized SVD [126, 127], and incremental SVD [86, 87]. The incremental SVD (iSVD) considered here can be used for high-dimensional POD analysis online, without storing the data. The iSVD is initialized with a small given data

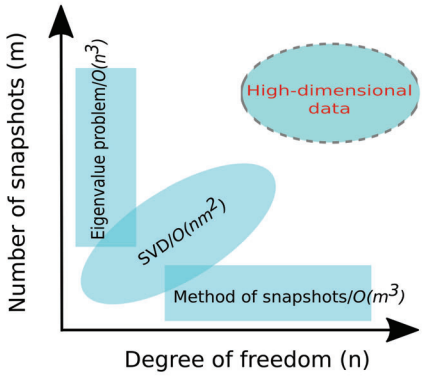


Figure 3.1: Common approaches for solving POD with applications on different structures of data sets, and the necessity of developing effective algorithms for high-dimensional data.

set with a known SVD and then updates the SVD in real-time during the simulation as new snapshots become available. The incremental SVD was proposed by Brand [86] for the incomplete data sets in computer vision and audio feature extraction. Fareed et al. [88] extended the iSVD algorithm with a weighted inner product and gave an error analysis [128] with a computed error bound for its approximation. Although the iSVD is capable of performing POD analysis online, the cumulative cost over all snapshots, $\mathcal{O}(nm^3)$, can be higher than that of offline approaches [87].

This limitation can be overcome by reducing the size of a snapshot matrix or improving the algorithm. The former can be achieved by the adaptive selection of snapshots [89, 90], which removes redundant information and thus reduces the data size. An example of the latter is to perform a low-rank incremental SVD with a prescribed number of POD modes. This has been used to efficiently solve unsteady adjoint equations [92, 129] in nonlinear problems, although the selection of truncation number has been *ad hoc* or heuristic. Phalippou et al. [91] proposed a new incremental SVD algorithm using an error estimator, which incorporates the lost information of skipped snapshots and missed singular values, which allows online truncation of the POD modes and the on-the-fly selection of the independent snapshots. However, the number of POD modes computed can rise significantly as the necessary snapshots are added.

Given that, in practice, a low-rank representation usually only needs a handful of POD modes, we propose truncating POD modes online as the incremental SVD proceeds, which we refer to as the enhanced iSVD. We also develop an aggregated expression for the complete solution snapshots and construct two estimators for evaluating the energy captured by the low-order approximation. These enable us to determine the number of POD modes necessary based on cumulative energy.

The parallel POD has been studied using domain decomposition techniques [130, 131], and thus it is natural to consider the parallelization algorithm for incremental SVD [132, 133] to make it more efficient and competitive in practical applications. Choi et al. [134] employed the incremental SVD for reduced-order models in space and time,

and studied large-scale particle transport simulations with the parallel incremental SVD. Therefore, we also investigate the parallel performance of the proposed enhanced online algorithm.

The remainder of this chapter is organized as follows. In Section 3.2, we present the methodologies for building PODs offline and online, including an overview of three basic offline POD approaches, the standard incremental SVD for online PODs, and the enhanced online algorithm and its parallelization implementation adapted from [133]. The enhanced online algorithm is further discussed in Section 3.3, with the development of an aggregated expression for the snapshot matrix and the establishment of a priori estimators. Serial numerical experiments are used to demonstrate the impacts of the enhanced process in this section. The parallel performance of the enhanced online algorithm is then investigated with high-dimensional data sets generated from an unsteady one-dimensional Burgers problem in Section 3.4. Finally, we conclude the chapter with Section 3.5.

3.2. METHODOLOGIES

There are different ways to achieve a POD analysis in literature, including offline POD and online POD. The offline POD requires access to the entire set of flow solution snapshots. However, finite computational memory can constrain the use of offline approaches for practical problems. In contrast, the online (or incremental) approach considers each snapshot solution one by one and builds the POD on the fly, making it useful for realistic large-scale problems. We will present the details of the offline and online POD approaches in Section 3.2.2 and Section 3.2.3. An enhanced incremental SVD (which is referred to as enhanced online algorithm) is proposed to improve the efficiency of the online POD in Section 3.2.4, followed by the design of parallelization for this enhanced online POD in Section 3.2.5.

3.2.1. DEFINITION OF POD

The goal of POD analysis is to identify an optimal combination of basis vectors for representing the original full-order dataset. Given a dataset including all snapshots $\boldsymbol{u}^{(i)}(x) \in \mathcal{V}(\Omega), i = 1,2,\cdots,N_\mathrm{t}, x \in \Omega$, we seek to find a set of functions $\boldsymbol{\varphi}(x) \in \mathcal{V}$ to represent the dataset so that the mean square projection on all snapshots is maximized, i.e.

$$\max_{\boldsymbol{\varphi}\in\mathcal{V}} \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \frac{|\langle \boldsymbol{u}^{(i)}, \boldsymbol{\varphi}\rangle|^2}{\langle \boldsymbol{\varphi}, \boldsymbol{\varphi}\rangle}, \tag{3.1}$$

where $\langle\cdot,\cdot\rangle$ denotes a inner product defined on Ω, and $\langle \boldsymbol{u}^{(i)}, \boldsymbol{\varphi}\rangle$ describes a projection of the solution snapshot $\boldsymbol{u}^{(i)}$ onto a function $\boldsymbol{\varphi}$. This maximization problem is equivalent to the following eigenvalue problem [77, 79, 121]

$$\boldsymbol{K}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}, \tag{3.2}$$

where $\boldsymbol{\phi}$ denotes the optimal function and

$$\boldsymbol{K}\boldsymbol{\phi} = \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \boldsymbol{u}^{(i)} \langle \boldsymbol{u}^{(i)}, \boldsymbol{\phi}\rangle = \frac{1}{N_\mathrm{t}} \sum_{i=1}^{N_\mathrm{t}} \int_\Omega \boldsymbol{u}^{(i)}(x)\, \boldsymbol{u}^{(i)}(x')\, \boldsymbol{\phi}(x')\, \mathrm{d}x'. \tag{3.3}$$

It is noted that $\boldsymbol{K}$ is expressed in a continuous function over Ω. For the discrete solutions $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, the problem Equation (3.2) can be interpreted as finding the eigenvalue λ_j and eigenvectors ϕ_j from

$$\boldsymbol{R}\boldsymbol{\phi}_j = \lambda_j \boldsymbol{\phi}_j\,, \tag{3.4}$$

where $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{n\times n}$ [1] is the covariance matrix with the inner product defined as $\langle \boldsymbol{a}, \boldsymbol{b}\rangle = \boldsymbol{a}^\top \boldsymbol{b}$.

3.2.2. OFFLINE POD

There are three different approaches to computing the POD offline after gathering complete snapshots into a solution matrix. By solving the eigenvalue problem in Equation (3.4), we can obtain the POD modes $\boldsymbol{\phi}^{(j)}$, where $j = 1, 2, \cdots, \min(n, m)$. For practical applications, the number of DoF of the computational mesh (n) can be larger than the number of snapshots (m), leading to a huge covariance matrix that is difficult to solve directly. The method of snapshots [122] was proposed to overcome this difficulty by instead finding the eigenvalue decomposition of $\boldsymbol{C} = \boldsymbol{X}^\top \boldsymbol{X}$. Its eigenvalues and eigenvectors are λ_j and $\boldsymbol{\psi}^{(j)}$, where $j = 1, 2, \cdots, m$. The POD modes are then computed as follows

$$\boldsymbol{\phi}^{(j)} = \boldsymbol{X}\boldsymbol{\psi}^{(j)} / \sqrt{\lambda_j}\,. \tag{3.5}$$

The eigenvalue decomposition of $\boldsymbol{X}\boldsymbol{X}^\top$ can also be related to the SVD of $\boldsymbol{X}$. In the following, the SVD will be done using the LAPACK library [78], which gives $\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}^\top$, where $\boldsymbol{\Sigma}, \boldsymbol{V}$ and $\boldsymbol{W}$ are the singular value matrix, left and right singular vector matrix, respectively. Each column of $\boldsymbol{V}$ denotes one POD mode $\phi^{(j)}$, and the coefficients $\alpha_j(t_i), i = 1, 2, \cdots, m$ are determined by $\boldsymbol{\Sigma}$ and $\boldsymbol{W}$. The resulting offline POD is utilized as a reference for comparison with online PODs introduced in the next sections. Based on the POD analysis, the reduced-order solution $\tilde{u}(x, t_i)$ is calculated by

$$\tilde{u}(x, t_i) = \bar{u}(x) + \sum_{j=1}^{M} \alpha_j(t_i)\,\phi^{(j)}(x)\,, \tag{3.6}$$

where $\phi^{(j)}$, $j = 1, 2, \cdots, M$ denote the selected POD modes, and M is usually smaller than $\min(n, m)$. $\bar{u}$ represents the mean value.

3.2.3. STANDARD INCREMENTAL SVD FOR ONLINE POD

The core updating step of the incremental SVD is demonstrated in Proposition 1 before explaining the standard and enhanced incremental SVD algorithm.

Proposition 1. *Suppose we have a dense matrix* $\mathbf{U}$ *with a known SVD expressed as* $\mathbf{U} = \mathbf{V}\Sigma\mathbf{W}^\top$, *where* $\Sigma \in \mathbb{R}^{k\times k}, \mathbf{V} \in \mathbb{R}^{n\times k}$ *and* $\mathbf{W} \in \mathbb{R}^{k\times k}$ *are a singular value matrix, left and right singular vector matrix, respectively. When a new column* $c \in \mathbb{R}^{n\times 1}$ *is added to formulate a new updated* $\mathbf{U}_u = [\mathbf{U}\ c]$, *a bordered-diagonal sparse matrix* $\mathbf{Q}$ *is formulated as*

$$\mathbf{Q} = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}, \tag{3.7}$$

[1] Strictly speaking, $\boldsymbol{R}$ should be expressed as $\boldsymbol{X}\boldsymbol{X}^\top / N_t$, but $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^\top$ has been widely used in literature for convenience since its eigenvectors are the same. The influence of N_t is thus lumped into the magnitude of eigenvalues. (also referred to see [78])

where $d = \mathbf{V}^\top c, h = c - \mathbf{V}d$, $p = \sqrt{h^\top h}$. *After applying SVD on* $\mathbf{Q}$ *as* $\mathbf{Q} = \mathbf{V}_Q \Sigma_Q \mathbf{W}_Q^\top$ *, we can have a SVD of the new updated matrix* $\mathbf{U}_u$ *as*

$$\mathbf{U}_u = \mathbf{V}_u \, \Sigma_Q \, \mathbf{W}_u^\top, \tag{3.8}$$

where Σ_Q, $\mathbf{V}_u$ *and* $\mathbf{W}_u$ *are the new singular value matrix, left and right singular vector matrix, respectively, given as*

$$\mathbf{V}_u = [\mathbf{V}\ j]\,\mathbf{V}_Q\,, \quad \mathbf{W}_u = \begin{bmatrix} \mathbf{W} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_Q\,, \tag{3.9}$$

where $j = h/p$.

Proof. Considering a new column c, we can project it onto a space spanned by $\mathbf{V}$ as $d = \mathbf{V}^\top c$. Then the orthogonal component of c is given as $h = c - \mathbf{V}d = pj$, where $p = \sqrt{h^\top h}$ and $j = h/p$. c can be re-expressed as a sum of the orthogonal and projected components as $c = pj + \mathbf{V}d$. This leads to

$$\mathbf{U}_u = [\mathbf{U}\ c] = [\mathbf{V}\Sigma\mathbf{W}^\top\ pj + \mathbf{V}d] = [\mathbf{V}\ j] \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix} \begin{bmatrix} \mathbf{W} & 0 \\ 0 & 1 \end{bmatrix}^\top = [\mathbf{V}\ j]\,\mathbf{Q} \begin{bmatrix} \mathbf{W} & 0 \\ 0 & 1 \end{bmatrix}^\top. \tag{3.10}$$

Apparently, $\mathbf{Q}$ is a bordered-diagonal sparse matrix with only the last column fully filled as shown in Figure 3.2.

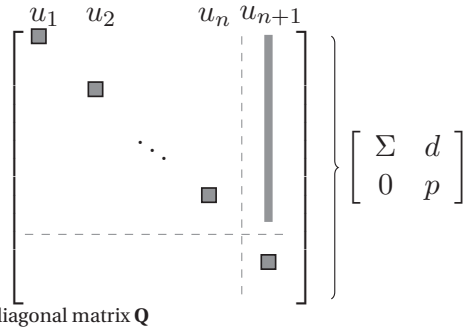


Figure 3.2: The bordered-diagonal matrix $\mathbf{Q}$

We can easily obtain $\mathbf{Q}$'s SVD as $\mathbf{Q} = \mathbf{V}_Q \Sigma_Q \mathbf{W}_Q^\top$, resulting in

$$\mathbf{U}_u = [\mathbf{V}\ j]\,\mathbf{V}_Q\,\Sigma_Q \left(\begin{bmatrix} \mathbf{W} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_Q \right)^\top = \mathbf{V}'_u\,\Sigma_Q\,\mathbf{W}'^{\top}_u\,, \tag{3.11}$$

where $\mathbf{V}'_u = [\mathbf{V}\ j]\,\mathbf{V}_Q\,, \mathbf{W}'_u = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_Q$. Here, a decomposition of $\mathbf{U}_u$ has been formulated with a diagonal matrix, Σ_Q. This decomposition will be an SVD only if $\mathbf{V}'_u$ and $\mathbf{W}'_u$ are orthogonal matrices. For $\mathbf{V}'_u$, we have

$$\mathbf{V}'^{\top}_u \mathbf{V}'_u = \mathbf{V}_Q^\top \begin{bmatrix} \mathbf{V}^\top \\ j^\top \end{bmatrix} [\mathbf{V}\ j]\,\mathbf{V}_Q = \mathbf{V}_Q^\top \begin{bmatrix} \mathbf{V}^\top\mathbf{V} & \mathbf{V}^\top j \\ j^\top\mathbf{V} & j^\top j \end{bmatrix} \mathbf{V}_Q\,. \tag{3.12}$$

Since $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$, the $\mathbf{V}^\top j$ is computed as

$$\mathbf{V}^\top j = \mathbf{V}^\top \frac{c - \mathbf{V}d}{p} = \frac{1}{p}(\mathbf{V}^\top c - \mathbf{V}^\top\mathbf{V}d) = \frac{1}{p}(d - d) = \vec{0}\,. \tag{3.13}$$

Thus, we have $j^\top\mathbf{V} = (\mathbf{V}^\top j)^\top = \vec{0}^\top$. $j^\top j$ is determined as

$$j^\top j = \left(\frac{h}{p}\right)^\top \frac{h}{p} = \frac{h^\top h}{p^2} = 1\,. \tag{3.14}$$

Since $\mathbf{V}_Q^\top\mathbf{V}_Q = \mathbf{I}$, we have $\mathbf{V}_u'^\top\mathbf{V}_u' = \mathbf{I}$, i.e. $\mathbf{V}_u = \mathbf{V}_u'$ is an orthogonal matrix. Similarly, due to $\mathbf{W}^\top\mathbf{W} = \mathbf{I}$ and $\mathbf{W}_Q^\top\mathbf{W}_Q = \mathbf{I}$, we can prove $\mathbf{W}_u'$ is also an orthogonal matrix as

$$\mathbf{W}_u'^\top\mathbf{W}_u' = \mathbf{W}_Q^\top \begin{bmatrix} \mathbf{W}^\top & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_Q = \mathbf{W}_Q^\top \begin{bmatrix} \mathbf{W}^\top\mathbf{W} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_Q = \mathbf{I}\,. \tag{3.15}$$

Consequently, Proposition 1 is confirmed. □

Based on the above, the standard incremental algorithm is formulated as shown in Algorithm 2. After the projection of $\boldsymbol{c}$ onto $\boldsymbol{V}$, the matrix $\boldsymbol{Q}$ is formulated, where small projections less than a threshold of **tol** [88] are neglected to avoid the effects of round-off errors. A standard SVD then is applied to $\boldsymbol{Q}$ before the updating process. It is then decided if the added column will increase the rank of the updated matrix (lines 8-13). Here the subscript denotes the index of row and column starting from 1 ($V_{Q_{(1:k,1:k)}}$ is a sub-matrix of V_Q with first k-th rows and columns, for instance). The truncation of small singular values less than a prescribed threshold $\mathbf{tol}_{\text{sv}}$ is used to improve efficiency without affecting the accuracy of the updated modes. Finally, the updated modes are re-orthogonalized by the modified Gram-Schmidt process if non-orthogonality occurs among them, which improves the robustness of the algorithm. Hence, the standard incremental SVD is applied on all snapshots to build POD online after its initialization with

$$\boldsymbol{\Sigma} = \|\boldsymbol{c}_0\|_2\,, \boldsymbol{V} = \boldsymbol{c}/\boldsymbol{\Sigma}_{(1,1)}\,, \boldsymbol{W} = [1]\,, \tag{3.16}$$

where $\boldsymbol{c}_0$ denotes the first snapshot solution.

3.2.4. ENHANCED INCREMENTAL SVD FOR ONLINE POD

Albeit the standard incremental SVD can achieve online POD analysis, the total computational cost is more expensive than the offline method. In practice, however, the number of POD modes (M) necessary for building an accurate reduced-order model can be far smaller than the number of DoF (n) and snapshots (m). Thus, to improve the computing efficiency, we introduce the truncation of a selected number of POD modes into the incremental SVD algorithm as shown in Algorithm 3 between line 14 and line 16, leading to an enhanced incremental SVD. This is referred to as enhanced online algorithm (EOA) in the following content. This truncation can remarkably reduce the computing cost of incremental SVD. This improvement comes with a sacrifice in the accuracy of POD analysis due to the influence of neglected high-order modes. However, this sacrifice is often within a reasonable range and it can be controlled by adding more POD modes during the incremental process. This will be demonstrated by the following numerical results.

Algorithm 2 Standard incremental SVD for building POD

Input: $V \in \mathbb{R}^{n\times k}, \Sigma \in \mathbb{R}^{k\times k}, W \in \mathbb{R}^{k\times k}, c \in \mathbb{R}^{n\times 1}$, **tol**, $\mathbf{tol}_{\text{sv}}$
Output: V, Σ, W

$k = \text{nColumns}(V)$
$d = V^\top c, \quad p = (|(c - Vd)^\top (c - Vd)|)^{1/2}$ ▷ **Part 1: Projection**
if $p <$ **tol** **then**
    $Q = \begin{bmatrix} \Sigma & d \\ 0 & 0 \end{bmatrix}$
else
    $Q = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}$
▷ **Part 2: SVD solution**
$V_Q, \Sigma_Q, W_Q = \text{SVD}\ \ (Q)$
▷ **Part 3: LSV update**
if ($p <$ **tol**) OR ($k \geq n$) **then**
    $V = V V_{Q(1:k,1:k)}, \quad \Sigma = \Sigma_{Q(1:k,1:k)}, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_{Q(1:k+1,1:k)}$
else
    $j = (c - Vd)/p$
    $V = [V\ j] V_Q, \quad \Sigma = \Sigma_Q, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q$
    $k = k + 1$
▷ **Part 5: Small SV truncation**
if ($\Sigma_{(k-1,k-1)} >$ $\mathbf{tol}_{\text{sv}}$) AND ($\Sigma_{(k,k)} <$ $\mathbf{tol}_{\text{sv}}$) **then**
    $k = k - 1$
    $\Sigma = \Sigma_{(1:k,1:k)}, \quad V = V_{(:,1,k)}, \quad W = W_{(:,1:k)}$
▷ **Part 6: Reorthogonalization**
if $|V_{(:,k)}^\top V_{(:,1)}| > \min(\mathbf{tol}, \varepsilon \times n)$ **then** ▷ ε is double-precision machine epsilon
    $V = \text{ModifiedGramSchmidt}(V)$

3.2.5. PARALLEL DESIGN OF ENHANCED ONLINE ALGORITHM

There exist two different frameworks to parallelize the iSVD in literature. Iwen et al. [132] proposed a hierarchical approach to computing local iSVD in each processor followed by a global agglomerative iSVD for all processors. This avoids data communication during local operations but does not complete the POD analysis across all processors at each incremental step. The computing cost of global operations increases when increasing the number of processors. Alternatively, one may note that the incremental SVD involves multiplications between vectors and matrices, and thus their parallelization can be used to improve computing performance. Arrighi et al. [133] developed a framework for the POD, *libROM*, using parallel operations on matrices and vectors. This approach gives a global POD analysis at each incremental step. We develop our method based on this open-source library, originally intended for standard incremental SVD, and improve it by introducing the enhanced process using the same parallel data structure.

In *libROM*, both vectors and matrices are stored in a distributed way so that the SVD

Algorithm 3 Enhanced incremental SVD for building POD

Input: $V \in \mathbb{R}^{n\times k}, \Sigma \in \mathbb{R}^{k\times k}, W \in \mathbb{R}^{k\times k}, c \in \mathbb{R}^{n\times 1}, \mathbf{tol}, \mathbf{tol}_{\text{sv}}$
Output: V, Σ, W

$k = \text{nColumns}(V)$
$d = V^\top c, \quad p = (|(c - Vd)^\top (c - Vd)|)^{1/2}$ ▷ **Part 1: Projection**
if $p < \mathbf{tol}$ **then**
    $Q = \begin{bmatrix} \Sigma & d \\ 0 & 0 \end{bmatrix}$
else
    $Q = \begin{bmatrix} \Sigma & d \\ 0 & p \end{bmatrix}$
▷ **Part 2: SVD solution**
$V_Q, \Sigma_Q, W_Q = \text{SVD} \quad (Q)$
▷ **Part 3: LSV update**
if $(p < \mathbf{tol})$ OR $(k \geq n)$ **then**
    $V = V V_{Q(1:k,1:k)}, \quad \Sigma = \Sigma_{Q(1:k,1:k)}, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_{Q(1:k+1,1:k)}$
else
    $j = (c - Vd)/p$
    $V = [V\, j] V_Q, \quad \Sigma = \Sigma_Q, \quad W = \begin{bmatrix} W & 0 \\ 0 & 1 \end{bmatrix} W_Q$
    $k = k + 1$
▷ **Part 4: Enhanced process**
if $(k > M)$ **then**
    $\Sigma = \Sigma_{(1:M,1:M)}, \quad V = V_{(:,1:M)}, \quad W = W_{(:,1:M)}$
    $k = M$
▷ **Part 5: Small SV truncation**
if $(\Sigma_{(k-1,k-1)} > \mathbf{tol}_{\text{sv}})$ AND $(\Sigma_{(k,k)} < \mathbf{tol}_{\text{sv}})$ **then**
    $k = k - 1$
    $\Sigma = \Sigma_{(1:k,1:k)}, \quad V = V_{(:,1,k)}, \quad W = W_{(:,1:k)}$
▷ **Part 6: Reorthogonalization**
if $|V_{(:,k)}^\top V_{(:,1)}| > \min(\mathbf{tol}, \varepsilon \times n)$ **then** ▷ ε is double-precision machine epsilon
    $V = \text{ModifiedGramSchmidt}(V)$

of Q can be computed locally. They are distributed to n_p processors equally or unequally for parallel computational operations, as illustrated in Figure 3.3. A vector is split into n_p parts stored in n_p processors, respectively. The matrix is stored in a similar way but for each column. This ensures that one part of the computational mesh will be operated by only one processor. Different types of computations between matrices and vectors are summarized in Table 3.1, where a local operation represents an operation with data that is undistributed or stored on every processor, whereas a global operation denotes an operation with distributed data. The parallel enhanced online algorithm developed here is based upon such distributed computations.

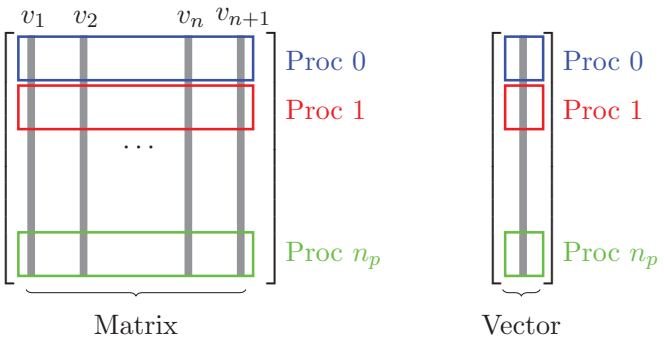


Figure 3.3: Parallelization design of matrices and vectors in incremental SVD

Table 3.1: Parallelization operations in incremental SVD

Different types of data	Local operations	Global operations
Vector$^\intercal$ · Vector	undis* · undis → undis	dis† · dis → undis
Matrix · Vector	undis · undis → undis	dis · undis → dis
Matrix$^\intercal$ · Vector	undis · undis → undis	dis · dis → undis
Matrix · Matrix	undis · undis → undis	dis · undis → dis
Matrix$^\intercal$ · Matrix	undis · undis → undis	dis · dis → undis

* denotes the undistributed data.
† denotes the distributed data.

3.3. PERFORMANCE OF THE ENHANCED ALGORITHM IN SERIAL

The number of selected POD modes (M) is an important parameter as it determines the accuracy of reconstructed low-order solutions. There are different methods to determine this number, including heuristic and statistical methods [135]. A common way is to quantify the energy captured by selected POD modes, namely the cumulative energy e_M,

$$e_M = \frac{\sum_{i=1}^{M} \sigma_i^2}{\sum_{i=1}^{\min(m,n)} \sigma_i^2}, \tag{3.17}$$

where σ_i denotes the singular values. e_M, typically larger than 90% [120], provides a way to evaluate whether the POD can rationally represent the full-order solutions. Specific thresholds of the cumulative energy captured, such as 99%, have been shown effective for representing the original system in literature [74, 77, 118, 119].

An aggregated expression of the solution matrix is formulated for the enhanced online algorithm, by which the solution matrix is divided into the reconstructed and truncated components. Using this expression, we establish two estimators for the lower bound of the cumulative energy during the incremental process. We present this aggregated expression in Section 3.3.1 and the lower-bound estimators in Section 3.3.2,

followed by the investigation of their performance using an unsteady Burgers problem under serial computations in Section 3.3.3.

3.3.1. AN AGGREGATED EXPRESSION FOR THE ENHANCED ALGORITHM

In this section, we analyze the influence of the enhanced process on the incremental SVD, focusing on the accuracy of reconstructed solutions based on selected POD modes. Assuming that we select M POD modes as the reduced basis functions, the reconstructed solution is expressed as $\widetilde{\boldsymbol{U}_M} = \boldsymbol{V}_M \boldsymbol{\Sigma}_M \boldsymbol{W}_M^{\mathsf{T}}$, where $\boldsymbol{V}_M, \boldsymbol{\Sigma}_M, \boldsymbol{W}_M$ are the first M-truncated matrices of $\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{W}$, respectively. We can express the solution data matrix at any k-th incremental step as follows

$$\boldsymbol{U}_k = \begin{cases} \widetilde{\boldsymbol{U}_k} & k \leqslant M \\ \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}_k^{'} & k = M+1 \\ \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}_k^{'} + \sum_{i=M+1}^{k-1} \boldsymbol{U}_i^{'(k)} & k \geqslant M+2 \end{cases}, \tag{3.18}$$

where $\boldsymbol{U}_k, k = 1, 2, \cdots, n$ is a sub-matrix of the entire solution matrix $\boldsymbol{X}$ defined by the first k columns. $\widetilde{\boldsymbol{U}_k} = \widetilde{\boldsymbol{V}}_k \widetilde{\boldsymbol{\Sigma}}_k \widetilde{\boldsymbol{W}}_k^{\mathsf{T}}$ denotes the reconstructed solution after the k-th incremental step, defined as

$$\widetilde{\boldsymbol{V}}_k = \begin{cases} \boldsymbol{V}^{(k)} \\ \boldsymbol{V}^{(k)}_{(:,1:M)} \end{cases} \quad \widetilde{\boldsymbol{\Sigma}}_k = \begin{cases} \boldsymbol{\Sigma}^{(k)} \\ \boldsymbol{\Sigma}^{(k)}_{(1:M,1:M)} \end{cases} \quad \widetilde{\boldsymbol{W}}_k = \begin{cases} \boldsymbol{W}^{(k)} & , k \leqslant M \\ \boldsymbol{W}^{(k)}_{(:,1:M)} & , k \geqslant M+1 \end{cases}, \tag{3.19}$$

where $\widetilde{\boldsymbol{\Sigma}}_k$ and $\widetilde{\boldsymbol{V}}_k, \widetilde{\boldsymbol{W}}_k$ denote the sigular value matrix and singular vector matrices. $\boldsymbol{U}_k^{'} = \boldsymbol{V}^{(k)}_{(:,M+1)} \boldsymbol{\Sigma}^{(k)}_{(M+1,M+1)} {\boldsymbol{W}^{(k)}_{(:,M+1)}}^{\mathsf{T}}$ is the truncated solution at the k-th incremental step, and it is a zero matrix when $k \leqslant M$. $\boldsymbol{U}_i^{'(k)} (k \geqslant M+2, k > i \geqslant M+1)$ represents solutions truncated at the i-th step but extended to the k-th step, and it is defined by

$$\boldsymbol{U}_i^{'(k)} = [\boldsymbol{U}_i^{'} \quad \underbrace{\vec{\boldsymbol{0}} \quad \cdots \quad \vec{\boldsymbol{0}}}_{k-i}]. \tag{3.20}$$

This implies

$$\begin{aligned} \boldsymbol{U}_k^{'(k+1)} &= [\boldsymbol{U}_k^{'} \quad \vec{\boldsymbol{0}}], \quad k \geqslant M+1 \\ \boldsymbol{U}_i^{'(k+1)} &= [\boldsymbol{U}_i^{'(k)} \quad \vec{\boldsymbol{0}}], \quad k \geqslant M+2, k > i \geqslant M+1. \end{aligned} \tag{3.21}$$

The first and second expressions in Equation (3.18) require the same operations as the standard incremental SVD, while the truncated solution $\boldsymbol{U}_{M+1}^{'}$ exists in the second expression. The third expression is the scenario we usually meet in practice, which is proved as follows.

Proof. When $k \leqslant M$, the expression is identical to the standard incremental SVD. This is to say that $\boldsymbol{U}_k = \boldsymbol{V}^{(k)} \boldsymbol{\Sigma}^{(k)} \boldsymbol{W}^{(k)\mathsf{T}} = \widetilde{\boldsymbol{U}_k}$. When $k = M+1$, we have $\boldsymbol{U}_{M+1} = [\boldsymbol{U}_M \ \boldsymbol{c}_{M+1}] = [\widetilde{\boldsymbol{U}_M} \ \boldsymbol{c}_{M+1}]$. The SVD is utilized to obtain

$$\begin{aligned} \boldsymbol{U}_{M+1} &= \boldsymbol{V}^{(M+1)} \boldsymbol{\Sigma}^{(M+1)} \boldsymbol{W}^{(M+1)\mathsf{T}} \\ &= \boldsymbol{V}^{(M+1)}_{(:,1:M)} \boldsymbol{\Sigma}^{(M+1)}_{(1:M,1:M)} {\boldsymbol{W}^{(k)}_{(:,1:M)}}^{\mathsf{T}} + \boldsymbol{V}^{(M+1)}_{(:,M+1)} \boldsymbol{\Sigma}^{(M+1)}_{(M+1,M+1)} {\boldsymbol{W}^{(k)}_{(:,M+1)}}^{\mathsf{T}} \quad \triangleright \text{Additivity} \\ &:= \widetilde{\boldsymbol{V}}_{M+1} \widetilde{\boldsymbol{\Sigma}}_{M+1} \widetilde{\boldsymbol{W}}_{M+1}^{\mathsf{T}} + \boldsymbol{v}_{M+1}^{(M+1)} \boldsymbol{\sigma}_{M+1}^{(M+1)} \boldsymbol{w}_{M+1}^{(M+1)\mathsf{T}} \quad \triangleright \text{Defined} \\ &:= \widetilde{\boldsymbol{U}_{M+1}} + \boldsymbol{U}_{M+1}^{'}, \end{aligned} \tag{3.22}$$

which gives the second expression. It is noted that $\boldsymbol{U}'_{M+1}$ will be truncated since we only consider M POD modes. The third expression is proved using mathematical induction as follows.

Base case: When $k = M+2$, the solution data is expressed as

$$\begin{aligned}\boldsymbol{U}_{M+2} &= [\boldsymbol{U}_{M+1} \quad \boldsymbol{c}_{M+2}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} + \boldsymbol{U}'_{M+1} \quad \boldsymbol{c}_{M+2}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] + [\boldsymbol{U}'_{M+1} \quad \vec{\boldsymbol{0}}] \\ &= [\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] + \boldsymbol{U}'^{(M+2)}_{M+1},\end{aligned} \tag{3.23}$$

We apply the SVD on $[\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}]$, resulting in

$$\begin{aligned}[\widetilde{\boldsymbol{U}_{M+1}} \quad \boldsymbol{c}_{M+2}] &= \boldsymbol{V}^{(M+2)}\boldsymbol{\Sigma}^{(M+2)}\boldsymbol{W}^{(M+2)\mathsf{T}} \\ &= \boldsymbol{V}^{(M+2)}_{(:,1:M)}\boldsymbol{\Sigma}^{(M+2)}_{(1:M,1:M)}\boldsymbol{W}^{(M+2)\,\mathsf{T}}_{(:,1:M)} + \boldsymbol{V}^{(M+2)}_{(:,M+1)}\boldsymbol{\Sigma}^{(M+2)}_{(M+1,M+1)}\boldsymbol{W}^{(M+2)\,\mathsf{T}}_{(:,M+1)} \\ &:= \widetilde{\boldsymbol{V}}_{M+2}\widetilde{\boldsymbol{\Sigma}}_{M+2}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{M+2} + \boldsymbol{v}^{(M+2)}_{M+1}\boldsymbol{\sigma}^{(M+2)}_{M+1}\boldsymbol{w}^{(M+2)\mathsf{T}}_{M+1} \\ &:= \widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}'_{M+2},\end{aligned} \tag{3.24}$$

where $\widetilde{\boldsymbol{U}_{M+2}} = \widetilde{\boldsymbol{V}}_{M+2}\widetilde{\boldsymbol{\Sigma}}_{M+2}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{M+2}$ and $\boldsymbol{U}'_{M+2} = \boldsymbol{v}^{(M+2)}_{M+1}\boldsymbol{\sigma}^{(M+2)}_{M+1}\boldsymbol{w}^{(M+2)\mathsf{T}}_{M+1}$. Consequently, the solution is stated as $\boldsymbol{U}_{M+1} = \widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}'_{M+2} + \boldsymbol{U}'^{(M+2)}_{M+1}$, i.e. $\boldsymbol{U}_{M+2}$ is true.
Inductive step: Assume that the induction hypothesis holds for a particular $k \geqslant M+2$, meaning

$$\boldsymbol{U}_k = \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}'_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}'^{(k)}_i. \tag{3.25}$$

When $n = k+1$, the updated solution matrix is expressed as

$$\begin{aligned}\boldsymbol{U}_{k+1} = [\boldsymbol{U}_k \quad c_{k+1}] &= [\widetilde{\boldsymbol{U}_k} + \boldsymbol{U}'_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}'^{(k)}_i \quad c_{k+1}] \\ &= [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + [\boldsymbol{U}'_k \quad \vec{\boldsymbol{0}}] + [\sum_{i=M+1}^{k-1} \boldsymbol{U}'^{(k)}_i \quad \vec{\boldsymbol{0}}] \\ &= [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + \boldsymbol{U}'^{(k+1)}_k + \sum_{i=M+1}^{k-1} \boldsymbol{U}'^{(k+1)}_i = [\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] + \sum_{i=M+1}^{k} \boldsymbol{U}'^{(k+1)}_i,\end{aligned} \tag{3.26}$$

We apply the SVD on $[\widetilde{\boldsymbol{U}_k} \quad c_{k+1}]$ and express it as follows

$$\begin{aligned}[\widetilde{\boldsymbol{U}_k} \quad c_{k+1}] &= \boldsymbol{V}^{(k+1)}\boldsymbol{\Sigma}^{(k+1)}\boldsymbol{W}^{(k+1)\mathsf{T}} \\ &= \boldsymbol{V}^{(k+1)}_{(:,1:M)}\boldsymbol{\Sigma}^{(k+1)}_{(1:M,1:M)}\boldsymbol{W}^{(k+1)\,\mathsf{T}}_{(:,1:M)} + \boldsymbol{V}^{(k+1)}_{(:,M+1)}\boldsymbol{\Sigma}^{(k+1)}_{(M+1,M+1)}\boldsymbol{W}^{(k+1)\,\mathsf{T}}_{(:,M+1)} \\ &:= \widetilde{\boldsymbol{V}}_{k+1}\widetilde{\boldsymbol{\Sigma}}_{k+1}\widetilde{\boldsymbol{W}}^{\mathsf{T}}_{k+1} + \boldsymbol{v}^{(k+1)}_{M+1}\boldsymbol{\sigma}^{(k+1)}_{M+1}\boldsymbol{w}^{(k+1)\mathsf{T}}_{M+1} \\ &:= \widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}'_{k+1},\end{aligned} \tag{3.27}$$

where $\widetilde{\boldsymbol{U}_{k+1}} = \widetilde{\boldsymbol{V}}_{k+1}\widetilde{\boldsymbol{\Sigma}}_{k+1}\widetilde{\boldsymbol{W}}_{k+1}^{\mathsf{T}}$ and $\boldsymbol{U}'_{k+1} = \boldsymbol{\nu}_{M+1}^{(k+1)}\boldsymbol{\sigma}_{M+1}^{(k+1)}\boldsymbol{w}_{M+1}^{(k+1)\mathsf{T}}$. Therefore, we deduce that

$$\boldsymbol{U}_{k+1} = \widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}'_{k+1} + \sum_{i=M+1}^{k} \boldsymbol{U}_i^{'(k+1)}. \tag{3.28}$$

In other words, the statement $\boldsymbol{U}_{k+1}$ also holds true, and we establish the inductive step.
Conclusion: As we have proved the base case and the inductive step, the original statement $\boldsymbol{U}_n$ holds for every natural number $n \geqslant M+2$. □

3.3.2. LOWER-BOUND ESTIMATORS OF CUMULATIVE ENERGY

By virtue of the above-mentioned aggregated expression, we establish two lower-bound estimators for the cumulative energy after applying the enhanced process. Since we can evaluate the summation of singular values by the Frobenius norm (F-norm) of matrices, i.e. $\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(\boldsymbol{A})}$, the energy ratio of the M-selected POD modes is computed as

$$e_M = \frac{\sum_{i=1}^{M} \sigma_i^2}{\sum_{i=1}^{\min(m,n)} \sigma_i^2} = \frac{\|\widetilde{\boldsymbol{U}}\|_{\mathrm{F}}^2}{\|\boldsymbol{U}\|_{\mathrm{F}}^2}. \tag{3.29}$$

Then these two lower-bound estimators at the k-th incremental step are given as

$$e_{\mathrm{con}}^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\left(\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} + F_1^{(k)}\right)^2} \tag{3.30}$$

and

$$e_{\mathrm{simp}}^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_2^{(k)}}, \tag{3.31}$$

where $\widehat{\boldsymbol{U}_k} = \widetilde{\boldsymbol{U}_k} + \boldsymbol{U}_k^{'}$, and thus $\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 = \|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}_k^{'}\|_{\mathrm{F}}^2$ due to the orthogonality of POD modes at k-th incremental step. $F_1^{(k)}$ and $F_2^{(k)}$ are given by

$$F_1^{(k)} = \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}} = F_1^{(k-1)} + \|\boldsymbol{U}'_{k-1}\|_{\mathrm{F}},\ k \geqslant M+2 \tag{3.32}$$

$$F_2^{(k)} = \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}}^2 = F_2^{(k-1)} + \|\boldsymbol{U}_{k-1}^{'}\|_{\mathrm{F}}^2,\ k \geqslant M+2, \tag{3.33}$$

where $F_1^{(M+1)} = 0$ and $F_2^{(M+1)} = 0$. These are only dependent on the solution truncated in previous steps and are utilized to evaluate the error at the next step. Using these two formulas enables us to avoid storing all $\|\boldsymbol{U}_j^{'(k)}\|_{\mathrm{F}}, i = M+1, \cdots, k-1$, and have a recursive approach to compute the $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$.

e_{con}^k is derived by the Cauchy–Schwarz inequality of the Frobenius inner product, viz.

$\langle \boldsymbol{A}, \boldsymbol{B}\rangle_{\mathrm{F}} \leqslant \|\boldsymbol{A}\|_{\mathrm{F}} \|\boldsymbol{B}\|_{\mathrm{F}}$. We can evaluate $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ as

$$\begin{aligned}
\|\boldsymbol{U}_k\|_{\mathrm{F}}^2 &= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \|\sum_{i=M+1}^{k-1} \boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}}^2 + 2\sum_{i=M+1}^{k-1} \langle \widehat{\boldsymbol{U}_k}, \boldsymbol{U}_i^{'(k)}\rangle_{\mathrm{F}} \\
&= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1}\sum_{j=M+1}^{k-1} \langle \boldsymbol{U}_i^{'(k)}, \boldsymbol{U}_j^{'(k)}\rangle_{\mathrm{F}} + 2\sum_{i=M+1}^{k-1} \langle \widehat{\boldsymbol{U}_k}, \boldsymbol{U}_i^{'(k)}\rangle_{\mathrm{F}} \\
&\leqslant \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1}\sum_{j=M+1}^{k-1} \|\boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}} \|\boldsymbol{U}_j^{'(k)}\|_{\mathrm{F}} + 2\sum_{i=M+1}^{k-1} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} \|\boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}} \\
&= \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_1^{(k)} F_1^{(k)} + 2F_1^{(k)} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} = \left(\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}} + F_1^{(k)}\right)^2 .
\end{aligned} \tag{3.34}$$

Then the first lower-bound estimator of e_M^k is derived as

$$e_M^k = \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\boldsymbol{U}_k\|_{\mathrm{F}}^2} \geqslant \frac{\|\widetilde{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2 + F_1^{(k)} F_1^{(k)} + 2F_1^{(k)} \|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}} := e_{\mathrm{con}}^k . \tag{3.35}$$

This estimator can be conservative in practice because the way chosen to bound $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ can produce a larger value. Therefore, this estimator provides a conservative evaluation of the cumulative energy. However, this feature can guarantee that all dominant modes are included using the enhanced incremental SVD. If we use this estimator to determine the number of POD modes, it will recover the standard incremental SVD when the value of the estimator is beyond the threshold.

Based on Equation (3.18), however, we can have an accurate computation for the Frobenius norm of $\boldsymbol{U}_k$ as stated in Proposition 2, which will be used to construct the other estimator, e_{simp}^k.

Proposition 2. *Suppose that we have the expression of $\boldsymbol{U}_k$ as shown in Equation (3.18), the Frobenius norm of $\boldsymbol{U}_k$ is given as*

$$\|\boldsymbol{U}_k\|_F^2 = \begin{cases} \|\widetilde{\boldsymbol{U}_k}\|_F^2 & k \leqslant M \\ \|\widetilde{\boldsymbol{U}_k}\|_F^2 + \|\boldsymbol{U}_k^{'}\|_F^2 & k = M+1 \\ \|\widetilde{\boldsymbol{U}_k}\|_F^2 + \|\boldsymbol{U}_k^{'}\|_F^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}_i^{'(k)}\|_F^2 & k \geqslant M+2 \end{cases} , \tag{3.36}$$

Proof. The F-norm for the truncated solutions $\boldsymbol{U}_i^{'(k)}$ is given before the detailed proof. It satisfies the following equation

$$\|\boldsymbol{U}_i^{'(k)}\|_{\mathrm{F}} = \|\boldsymbol{U}_i^{'}\|_{\mathrm{F}}, k \geqslant M+2, i = M+1, \cdots, k-1 . \tag{3.37}$$

This is an inherent property of $\boldsymbol{U}_i^{'(k)}$, which can be verified using Equation (3.20) and the definition of F-norm.

When $k \leqslant M+1$, this expression is satisfied as there is no difference from the standard incremental SVD. We will prove the general case by mathematical induction as follows.

Base case: When $k = M+2$, the solution data is expressed as

$$\begin{aligned}
\|\boldsymbol{U}_{M+2}\|_{\mathrm{F}}^2 &= \|\boldsymbol{U}_{M+1} \quad \boldsymbol{c}_{M+2}\|_{\mathrm{F}}^2 \\
&= \|\boldsymbol{U}_{M+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{M+2}\|_2^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{M+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{M+2}\|_2^2 && \triangleright \text{When } k = M+1 \\
&= \|\widetilde{\boldsymbol{U}_{M+1} \quad \boldsymbol{c}_{M+2}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+1}\|_{\mathrm{F}}^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{M+2}} + \boldsymbol{U}'_{M+2}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+1}\|_{\mathrm{F}}^2 && \triangleright \textit{Equation } (3.24) \\
&= \|\widetilde{\boldsymbol{U}_{M+2}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+2}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+1}\|_{\mathrm{F}}^2 && \triangleright \text{Orthogonality} \\
&= \|\widetilde{\boldsymbol{U}_{M+2}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{M+2}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'^{(M+2)}_{M+1}\|_{\mathrm{F}}^2 && \triangleright \text{Equation } (3.37).
\end{aligned} \tag{3.38}$$

Therefore, $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ is satisfied when k is equal to $M+1$.
Inductive step: Assume that the induction hypothesis holds for a particular $k \geqslant M+2$, viz. $\|\boldsymbol{U}_k\|_{\mathrm{F}}^2$ is expressed as

$$\|\boldsymbol{U}_k\|_{\mathrm{F}}^2 = \|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2, \tag{3.39}$$

when $n = k+1$, we can obtain the F-norm of the solution matrix as

$$\begin{aligned}
\|\boldsymbol{U}_{k+1}\|_{\mathrm{F}}^2 &= \|\boldsymbol{U}_k \quad \boldsymbol{c}_{k+1}\|_{\mathrm{F}}^2 \\
&= \|\boldsymbol{U}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{k+1}\|_2^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2 + \|\boldsymbol{c}_{k+1}\|_2^2 && \triangleright \text{Equation } (3.39) \\
&= \|\widetilde{\boldsymbol{U}}_k \quad \boldsymbol{c}_{k+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2 && \triangleright \text{Definition of } F\text{-norm} \\
&= \|\widetilde{\boldsymbol{U}_{k+1}} + \boldsymbol{U}'_{k+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2 && \triangleright \text{Equation } (3.27) \\
&= \|\widetilde{\boldsymbol{U}_{k+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{k+1}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k-1} \|\boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2 && \triangleright \text{Orthogonality} \\
&= \|\widetilde{\boldsymbol{U}_{k+1}}\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_{k+1}\|_{\mathrm{F}}^2 + \sum_{i=M+1}^{k} \|\boldsymbol{U}'^{(k+1)}_i\|_{\mathrm{F}}^2, && \triangleright \text{Equation } (3.37)
\end{aligned} \tag{3.40}$$

which proves that the statement for $\|\boldsymbol{U}_{k+1}\|_{\mathrm{F}}^2$ is held, thereby establishing the inductive step.

Conclusion: As the base case and the inductive step have been proved, the original statement $\|\boldsymbol{U}_n\|_{\mathrm{F}}^2$ holds for every natural number $n \geqslant M+2$. □

The second estimator is given as

$$e_M^k = \frac{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2}{\|\boldsymbol{U}_k\|_{\mathrm{F}}^2} = \frac{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2}{\|\widetilde{\boldsymbol{U}}_k\|_{\mathrm{F}}^2 + \|\boldsymbol{U}'_k\|_{\mathrm{F}}^2 + \|\sum_{i=M+1}^{k-1} \boldsymbol{U}'^{(k)}_i\|_{\mathrm{F}}^2} = \frac{\|\widehat{\boldsymbol{U}_k}\|_{\mathrm{F}}^2}{\|\widehat{\boldsymbol{U}}_k\|_{\mathrm{F}}^2 + F_2^{(k)}} := e_{\mathrm{simp}}^k. \tag{3.41}$$

It is worth noting that during the incremental process $\widetilde{\boldsymbol{U}_k}$ is a combination of M-selected modes for the complete solution $\boldsymbol{U}_k$ when $k \geqslant M$. The approximation of solution matrix with the first M POD modes is the most optimal combination [82]. In other words, the summation of its first k singular values is larger than or equal to those from any other rank-k approximation, thereby leading to $\sum_{i=1}^{M} \sigma_i^2(\widetilde{\boldsymbol{U}_k}) \leqslant \sum_{i=1}^{M} \sigma_i^2(\boldsymbol{U}_k)$. As a result, this estimator is a lower bound for the cumulative energy captured by standard iSVD.

3.3.3. IMPACTS OF TRUNCATION NUMBER ON THE ENHANCED ALGORITHM

The truncation number of the enhanced process generally can be larger than the number of selected POD modes (M). We refer to this truncation number as M_e in the subsequent text. The gap between M_e and M can compensate for interactions with high-order modes that are not chosen yet, yielding more accurate M-selected POD modes. In this section, we investigate the impact of different M_e for the enhanced process on the lower-bound estimators and POD modes, using the solutions from a one-dimensional (1D) unsteady Burgers problem. Results are presented for serial iSVD and enhanced iSVD computations.

We consider a space-time domain $\Omega : [0,1] \times \mathrm{I} : [0,20]$. The Burgers equation is often used as a mathematical model for applications that involve shock wave propagation in viscous flows or idealized turbulence [136], and it is expressed as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f\,, \tag{3.42}$$

where u is the solution, with boundary conditions $u(0,t) = u(1,t) = 0$ and an initial condition $u(x,0) = 0$. ν, the viscosity coefficient, is chosen as 0.01 here, and $f \in \mathbb{R}$ is a known forcing term as

$$f(x,t) = 1 + \frac{5}{30}\sum_{i=1}^{3} \sin(i\pi t)\sin(i\pi x)\,. \tag{3.43}$$

This forcing term f is introduced to produce a solution with large fluctuations and a boundary layer near the right boundary. The primal problem is discretized by piecewise linear bases in space and solved by the variational multiscale method with 256 elements. We apply a four-stage second-order Runge-Kutta time marching scheme to solve this unsteady problem with a time step of $\Delta t = 0.001$. The time interval of $[0,10]$ is utilized for flow development before a statistical steady state is reached during the time period of $[10,20]$.

The data used is sampled every 20 time steps in a time interval $[10,12]$, generating a solution matrix of $257 * 100$. We applied the enhanced online algorithm to analyze this dataset and compare it with the reference values obtained by the standard iSVD. Figure 3.4 shows the cumulative energy computed based on two estimators, e_{con} and e_{simp}, and the reference value at the terminating step when different truncation numbers (M_e) of the enhanced process are considered. As expected, a great portion of the solution energy is captured as M_e is increased. It is noted that the estimator e_{con} lies far below the actual value when M_e is small but does converge to the actual value for $M_e > 10$. e_{simp} is also lower than the reference value, but it provides a more accurate estimate than e_{con}. In this case, a difference of less than 0.1% is observed for $M = 3$, which is considered

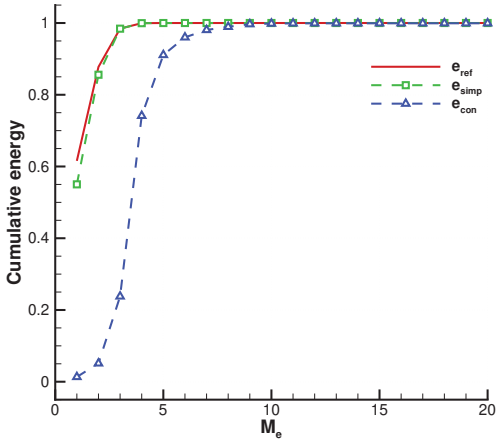
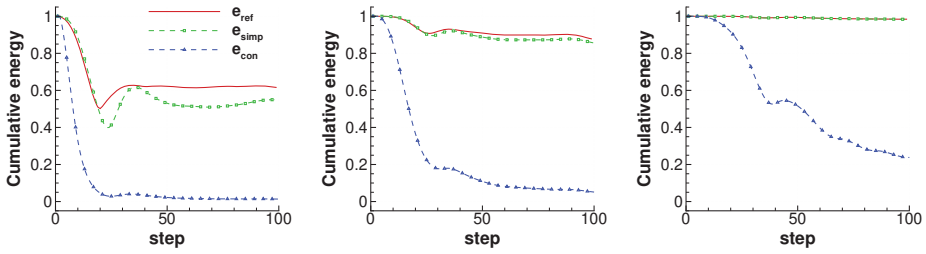


Figure 3.4: Cumulative energy of two estimators, e_{con} and e_{simp} at the terminating step (100 steps), compared with the reference value from the standard incremental SVD, with different numbers (M_e) for the enhanced online process.

sufficiently accurate for practical purposes. Using $M = 4$ captures 99.96% of the energy and reduces the computation run time by a factor of 125 relative to a full iSVD. Figure 3.5 presents the change of cumulative energy for these two estimators during the incremental process when M_e is equal to 1, 2 and 3, respectively. We can observe e_{simp} is more accurate than e_{con} over the whole incremental process.



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

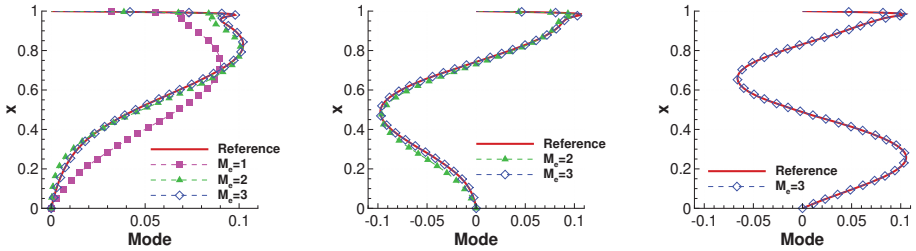
Figure 3.5: Cumulative energy computed by estimators, e_{con} and e_{simp}, and reference value from the standard incremental SVD during the incremental process for first three truncation numbers (M_e) of the enhanced process.

It is noted that $F_2^{(k)}$ can be related to the aforementioned $F_1^{(k)}$ as

$$\left(F_1^{(k)}\right)^2 = F_2^{(k)} + 2\sum_{i=M+1}^{k-1}\sum_{j=i+1}^{k-1} \|\boldsymbol{U}_i'^{(k)}\|_{\text{F}} \|\boldsymbol{U}_j'^{(k)}\|_{\text{F}}. \tag{3.44}$$

Considering the non-negative values of $\|\widehat{\boldsymbol{U}}_k\|_{\mathrm{F}}$ and $F_1^{(k)}$ in Equation (3.30), we have $F_2^{(k)} \leqslant (F_1^{(k)})^2$ and thus e_{simp} is more accurate than e_{con}. Although the performance of the enhanced iSVD depends on the problems considered, these two estimators equip the user of the enhanced iSVD with a priori knowledge of the accuracy of the POD analysis, allowing the computing cost for a desired accuracy to be minimized.

Figure 3.6 shows the comparison of the POD modes computed based on different enhanced iSVDs and standard iSVD. As shown in Figure 3.6(a), the shape of the first POD mode computed from the enhanced algorithm with $M_e = 1$ is significantly different from the reference, while increasing M_e reduces this discrepancy.



(a) First mode (b) Second mode (c) Third mode

Figure 3.6: Comparisons of the first three POD modes computed by enhanced and standard iSVDs, with truncations numbers of $M_e = 1, 2, 3$, respectively.

This phenomenon can be observed for the second and third modes as well. Table 3.2 shows the first three eigenvalues under these three enhanced iSVDs in comparison with the ones from standard iSVD. We can see that increasing M_e allows the POD analysis to capture more kinetic energy, but also improves the accuracy of the dominant modes. $M_e = 3$ is enough to produce an accurate POD analysis for the current problem with a relative eigenvalue error less than 0.1%.

Table 3.2: First three eigenvalues computed by different enhanced iSVDs in relative to the reference from the standard iSVD.

	Reference	Truncation number (M_e)		
		3	2	1
First eigenvalue	27.00840	27.00677	26.54177	24.13557
Second eigenvalue	11.48709	11.48611	10.99621	
Third eigenvalue	4.682270	4.678387		

3.4. NUMERICAL RESULTS IN PARALLEL

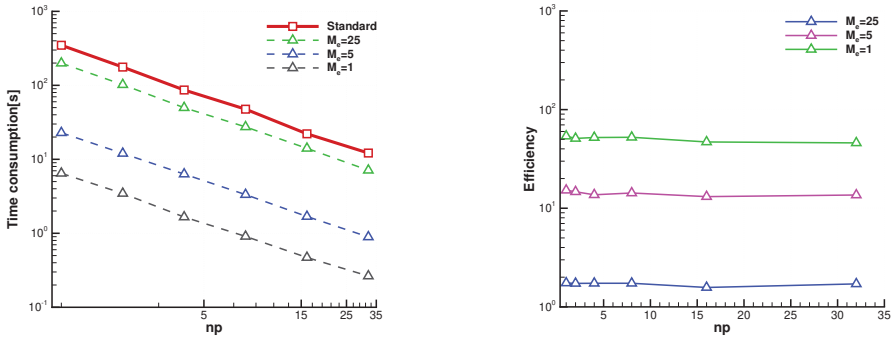
In this section, we investigate the parallel performance of the enhanced online algorithm using both a synthetic matrix and numerical solutions from the 1D unsteady Burger problem. For the latter case, we generate two types of data, a matrix with more DoF ($n > m$) and a matrix with more snapshots ($n < m$), to study the impacts of the enhanced

process on computing expense. All computations were performed on a 32-core node of a Beowulf Linux cluster equipped with AMD Opteron(tm6136) processors and 128 GB of RAM.

The enhanced online algorithm is divided into six parts, as shown on the right side of Algorithm 3, to identify the main contributors to computing cost and to study their parallel performance. These parts consist of projection, SVD solution, LSV (left singular vectors) update, enhanced process, small SV (singular values) truncation, and reorthogonalization. In addition, we define the major parallel operations as the summation of the projection, LSV update and reorthogonalization, which will be studied in the subsequent cases.

3.4.1. ANALYSIS ON A SYNTHETIC MATRIX

A synthetic matrix of 320000 × 50 is formed based on randomized values from a uniform distribution. Strong scaling performance is evaluated by performing the analysis on this matrix using up to 32 cores. Figure 3.7(a) shows the total time consumption for standard and enhanced iSVDs. The computing cost of the standard iSVD is much higher than that of the enhanced one, even up to two orders higher in this case. Although using a very small number of modes can affect the accuracy of POD analysis, the main dynamics can normally be captured by relatively few modes, resulting in considerably lower computing costs. Figure 3.7(b) shows the truncated speedup defined as the ratio of the total time consumption of the standard iSVD to that of the enhanced iSVD. All enhanced iSVDs are able to achieve constant improvement, which indicates the high scalability of the enhanced algorithms.



(a) Computing time (b) Truncated speedup

Figure 3.7: Time consumption of standard incremental SVD and enhanced online algorithm in parallel for a 320000-by-50 synthetic matrix, and the truncated speedups of these enhanced iSVDs. np denotes the number of cores.

The parallel speedups for the total incremental time, major parallel operations, and LSV update are shown in Figure 3.8. Results from a standard iSVD and three enhanced iSVDs are shown. All of them scale well in parallel. For the speedup of the LSV update, the enhanced iSVDs produce scaling performance similar to the standard one, linearly

below the ideal value. This good agreement is observed in the major parallel operations as well, although this is less significant. The speedup of the total incremental time by the enhanced iSVDs increases linearly but is much lower than the standard iSVD as we use more cores. Furthermore, using a smaller number M_e for the enhanced process can degrade the speedup to some extent. This is because the enhanced process reduces the computing cost of the major parallel operations, and thus the proportion of the computing cost from other undistributed operations becomes more and more apparent with increased cores, which is explained in detail below. However, this degradation is not significant compared to the speedup improvement.

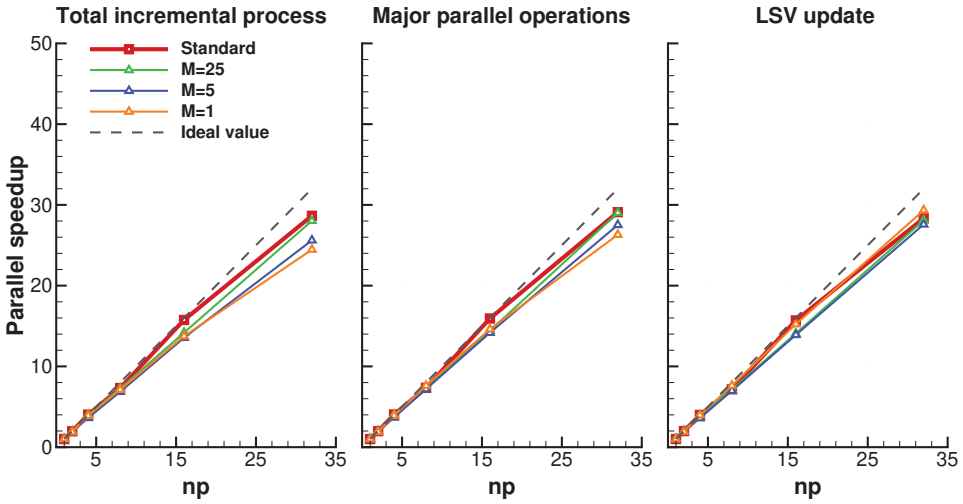


Figure 3.8: The speedup of standard/enhanced parallel incremental SVDs for the total incremental time, major parallel operations and LSV update for a 320000-by-50 synthetic matrix.

The time ratio of the computing cost for different parts is given in Figure 3.9. Among these six parts, the LSV update is the major contributor to the total computing time for both standard and enhanced iSVDs. However, the enhanced algorithm mitigates this contribution. The smaller the number M_e for the enhanced process is, the lower the contribution of the LSV update is to the total cost. The enhanced process reduces the dimension of V_Q and W_Q, and thus multiplications involving these matrices are changed to relatively cheap calculations. For a small M_e, the computing cost of the projection becomes significant. Additionally, the state vectors are generated by randomized values and are independent, leading to good orthogonality of the singular vectors. Thus reorthogonalization is not a significant cost for this problem.

3.4.2. UNSTEADY FLOW PROBLEM

The Burgers problem of Section 3.3.3 is used to generate large-scale matrices for parallel performance studies. We construct two types of matrices, a matrix with more DoF and a matrix with more snapshots. We refer to the former as a deep matrix and the latter as a

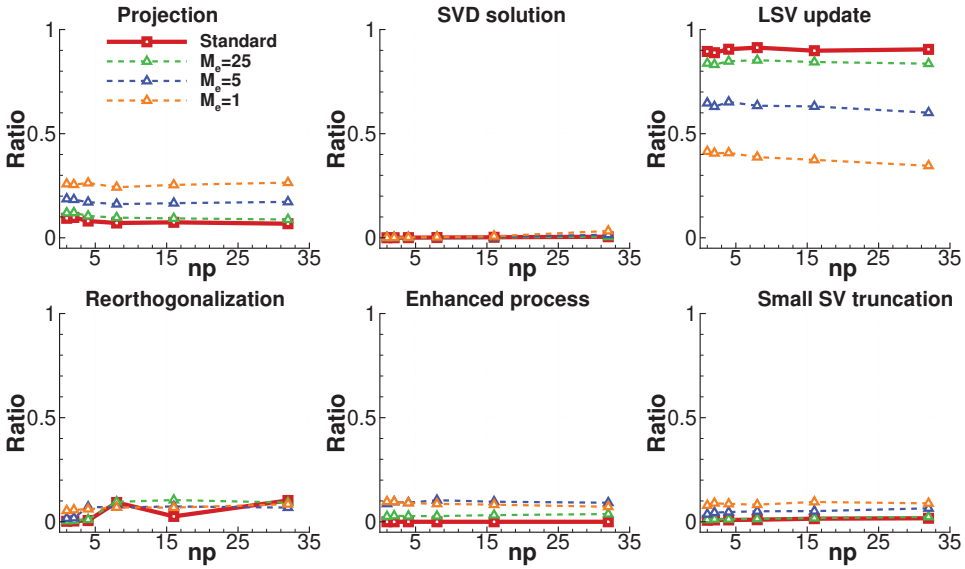


Figure 3.9: The time ratio of computing cost for different parts during the standard/enhanced parallel incremental SVDs, including the projection, SVD solution, LSV update, reorthogonalization, enhanced process, small SV truncation.

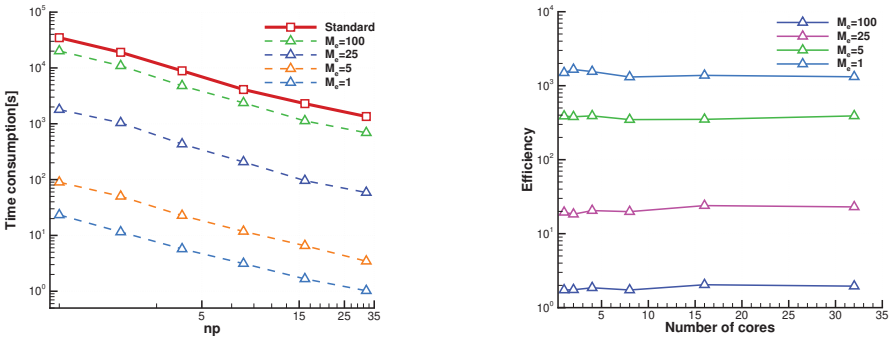
wide matrix. Using interpolations of the original data set, the deep matrix of 327681×200 is used to study a strong scaling performance, in which each core holds at least 10000 data points. The wide matrix (257×2000) is used to perform a weak scaling study.

A DEEP MATRIX WITH MORE DOF

This scenario corresponds to when we solve the problem with a fine mesh but we only need to gather a small number of snapshots for modal analyses in order to study the dominant flow structures. A small number of snapshots may arise, for instance, because linearly dependent solutions are skipped for the analysis, using a selection of snapshots [89, 91].

Figure 3.10 shows the computing time of the enhanced and standard iSVDs for this case and the speedup of the enhanced online algorithm. We can observe that the computing cost of both standard and enhanced iSVDs is constantly reduced when using more cores. The enhanced algorithm is capable of improving the computational efficiency at a stable rate when multiple cores are used.

Figure 3.11 shows the parallel speedup of computing cost for the total incremental process, major parallel operations and LSV update for the standard incremental SVD and enhanced online algorithm. For the total incremental time, a good strong-scaling performance is observed, occasionally outperforming the ideal value. The performance for major parallel operations behaves in a similar vein, although the parallel speedup of the LSV update is well below the ideal value. The pseudo hyper-speedup results from the computational cost of the reorthogonalization. Figure 3.12(a) presents the number of reorthogonalizations versus the number of cores in-



(a) Computing time (b) Truncated speedup

Figure 3.10: Time consumption of standard incremental SVD and enhanced online algorithm in parallel, and the truncated speedup of the enhanced process for a 327681-by-200 deep matrix.

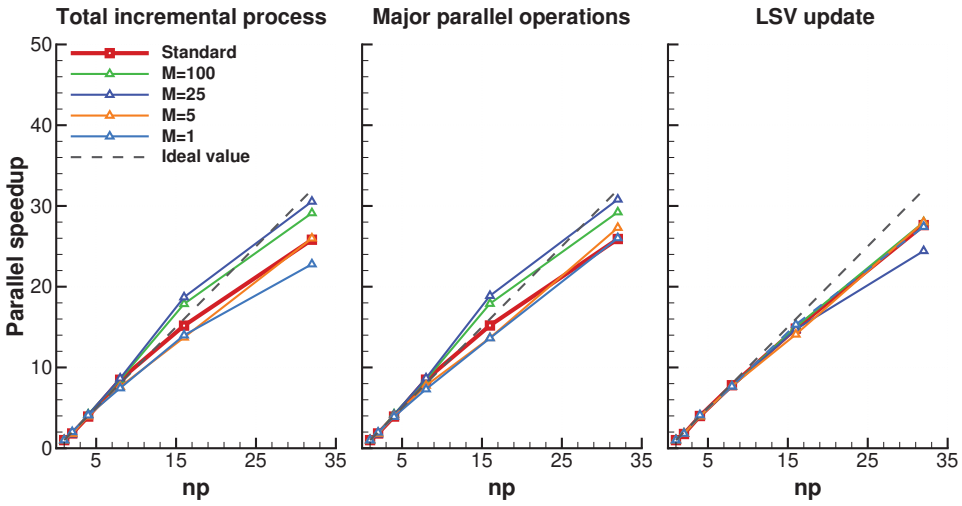
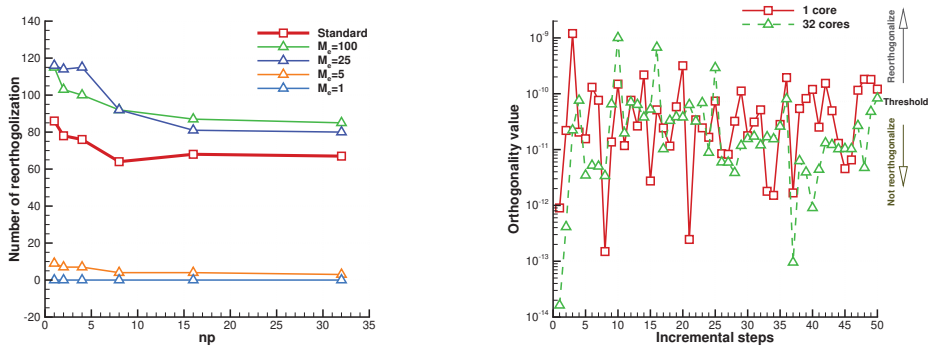


Figure 3.11: The parallel speedup of standard/enhanced parallel incremental SVDs for the total incremental time, major parallel operations and updating left singular vectors on a thin matrix.

creases. The computational cost of reorthogonalition can benefit from the cancellation of round-off errors when using more cores. Figure 3.12(b) compares the orthogonality values computed by the standard iSVD with 1 and 32 cores respectively. The number of orthogonality values exceeding the threshold in the 1-core case is larger than that number with 32 cores.

Figure 3.13 shows the time ratio of each part for the standard and enhanced iSVDs. These ratios for all iSVDs keep roughly a constant value in parallel. The LSV update and reorthogonalization are two main contributors to the computing cost. When using a



(a) Reorthogonalization number (b) Orthogonality values for standard iSVD

Figure 3.12: The number of reorthogonalizations of the standard incremental SVD and enhanced online algorithm in parallel, and the comparison of orthogonality values computed for 1-core and 32-core cases for a deep matrix (327681 × 200).

small M_e, the contribution of projection also plays a crucial role in computing cost.

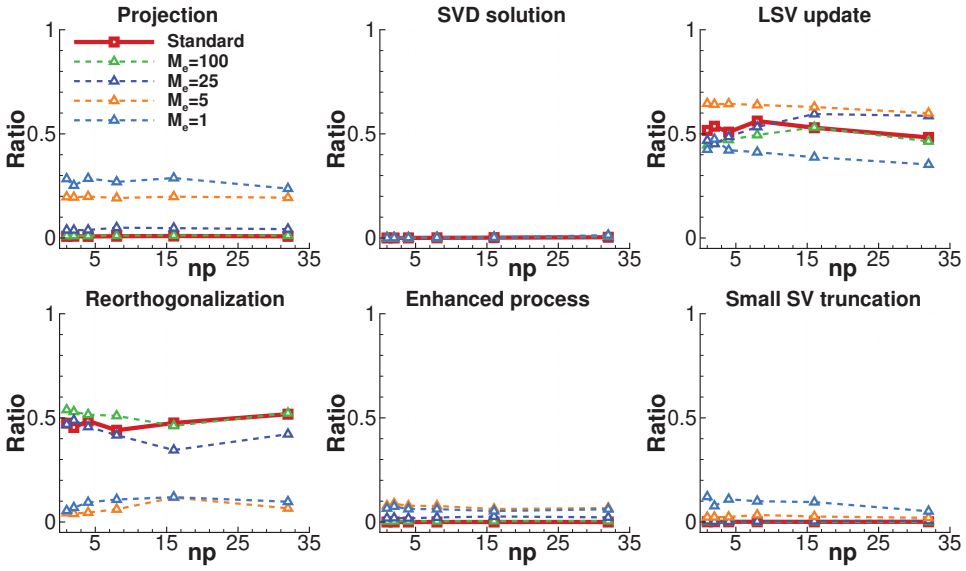


Figure 3.13: The time ratio of computing cost for different parts during the standard/enhanced parallel incremental SVDs on a 327681-by-200 deep matrix, including the projection, SVD solution, LSV update, reorthogonalization, enhanced process, small SV truncation.

Normalized time consumption Since the re-orthogonalization relies on the arithmetical operations that are affected by the round-off error, the number of reorthogonaliza-

tions varies when we run the problem with different numbers of cores. Therefore the scaling performance is oscillatory rather than monotonic, making the results of the scaling study difficult to interpret. We thus use the number of re-orthogonalizations to normalize the reorthogonalization computing time to obtain

$$t_{\text{normal}} = t_{\text{total}} - t_{\text{reorthogonal}}(1 - \frac{1}{n_{\text{reorthogonal}}}), \tag{3.45}$$

where t_{total} and t_{normal} denote the computing time before and after the normalization. $t_{\text{reorthogonal}}$ and $n_{\text{reorthogonal}}$ represent the computing cost and the number of reorthogonalizations. When the $n_{\text{reorthogonal}} = 0$, the computing time will be kept the same; otherwise, the normalized process will be utilized. Figure 3.14 shows the parallel speedups of normalized computational time, which now shows scaling just below the ideal value. This demonstrates the scalability of the enhanced online algorithm.

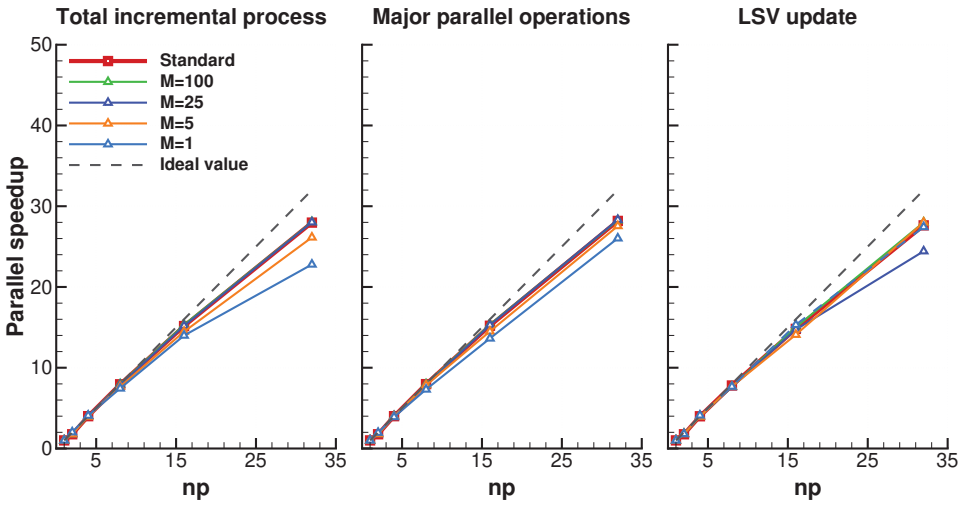
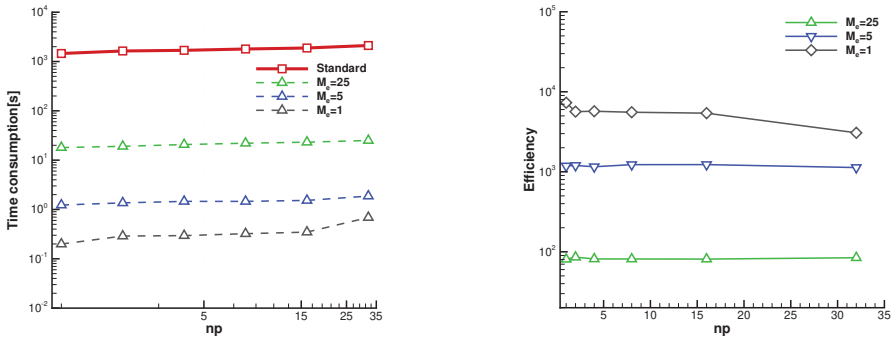


Figure 3.14: Parallel speedups of standard incremental SVD and enhanced online algorithm on a deep matrix (327681 × 200), with the normalized computing time for the total incremental process, major parallel operations and LSV update.

A WIDE MATRIX WITH MORE SNAPSHOTS

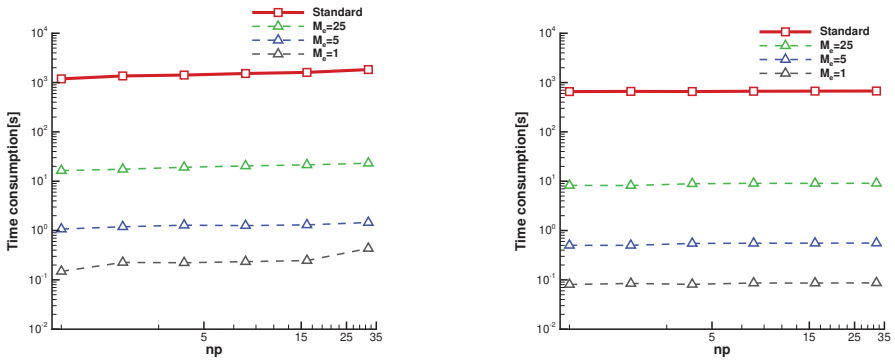
There is another typical scenario where we are interested in statistical steady phenomena. In this case, we consider a large number of snapshots exceeding the number of DoF of a computational mesh. Figure 3.15(a) presents the total computing time of the standard and enhanced iSVDs for a wide matrix (257 × 2000). Here, we examine the weak scaling in which the increase of computing time is reasonably small while the number of cores is raised. As shown in Figure 3.15(b), the truncated speedup of the enhanced algorithm remains constant although this value is degraded for the case of $M_e = 1$. Figure 3.16 shows the computing cost of the major parallel operations and LSV update. The cost of major parallel operations exhibits a trend similar to that of the total computing

time. However, the computing time of the LSV update shows a constant computing cost while increasing the number of cores. The enhanced online algorithm is thus weakly scalable for this wide matrix.



(a) Time consumption (b) Enhanced efficiency

Figure 3.15: Time consumption of standard incremental SVD and enhanced online algorithm in a weak study on a wide matrix (257 × 2000), and the truncated speedup by the enhanced process.



(a) Major parallel operations (b) LSV update

Figure 3.16: Time consumption of the major parallel operations and LSV update with the standard incremental SVD and enhanced online algorithm during a weak scaling study for a wide matrix (257 × 2000).

IMPROVEMENT TO COMPUTATIONAL PERFORMANCE BY THE ENHANCED ALGORITHM

The truncated speedup of the proposed enhanced online algorithm is shown to be constant for both deep and wide matrices. Considering that the dominant computational cost results from LSV update, we can estimate this improved value by an a priori analysis of the computing complexity, as described in Proposition 3.

Proposition 3. *Suppose we have a dense matrix* $\boldsymbol{U}^{n\times m}$*, the float-point operations of the*

LSV update in the standard iSVD can be estimated by $\mathcal{O}(t_{\text{standard}})$,

$$t_{\text{standard}} = \begin{cases} 2n[\frac{m}{6}(m+1)(2m+1)-1]+3n, & n \geqslant m \\ 2n[\frac{n}{6}(n+1)(2n+1)-1+n^2(m-n)]+3n, & n < m. \end{cases} \tag{3.46}$$

If we use $M_e = k_e$, $k_e < \min(n,m)$, *for the enhanced iSVD, these float-point operations can be reduced to* $\mathcal{O}(t_{\text{enhanced}})$,

$$t_{\text{enhanced}} = 2n[\frac{k_e}{6}(k_e+1)(2k_e+1)-1+(k_e+1)^2(m-k_e)]+3n. \tag{3.47}$$

The t_{enhanced} *is equivalent to* t_{standard} *when* $k_e = \min(n,m)$.

Proof. We first verify the expression for a deep matrix ($n \geqslant m$). For the standard iSVD, we need an initialization with the float-point operations (FLOPs) of $\mathcal{O}(3n)$ before proceeding with the LSV update. We can evaluate the FLOPs in each LSV update as $\mathcal{O}(2nl^2)$, where l denotes the number of snapshots. This relies on $[\boldsymbol{U}j]^{n\times l}$ and $\boldsymbol{V}_Q^{l\times l}$. The LSV update starts with the second snapshot ($l = 2$) after the initialization. Therefore we can estimate the complexity of the total computational cost as

$$t_{\text{standard}} = 2n(2^2+\cdots+m^2)+3n = 2n[\frac{m}{6}(m+1)(2m+1)-1]+3n. \tag{3.48}$$

Likewise, we can compute the complexity of the enhanced iSVD by replacing the cost with $\mathcal{O}(2n(k_e+1)^2)$ after k_e-th incremental step, leading to

$$\begin{aligned} t_{\text{enhanced}} &= 2n[2^2+\cdots+k_e^2+(k_e+1)^2+(k_e+1)^2\cdots+(k_e+1)^2]+3n \\ &= 2n[\frac{k_e}{6}(k_e+1)(2k_e+1)-1+(k_e+1)^2(m-k_e)]+3n. \end{aligned} \tag{3.49}$$

For a wide matrix ($n < m$), t_{enhanced} is the same as shown for the deep matrix, and t_{standard} can be computed by

$$\begin{aligned} t_{\text{standard}} &= 2n(2^2+3^2+\cdots+n^2+n^2+\cdots+n^2)+3n \\ &= 2n[\frac{n}{6}(n+1)(2n+1)-1+n^2(m-n)]+3n. \end{aligned} \tag{3.50}$$

When $k_e = \min(n,m)$ for a deep or wide matrix, the enhanced iSVD is equivalent to the standard iSVD, leading to $t_{\text{enhanced}} = t_{\text{standard}}$. Consequently, Proposition 3 is confirmed. □

As the computing cost mainly results from matrix/vector multiplications, we can assume that the LSV update is the leading contribution as shown in Sections 3.4.1 and 3.4.2. We can then estimate the truncated speedup of the enhanced online algorithm as $\mathcal{O}(\eta)$,

$$\eta = \frac{t_{\text{standard}}}{t_{\text{enhanced}}}. \tag{3.51}$$

For convenience, $\mathcal{O}$ is ignored for the description of the computing complexity in the following content if there are no conflicts.

When the truncation number M_e is small, the computing cost of the LSV update is not the only large contribution due to the reduction of the size of $\boldsymbol{V}, \boldsymbol{V}_Q$. In such situations, the computing cost of projection becomes crucial, as shown in Figures 3.9 and 3.13. Its computing complexity can be estimated based on the number of FLOPs in the projection, as summarized in Table 3.3.

Table 3.3: Float-point operations in the projection part at the k-th incremental step with $\boldsymbol{V}^{n\times(k-1)}(k\geq 2)$

Input operation	Float-point operations
$d=\boldsymbol{V}^\top c$	$\mathcal{O}(2n(k-1))$
$\boldsymbol{V}d$	$\mathcal{O}(2n(k-1))$
$c-\boldsymbol{V}d$	$\mathcal{O}(n)$
$(c-\boldsymbol{V}d)^\top(c-\boldsymbol{V}d)$	$\mathcal{O}(2n)$
Total	$\mathcal{O}(4nk-n)$

For a standard iSVD of a deep matrix, we can express the computing complexity of projection over the entire incremental process as

$$t_s^{proj}=4n(2+3+\cdots+m)-n*(m-1)=2n(m+2)(m-1)-n(m-1)\,. \tag{3.52}$$

This complexity for a wide matrix is stated as

$$\begin{aligned} t_s^{proj} &= 4n(2+3+\cdots+n+n+\cdots+n)-n*(m-1)\\ &= 2n(n+2)(n-1)+4n^2(m-n)-n(m-1)\,. \end{aligned} \tag{3.53}$$

The computing complexity for the enhanced iSVD is expressed as

$$\begin{aligned} t_e^{proj} &= 4n[2+3+\cdots+k_e+(k_e+1)+\cdots+(k_e+1)]-n(m-1)\\ &= 2n(k_e+2)(k_e-1)+4n(k_e+1)(m-k_e)-n(m-1)\,. \end{aligned} \tag{3.54}$$

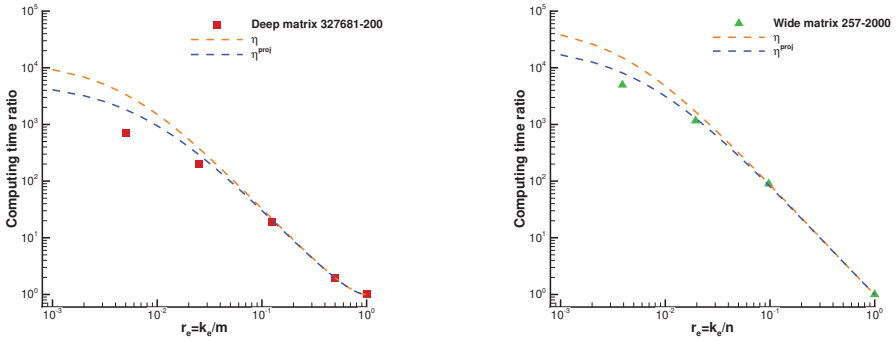
By virtue of supplementing the cost of projection, we can estimate the truncated speedup as $\mathcal{O}(\eta^{proj})$,

$$\eta^{proj}=\frac{t_{\text{standard}}+t_s^{proj}}{t_{\text{enhanced}}+t_e^{proj}}\,, \tag{3.55}$$

which consists of the LSV update and projection. The difference between η and η^{proj} is the impact of the projection on the incremental SVDs.

Figure 3.17(a) compares the computations of η and η^{proj} using the numerical dataset from the above-mentioned deep matrix. These two estimations are similar when predicting the truncated speedup for a large number ratio $r_e=k_e/m$, while the η^{proj} is more accurate for small r_e values. This is because the projection makes a great difference in the computational cost of the enhanced iSVD for smaller truncation numbers k_e, i.e. a small r_e here. In addition, the performance for the wide-matrix case bears a striking resemblance to that of the deep matrix, as shown in Figure 3.17(b). Note that computing complexity is utilized to quantify the intrinsic time requirements of the algorithm, whereas the actual run time will depend on the computer hardware and software

implementation. Therefore, the computing complexity gives only an indication of the run time. Furthermore, the actual run time can also be affected by other operations that are here neglected but can have an effect when the ratio r_e/k_e is small, as shown in Figure 3.13. Overall, the estimation with the projection gives a reasonable prediction for improved efficiency.



(a) A deep matrix (b) A wide matrix

Figure 3.17: A priori analyses on the efficiency improved by the enhanced online algorithm for incremental SVD over (a) a deep matrix (327681×200) and (b) a wide matrix (257×2000), compared with the physical data.

3.5. SUMMARY

We developed an enhanced online algorithm based on the incremental SVD for modal analysis, e.g. POD, which can efficiently perform the POD analysis on the fly. Two lower-bound estimators are formulated to give a posteriori analysis of the enhanced online algorithm so that the accuracy of the reconstructed solution from POD modes can be calculated. e_{simp} is shown to be more accurate and effective for evaluating the cumulative energy captured by selected POD modes. Numerical results demonstrate this algorithm can significantly improve efficiency compared with the standard incremental SVD. The eigenvalues can be influenced if a very small truncation number (M_e) is considered for the enhanced process, but good accuracy can be obtained with a reasonable M_e. What's more, it is shown that the enhanced online algorithm can be scaled well in parallel. This is also true for standard incremental SVD. But it is more efficient to build POD modes with the enhanced online algorithm. The speedup of the enhanced algorithm is independent of the number of cores.

4

ROR-DRIVEN ADJOINT-BASED AMR ON A 1D FORCED BURGERS PROBLEM

Parts of this chapter have been published in Computer Methods in Applied Mechanics and Engineering **379**, (2021) [129].

The potential of adjoint-based mesh adaptation for LES is analysed based on a 1D Burgers problem with a multiple-frequency forcing term. A POD-based Reduced-Order Representation (ROR) is introduced to reduce the storage requirement for the LES adjoint problem. We investigate the effect of the ROR on the accuracy of the adjoint and the resulting AMR outputs.

4.1. INTRODUCTION

Large Eddy Simulation (LES), in which one resolves large-scale turbulent structures while modelling the impact from smaller turbulent scales, has the potential to deliver reliable flow predictions for many applications [137]. LES can resolve an appropriate range of large scales when the computational mesh is well defined [8]. In practice, the construction of a computational mesh for LES usually involves trial and error, even for engineering experts, since it is difficult to anticipate the effects of complex flow features, such as laminar-turbulent transition, boundary layer separation or vortex interactions, on a desired Quantity of Interest (QoI). It is thus natural to consider Adaptive Mesh Refinement (AMR) [1] for automatically constructing the computational mesh.

Adjoint methods have been developed to determine local contributions to the error in a chosen QoI so that the adapted mesh can provide the highest accuracy per degree of freedom. The accuracy and efficiency of adjoint-based AMR have been demonstrated in a wide range of steady problems, as mentioned in Section 1.2. Although there are studies about the application of adjoint-based mesh adaptation to unsteady simulations, the application of adjoint-based AMR to LES poses new challenges, as discussed in Section 1.3, i.e. high-dimensional data, model error, chaotic features of turbulence. The work here focuses on the high computing costs of adjoint solutions. We will investigate how to overcome unfeasible storage requirements and incorporate the modelling error in AMR.

We propose the use of Reduced-Order Representations (RORs) to efficiently represent the primal solution when solving unsteady adjoint problems. The performance of this approach is studied using the 1D Burgers equation, which is often used in the development of numerical schemes for turbulent flows due to its convective nonlinearity and forward energy cascade. We place the problem in the context of AMR for LES by considering a time-averaged QoI and by employing coarse meshes, with which the influence of the subgrid-scale model is significant. The enhanced online algorithm is considered to reduce the computing cost of constructing a ROR for high-dimensional LES data.

This chapter is organized as follows. A paradigm for mesh adaptation in LES is described in Section 4.2, in which an LES model and discretization scheme, adjoint method, a posteriori error estimation, and mesh adaptation strategy are formulated. Section 4.3 describes the use of a standard ROR and an EOA ROR for adjoint-based AMR. After the validation of error estimation on a Burgers problem with a manufactured solution in Section 4.5, we present numerical experiments with the proposed AMR approach on an unsteady non-linear Burgers problem with a multi-frequency forcing term in Section 4.6. Results obtained using full-order solutions, a standard ROR and an EOA ROR are compared in Section 4.7. Concluding remarks appear in Section 4.8.

4.2. A PARADIGM FOR ADJOINT-BASED MESH ADAPTION

4.2.1. PROBLEM FORMULATION AND DISCRETIZATION

We consider the one-dimensional (1D) Burgers equation over a space-time domain $\Omega : [0,1] \times \mathrm{I} : [0,\mathrm{T}]$. The Burgers equation is often used as a mathematical model for applications that involve shock wave propagation in viscous flows or idealized turbulence [136]. The Burgers equation is expressed as

$$\mathscr{N}(u) = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = f\,, \tag{4.1}$$

where $\mathscr{N}(\cdot)$ is a non-linear operator and u is the solution with boundary conditions $u(0,t) = u(1,t) = 0$ and an initial condition $u(x,0) = u_0$. ν is the viscosity coefficient and $f \in \mathbb{R}$ is a known forcing term. Note that we use a 1D problem here to explore the methodology; however, the approaches considered can be directly extended to multi-dimensional problems.

VARIATIONAL MULTISCALE METHOD

We employ the finite-element method to solve the primal problem with Dirichlet boundary conditions using the weak form

$$\mathscr{R}(u,\omega) = (u_t,\omega) - (uu/2,\omega_x) + (\nu u_x,\omega_x) - (f,\omega) = 0\,, \forall \omega \in \mathscr{V}\,, \tag{4.2}$$

where $\omega \in \mathscr{V}$ are weighting functions and $\mathscr{V} = \mathscr{V}(\Omega)$ denotes both the solution space and weighting space. $\mathscr{R}(\cdot,\cdot)$ denotes the weak form of the residual operator and $(\cdot,\cdot)$ is the L_2 inner product. The inner product used in this chapter is defined by the spatial domain Ω by default.

The Variational Multiscale Method (VMM) [95, 98] is then used to derive a form suitable for LES. In VMM, the flow solution is split into two components, the resolved scales $\bar{u}$ and the unresolved scales u'. The unresolved-scale equations are driven by the strong residual, i.e. $R(\bar{u}) = \mathscr{N}(\bar{u}) - f$. The simplest algebraic model for u' uses a quasi-static subscale assumption and a volume-averaged Green's function to write $u' \approx -\tau R(\bar{u})$. For the current problem, we use this approximation along with an expression for τ from Wang et al. [138], viz. $\tau = [\frac{4}{h^2}\bar{u}^2 + 3\pi\nu^2(\frac{4}{h^2})^2]^{-1/2}$. Substituting $u = \bar{u} + u'$ into Equation (4.2), the weak form is then:

$$\mathscr{A}(\bar{u},\omega) = (\bar{u}_t,\omega) - (\frac{1}{2}\bar{u}\bar{u},\omega_x) + (\nu\bar{u}_x,\omega_x) - (f,\omega) - (\bar{u}u',\omega_x) - (\frac{1}{2}u'^2,\omega_x) - (\nu u',\omega_{xx}) = 0\,, \tag{4.3}$$

where it has been assumed $u' = 0$ on the boundary. By virtue of $u' = -\tau R(\bar{u})$, $\bar{u}$ is exclusively determined from Equation (4.3). In analogy with LES, the last three terms of Equation (4.3) correspond to a model for the effects of the subgrid scales.

DISCRETIZATION

When numerically solving the flow problem, we replace $\bar{u}$ by $\bar{u}_h$, leading to a discrete system

$$\mathscr{A}_h(\bar{u}_h,\omega_h) = 0\,, \forall \omega_h \in \mathscr{V}_h\,, \tag{4.4}$$

where the subscript h denotes element size within a computational mesh with N_v degrees of freedom. $\bar{u}_h \in \mathcal{V}_h$ is abbreviated to u_h in the subsequent text where there is no conflict. The semi-discrete technique is used to discretize this unsteady model problem. Specifically, we use piecewise linear basis functions for spatial discretization and a four-stage Runge-Kutta scheme to advance the primal problem in time from $t = 0$ to $t = \mathrm{T}$. Note that the $(\nu u', \omega_{xx})$ term in Equation (4.3) is zero in this case due to the use of piecewise linear functions.

4.2.2. ADJOINT METHOD

The adjoint method is used to provide estimates of local contributions to the error in a chosen QoI, allowing for the construction of goal-oriented adapted meshes. In unsteady simulations, a QoI is often a statistical function of the primal solution. In this chapter, we consider a volume-integrated statistical function $\bar{J}(u) = \frac{1}{\mathrm{T}}\int_{\mathrm{I}} J(u)\,\mathrm{d}t = \frac{1}{\mathrm{T}}\int_{\mathrm{I}}(g,u)_\Omega\,\mathrm{d}t$, where g is a real function from $\mathbb{R} \to \mathbb{R}$ and $\bar{J}(u) \in \mathbb{R}$. I represents the temporal space. Note that although only results for linear QoI will be discussed in later sections, the adjoint equation and error expressions given below are also valid for non-linear QoI.

By virtue of a Lagrange function, the adjoint equation derived for the current problem is

$$\mathscr{L}_u^* v = -\frac{\partial v}{\partial t} - u\frac{\partial v}{\partial x} - \nu\frac{\partial^2 v}{\partial x^2} = g_u\,, \tag{4.5}$$

where v is the Lagrange multiplier or adjoint variable with homogeneous boundary conditions $v(0,t) = v(1,t) = 0$ and an initial condition $v(x,\mathrm{T}) = 0$. $\mathscr{L}_u^*(\cdot)$ is a linearised adjoint operator which relies on the primal solution u. g_u is a Fréchet derivative of $J(u)$, defined as

$$(g_u, \tilde{u}) \coloneqq \lim_{\epsilon\to 0}\frac{J(u+\epsilon\tilde{u}) - J(u)}{\epsilon}\,, \quad \forall\, u, \tilde{u} \in \mathcal{V}\,. \tag{4.6}$$

The adjoint residual operator is expressed as $R^*_{[u]}(\cdot) = \mathscr{L}_u^*(\cdot) - g_u$ with respect to a given u. We solve the adjoint problem using the same VMM employed for the primal problem.

4.2.3. A POSTERIORI ERROR ESTIMATION FRAMEWORK

Before developing an a posteriori error estimation, we consider two fundamental properties, the *adjoint identity* and the *averaging Fréchet operator*. The adjoint identity, which can be easily verified by partial integration, is

$$(\mathscr{L}_u^* v, \tilde{u})_{\Omega\times\mathrm{I}} = (v, \mathscr{L}_u\tilde{u})_{\Omega\times\mathrm{I}} + (v,\tilde{u})_\Omega|_{t=t_0}\,, \tag{4.7}$$

for $\forall\, v, u, \tilde{u} \in \mathcal{V}$. I is referred to time integration. The last term originates from the contribution of non-zero values at statistical-starting time t_0. $\mathscr{L}_u$ is the Fréchet derivative of $\mathcal{N}(u)$,

$$\mathscr{L}_u\tilde{u} \coloneqq \lim_{\epsilon\to 0}\frac{\mathcal{N}(u+\epsilon\tilde{u}) - \mathcal{N}((u)}{\epsilon}\,, \quad \forall\, u, \tilde{u} \in \mathcal{V}\,. \tag{4.8}$$

We can formulate the averaging Fréchet operator, $\bar{\mathscr{L}}^*_{(u_1,u_2)}(\cdot)$, by integrating Equation (4.5) of u from u_1 to u_2, which enables us to estimate the error for non-linear prob-

lems. By defining $u = u_1 + \theta(u_2 - u_1)$, the adjoint equation is integrated as

$$\bar{\mathscr{L}}^*_{(u_1,u_2)} v \equiv \int_0^1 \mathscr{L}^*_{u=u_1+\theta(u_2-u_1)} v \, \mathrm{d}\theta \overset{Equation\ (4.5)}{=} \int_0^1 g_{u=u_1+\theta(u_2-u_1)} \mathrm{d}\theta \equiv \bar{g}_{(u_1,u_2)}\,, \tag{4.9}$$

where the subscripts, u_1 and u_2, denote the integration domain and $\bar{g}_{(u_1,u_2)}$ represents an averaged value of g_u on this domain. Considering the definition in Equation (4.6), we can use this $\bar{g}_{(u_1,u_2)}$ to express the difference of a QoI as

$$J(u_1) - J(u_2) = (\bar{g}_{(u_1,u_2)}, u_1 - u_2)\,. \tag{4.10}$$

For the unsteady Burgers problem, we have $\bar{\mathscr{L}}^*_{(u_1,u_2)} v = \mathscr{L}^*_{\frac{u_1+u_2}{2}} v$. Likewise, the average linearized operator in Equation (4.9) maintains the adjoint identity as follows

$$(\bar{\mathscr{L}}^*_{(u_1,u_2)} v, u)_{\Omega\times \mathrm{I}} = (v, \bar{\mathscr{L}}_{(u_1,u_2)} u)_{\Omega\times \mathrm{I}} + (v,u)_{\Omega}|_{t=t_0}\,, \tag{4.11}$$

where $v, u, u_1, u_2 \in \mathcal{V}$. $\bar{\mathscr{L}}_{(u_1,u_2)}(\cdot)$ is the averaging operator of $\mathscr{L}_u(\cdot)$ by integrating u from u_1 to u_2, which satisfies

$$\bar{\mathscr{L}}_{(u,u_h)}(u_h - u) = \mathscr{N}(u_h) - \mathscr{N}(u) = \mathscr{R}(u_h)\,. \tag{4.12}$$

Substituting $u_1 = u$ and $u_2 = u_h$ into Equations (4.9) to (4.11), we introduce an expression for estimating the error ϵ of the QoI

$$\begin{aligned}\epsilon &= \bar{J}(u) - \bar{J}(u_h)\\ &= \underbrace{-\frac{1}{\mathrm{T}}(v_h, R(u_h))_{\Omega\times \mathrm{I}} - \frac{1}{\mathrm{T}}(v_h, u_h - u)_{\Omega}|_{t=t_0}}_{\text{Adjoint correction}} + \underbrace{\frac{1}{\mathrm{T}}(R^*_{\left[\frac{u+u_h}{2}\right]}(v_h), u_h - u)_{\Omega\times \mathrm{I}}}_{\text{Remaining error}} = \sum_{e=1}^{N_{\text{cells}}} \epsilon_e\,,\end{aligned} \tag{4.13}$$

where N_{cells} denotes the number of elements at the current AMR level. The error estimation is divided into two parts, adjoint correction and remaining error. There is a temporal contribution to the adjoint correction due to the non-zero value of u at the starting time of the statistical time period, which does not appear in steady simulations. In practice, $R^*_{[\frac{u+u_h}{2}]}(v_h)$ is replaced by $R^*_{[u_h]}(v_h)$, where u_h is the discrete primal solution. Here, the VMM is used to approximate the exact solution u as $u_h + u'_h$. The resulting error estimation can be constrained to each element as shown in the last equal sign of Equation (4.13), and expressed in terms of elemental error estimators ϵ_e.

4.2.4. MESH ADAPTATION STRATEGY

We start from a very coarse uniform mesh and only use refinement. A local error indicator, η_e, is determined from the elemental error estimator as $\eta_e = |\epsilon_e|$ to facilitate mesh adaptation. Different adaptation strategies [9] can be developed based on η_e. We employ a prescribed percentage (10%) of mesh cells with the largest errors as the criterion for mesh refinement [14, 30]. The resulting procedure of adjoint-based AMR is presented in Figure 4.1. In each mesh refinement loop, the error estimation and adaptation strategy are executed to generate a new computational mesh for the next AMR level.

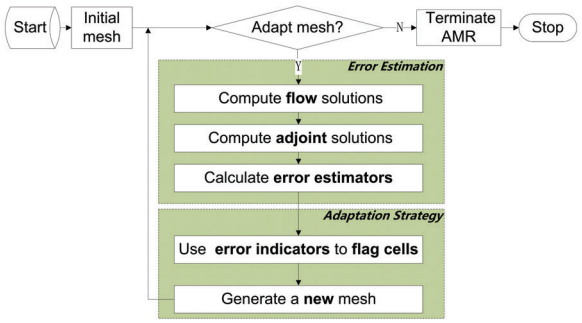


Figure 4.1: A general procedure of mesh adaptation based on the adjoint method for unsteady simulations.

Without additional treatment, the smoothness of the mesh will deteriorate during AMR. Thus, a balancing step is introduced to improve mesh smoothness, as shown in Algorithm 4. The basic principle is that a cell will be flagged for refinement if the ratio between the size of this cell and the size of its neighbors would become larger than 2. This balancing step is recursively executed until there are no more elements that need to be refined.

Algorithm 4 Balancing procedure for AMR

```
refined = 1
while refined > 0 do                                   ▷ Loop if refinement is needed
    refined = 0, temp = cellFlag
    for i ≤ N do                                       ▷ Loop all cells indexed from 1 to N
        if cellFlag(i) = 1 then                        ▷ A marked cell
            if i ≠ 1 and cellFlag(i − 1)=0 then
                if Length(e_i)/Length(e_{i−1}) < 1 then
                    temp(i − 1) =1, refined += 1       ▷ Flag the adjacent cell
            if i ≠ N and cellFlag(i + 1)=0 then
                if Length(e_i)/Length(e_{i+1}) < 1 then
                    temp(i + 1) =1, refined += 1       ▷ Flag the adjacent cell
    cellFlag = temp
```

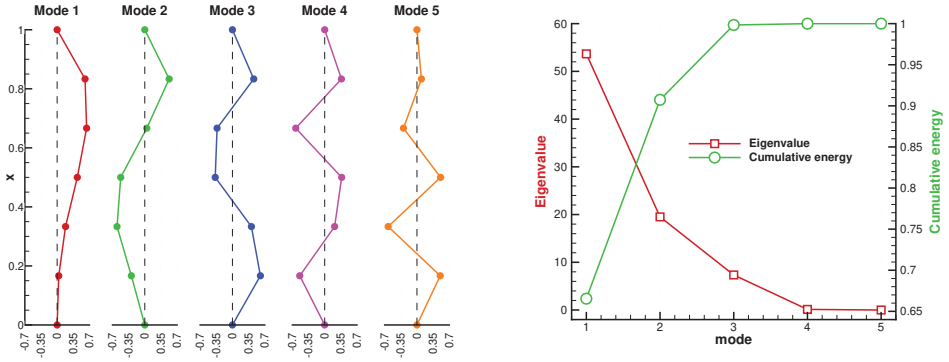
4.3. POD-BASED REDUCED-ORDER REPRESENTATION FOR AMR

In order to make adjoint-based AMR affordable for LES, we represent the primal flow problem in a low-order space that can be accessed efficiently. For unsteady flow problems, Reduced-Order Models (ROM) based on projection have been widely used. In this case, one projects representative modes (typically a truncated set of POD modes) onto the governing equations to obtain a low-order system. For the current application, however, there is no need to predict the primal solution beyond the original dataset. Thus

representative modes and their coefficients can be used directly as a Reduced-Order Representation (ROR) of the primal solution in the adjoint problem. In the following, we initially construct the ROR by applying the standard SVD implemented in LAPACK [78] to store primal solution data, which results in exact POD modes. We refer to this method as the offline SVD. The procedure then used for ROR-driven AMR is described in Section 4.3.1. For realistic applications, however, very large datasets would need to be considered, making the cost of an offline SVD prohibitive. Therefore, we also introduce an enhanced online algorithm to build the ROR based on an incremental SVD [86, 88], described in Section 4.3.2. This produces approximations for the POD modes and their amplitudes.

4.3.1. OFFLINE ROR

The standard approach constructs a ROR offline by gathering complete snapshots into a solution matrix and then applying the SVD analysis, which gives the POD modes (ϕ) and coefficients (α) at once. Figure 4.2 shows the distribution of POD modes, eigenvalues and cumulative energy from the solution of a typical Burgers problem, which is also the starting situation for AMR. In this case, the first mode represents a significant part of the instantaneous solution as it accounts for 66.5% of the total energy. The ROR solution is



(a) Distribution of POD modes (b) Eigenvalues and cumulative energy

Figure 4.2: POD analysis of a Burgers problem solved on a stationary mesh with 5 primal flow variables.

then expressed as

$$u(x,t_i) = \bar{u}(x) + \sum_{j=1}^{M} \alpha_j(t_i)\,\phi^{(j)}(x), \tag{4.14}$$

where $\phi^{(j)}$, $j = 1,2,\cdots,M$, denote a low number of selected POD modes, and $\bar{u}$ represents the mean value. As shown in Figure 4.3, the AMR procedure is modified by replacing the primal solution in the adjoint problem with a ROR determined using the standard SVD. We refer to this as AMR using an offline ROR. Note that the adjoint problem is still solved in a full-order space. The offline ROR is utilized as a benchmark for the online RORs introduced in the next section.

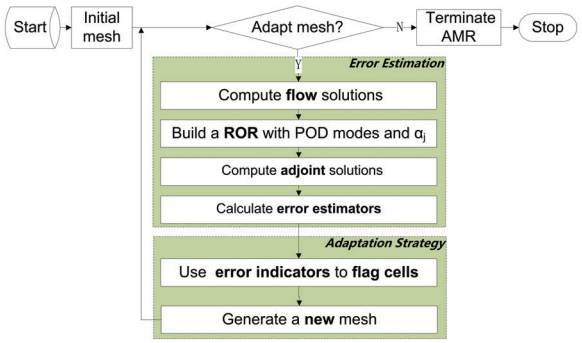


Figure 4.3: A general procedure of mesh adaptation based on adjoint method and ROR for unsteady simulations.

4.3.2. ENHANCED ONLINE ALGORITHM FOR RORS

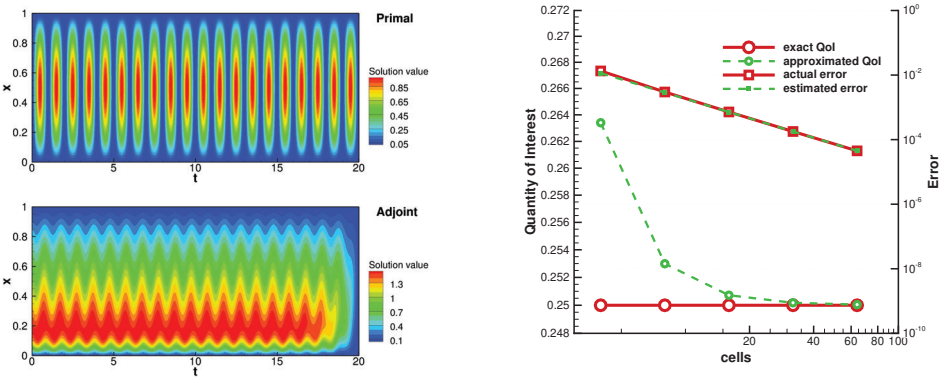
The modal decomposition of large flow datasets can lead to intractable storage requirements [88]. To overcome this, we introduce an enhanced online algorithm (EOA) based on the incremental SVD [86, 88]. Algorithm 3 describes this enhanced online algorithm for building a ROR online. The improvement is achieved by incorporating the truncation of a selected number of POD modes (M) into the incremental algorithm as shown in Algorithm 3 between line 14 and line 16, leading to the EOA. This is because the number of POD modes necessary for building an accurate ROM is usually far fewer than the number of variables and time steps.

We present the validation of the proposed mesh adaptation strategy and describe the performance of AMR computations using POD-based RORs. We consider Equation (4.1) with $\nu = 0.01$ and a QoI defined as $\bar{J}(u) = \frac{1}{\mathrm{T}}(\sin(\pi x), u)_{\Omega\times \mathrm{I}}$. To avoid the influence of temporal discretization errors, a small time step, $\Delta t = 10^{-3}$, is used for solving both primal and adjoint problems. This value gives negligible time discretization errors for the complete range of AMR meshes considered in the numerical results. Adjoint-based error estimations are obtained using Equation (4.13) with cubic spline reconstructions for the primal and adjoint solutions. We first verify the error estimation using a manufactured solution in Section 4.4. Then, the effectiveness of the proposed AMR strategy is validated on a Burgers problem with a multi-frequency forcing term in Section 4.5. The use of offline and online RORs is studied in Section 4.6 and Section 4.7, respectively.

4.4. VERIFICATION OF ADJOINT-BASED ERROR ESTIMATION

First of all, the error estimation procedure is verified using a force f corresponding to the manufactured solution $u(x,t) = \sin^2(\pi t)\sin(\pi x)$, from $t = 0$ to $t = 20$. This exact solution is used to compute the actual value of the QoI. Discrete solutions of the primal and adjoint problems are shown in Figure 4.4(a). The adjoint solution is propagated in a direction opposite to the primal velocity. The most sensitive regions are not those with the largest primal solution or weighting function values, making it difficult to use feature-based AMR for this case. Figure 4.4(b) depicts the QoI's approximation and cor-

responding error as the mesh is refined uniformly. The computed QoI converges to the exact value and the error estimation displays a good agreement with the actual value.



(a) Instantaneous solutions

(b) Error estimation

Figure 4.4: Instantaneous solutions for a manufactured Burgers problem and associated error analysis of a volume-integrated output on uniformly refined meshes. Solid lines denote the values computed from analytical solutions while dashed lines are the approximation from numerical solutions and the adjoint-based error estimation with VMM.

4.5. VALIDATION OF ADJOINT-BASED AMR WITH FULL-ORDER SOLUTIONS

We now introduce a forcing term f to produce a solution with large fluctuations and a boundary layer near the right boundary

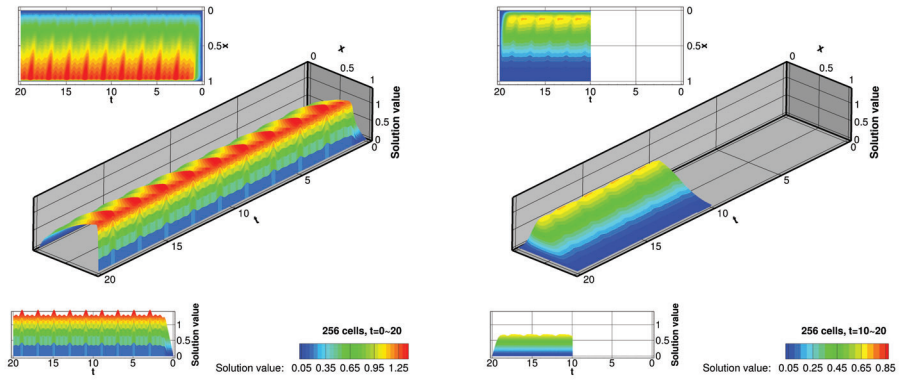
$$f(x,t) = 1 + q(x) * \sum_{i=1}^{N_f} g_i(t)\sin(k_i x), \tag{4.15}$$

where $N_f = 3$ and the $g_i(t)$ are chosen so $|g_i(t)| \leq 1$, specifically,

$$g_i(t) = \sin(i\pi t), k_1 = i\pi. \tag{4.16}$$

$q(x) = 5/30$ is a coefficient used to tune the amplitude of the forcing term at various wave numbers so that the fluctuations can be controlled independently. The primal problem is advanced from $t = 0$ to $t = 20$ while the adjoint problem is solved backward from $t = 20$ to $t = 10$. The temporal interval ($t \in [0,10]$) is sufficiently long to allow the primal flow problem to arrive at a statistically steady state. Instantaneous solutions are shown in Figure 4.5. The primal solution changes periodically and a reverse propagation of the adjoint solution can be observed as well.

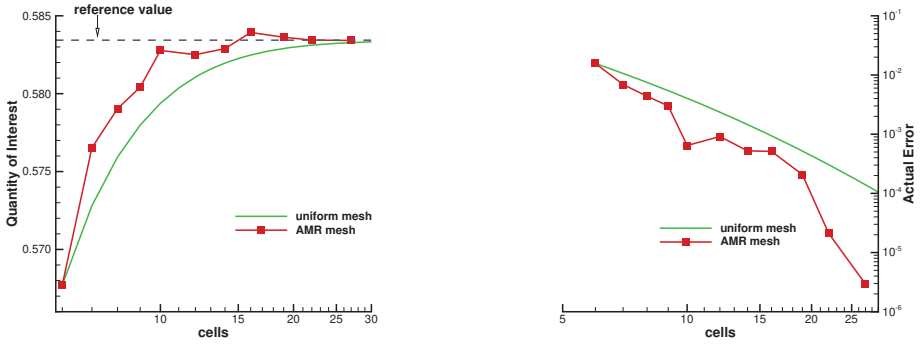
The mesh adaptation strategy of Section 4.2.4 is compared with uniform mesh refinement for this case. Here, the reference value of the QoI is calculated on a fine mesh with 256 elements, as there is no analytical solution. Figure 4.6 presents the QoI and associated errors with increasing levels of mesh refinement determined by the proposed



(a) Primal solutions (b) Adjoint solutions

Figure 4.5: The distribution of primal and adjoint solutions for the Burgers problem with a forcing term.

AMR procedure. The AMR technique is clearly more accurate than uniform enrichment for a given number of degrees of freedom. The corresponding error also converges faster but in a less regular manner.



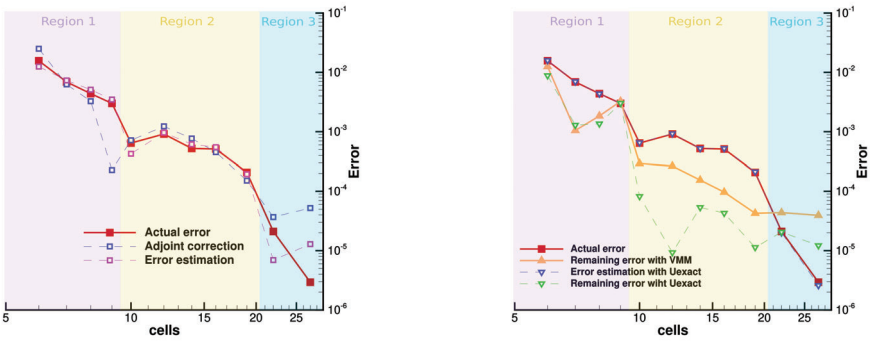
(a) Approximation of QoI (b) Approximation error

Figure 4.6: AMR analyses compared with uniform refinement, (a) QoI and (b) associated approximating error.

4.5.1. DISCUSSIONS OF ERROR ESTIMATION

The convergence of the error estimate is shown in Figure 4.7(a), where we mark three different regions. In region 1, the error estimation is reasonably accurate compared to the actual error although the adjoint correction changes significantly. The adjoint correction is not enough to capture the dominant variation of error estimation in this region, which is also affected by the remaining error term. In region 2, a good error estimate is primarily obtained from the adjoint correction, although the remaining error still contributes favourably. In region 3, the error estimate is less reliable, while the adjoint correction has

a trend similar to that of the total error estimation.



(a) AMR with full-order solutions (b) Error analyses with u from a fine mesh

Figure 4.7: Convergence of actual error, error estimation and adjoint correction during (a) AMR based on full-order primal solutions and (b) error analyses with u from a fine mesh (384 cells).

The computational mesh has few elements in region 1 and thus the accuracy of resolved solutions is limited. The VMM unresolved-scale model is thus vital for some cases in this region, leading to a significant influence of the remaining error. When combined with the adjoint correction, a good prediction is obtained. As the mesh is refined towards region 2, more of the solution is resolved and the adjoint correction begins to be dominant. This is actually the scenario we usually meet in literature or practical problems, i.e. we start AMR with a mesh that can capture important features based on experience and the adjoint correction is used to improve the computation of a QoI. In region 3, we reach a range of fine meshes with high spatial resolutions, where the computed QoI manifests an oscillatory convergence to the exact value. In this region, the remaining error is as important as the adjoint correction. Since the remaining error relies on the exact primal solution, u, the accuracy of the VMM unresolved-scale model can have a significant impact on its evaluation. Non-uniform meshes from AMR can affect the reliability of the unresolved-scale model due to its reliance on mesh size. When comparing the remaining error computed by VMM and an accurate reference u from a fine mesh (384 cells) as shown in Figure 4.7(b), we can see that VMM overestimates the value, leading to a less reliable error estimation in region 3. In spite of this, the approximation of the QoI in region 3 is still improved as the actual error keeps reducing. This may be because the relative size of local error contributions can be recognized by the error estimation.

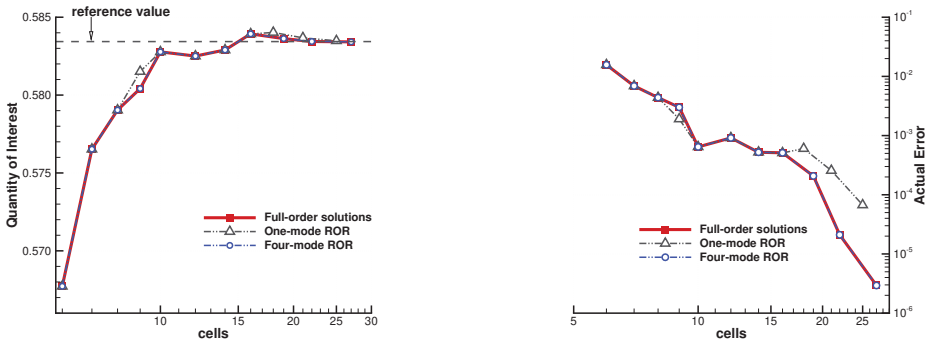
4.6. OFFLINE ROR-DRIVEN ADJOINT-BASED AMR

We consider the same problem used in Section 4.5 to investigate how the ROR influences the performance of adjoint-based AMR. Since the error estimation in Section 4.5 is computed from full-order primal solutions, we refer to it as the baseline AMR. As an alternative, two RORs are considered, one with four POD modes and the other with one POD mode. Naturally, a ROR will produce a good representation of the primal solution

if all significant POD modes are included. In this case, four POD modes are sufficient to capture 99.9% of the total energy (shown in Figure 4.2). In contrast, the one-mode ROR is an extreme case, with a less accurate reconstruction of the primal solution but high computing efficiency. The mesh adaptation starts with a coarse mesh of 6 cells and terminates after 10 AMR levels with the aforementioned strategy.

4.6.1. ROR WITH FOUR POD MODES

The computation of the QoI from ROR-driven AMR is compared to that of the baseline AMR in Figure 4.8(a). Both converge to the actual value in a quantitatively similar way, as do their approximation errors shown in Figure 4.8(b). Figure 4.9 demonstrates that the mesh refinement patterns computed based on the four-mode ROR agree well with those from the baseline AMR. Figure 4.10(a) depicts the development of cumulative energy within various AMR levels. Four POD modes can still capture more than 99.9% of the total energy even for the fine meshes obtained during later AMR levels. Thus, the four-mode ROR-driven AMR does not behave differently from the baseline AMR.



(a) Approximation of QoI

(b) Actual error

Figure 4.8: Grid convergence of (a) the QoI and (b) the associated error from AMR based on full-order solutions (—■—), a four-mode ROR (- - o - -) and a one-mode ROR (- - △ - -).

4.6.2. ROR WITH ONE POD MODE

We now study an extreme situation with only one mode used to construct the ROR. As before, Figure 4.8 compares values of the QoI and their corresponding errors. The QoI's approximations from a one-mode ROR are similar to those from the baseline AMR, although there are differences at some AMR levels. But this AMR method is still much better than uniform refinement. From the mesh refinement pattern shown in Figure 4.9, we can observe that the meshes obtained from the one-mode ROR AMR are not completely the same as those from the baseline AMR. The one-mode ROR changes the AMR sequence in coarse mesh regions at the 3rd AMR level, for example, but reaches the same computational mesh at the 4th AMR level. On the other hand, the actual error is affected by using only one mode as the mesh becomes fine in later AMR levels (see Figure 4.9).

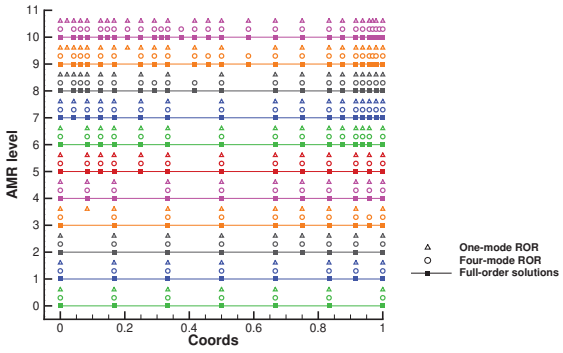
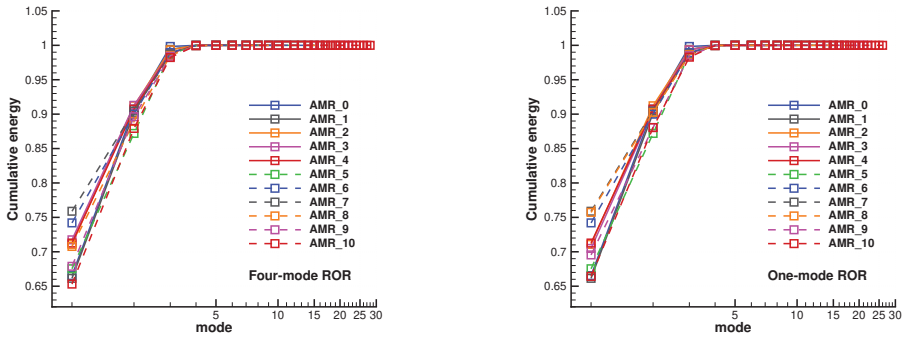


Figure 4.9: Mesh refinement pattern for AMR based on full-order primal solutions (—■—), a four-mode ROR (○) and a one-mode ROR (△). Different colors denote different AMR levels.



(a) Four-mode ROR

(b) One-mode ROR

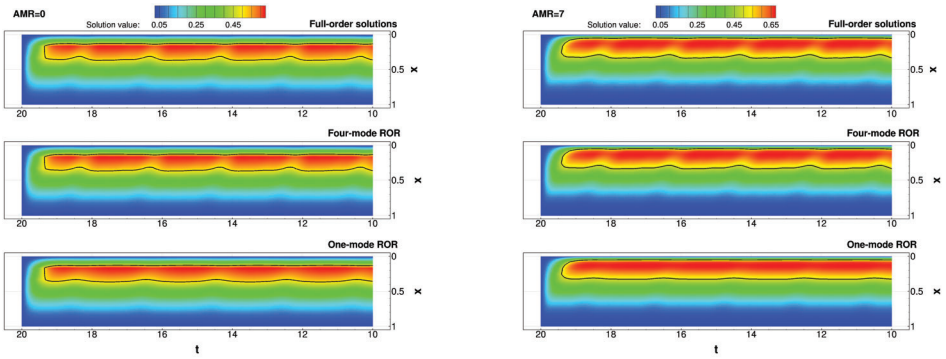
Figure 4.10: Cumulative energy on different AMR levels computed with (a) a four-mode ROR and (b) a one-mode ROR.

The cumulative energy for this one-mode ROR is shown in Figure 4.10(b). Since the first POD mode accounts for 65-75% of total energy during AMR, it can describe the main features of the primal solution across all AMR levels. Thus, the calculation from the one-mode ROR still considerably outperforms uniform refinement.

4.6.3. IMPACT OF ROR TRUNCATION ON ADJOINT SOLUTIONS AND ERROR INDICATORS

Figure 4.11 presents discrete adjoint solutions on two different computational meshes encountered during AMR, a coarse mesh in Figure 4.11(a) and a fine mesh in Figure 4.11(b). The adjoint solutions are calculated based on full-order primal solutions, a four-mode ROR and a one-mode ROR, respectively. On a coarse mesh, the one-mode ROR is able to

produce adjoint solutions with both features and magnitudes similar to those obtained using full-order solutions. This implies good error estimates. As the AMR proceeds to finer meshes, the one-mode ROR is unable to present high-wavenumber information and thus produces relatively smooth adjoint solutions, as in Figure 4.11(b) for instance. Thus the error estimation with the one-mode ROR is affected. Conversely, the four-mode ROR includes both low- and high-wavenumber information and thus provides good estimates over both meshes.



(a) Mesh at starting AMR level

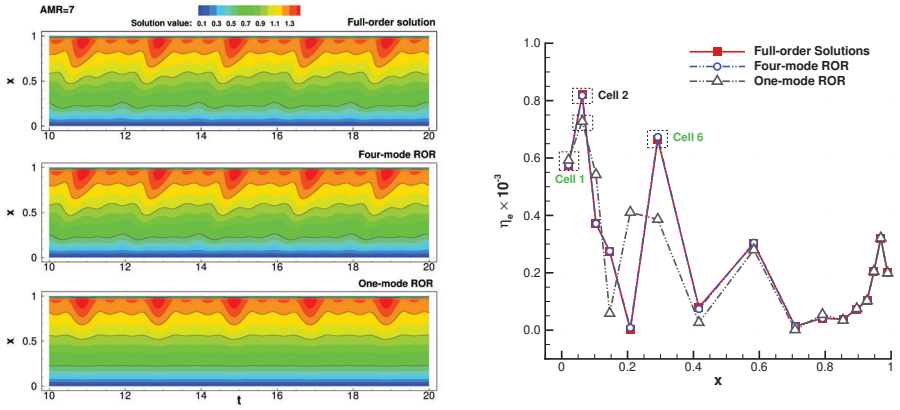
(b) Mesh at 7th AMR level

Figure 4.11: Discrete adjoint solutions on (a) a coarse mesh at starting AMR level and (b) a fine mesh at 7th AMR level, computed based on full-order primal solutions, a four-mode ROR and a one-mode ROR.

The primal solutions reconstructed from the RORs are also compared to the full-order one on the 7th AMR iteration in Figure 4.12(a). We can observe that the one-mode ROR is able to give a good prediction for oscillations with large magnitudes but filters medium- and small-amplitude oscillations. The associated adjoint solution reflects this characteristic and becomes smoother as well. In that sense, their elemental error indicators η_e exhibit different distributions, as shown in Figure 4.12(b). Based on full-order solutions, the algorithm marks cell 2 and cell 6 for refinement. However, the ones obtained using a one-mode ROR are cell 2 and cell 1. Consequently, a slightly different mesh is obtained in the one-mode ROR-driven AMR. This is because the high-wavenumber components of the primal solution play an important role in the error estimation on fine meshes, and those are not captured when using only one POD mode. It seems that using a very low-order ROR can be effective when the QoI's calculation is not dependent on smaller scales of the primal solution. As shown in Figure 4.12(b), a four-mode ROR, which includes higher-wavenumber features, is capable of reliably reconstructing the primal solution and the corresponding error indicators.

4.6.4. IMPACT OF ROR TRUNCATION ON ERROR ESTIMATION

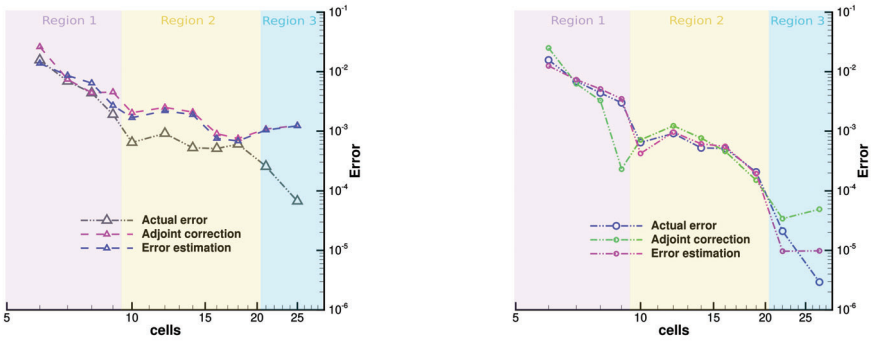
Figure 4.13 shows the change of error estimation and adjoint correction during AMR obtained using one-mode and four-mode RORs. The four-mode ROR-driven AMR has good error predictions in regions 1 and 2, but overestimates errors in region 3, as does the baseline AMR. The error estimation from the one-mode ROR is accurate in region 1



(a) Spatial distribution (b) Error indicator

Figure 4.12: Comparison of (a) reconstructed primal solutions and (b) error indicators at 7th AMR level with full-order primal solutions, a four-mode ROR and a one-mode ROR.

but overestimates errors in regions 2 and 3. Still, the first POD mode is capable of capturing the dominant solution feature crucial in region 2, and thus gives a sufficiently accurate estimate for AMR. But this advantage vanishes in region 3 where the high-frequency spatial information becomes important as well. The adjoint correction is the dominant error estimation term in the last two regions and is affected by the filtering of higher wavenumbers.



(a) One-mode (b) Four-mode

Figure 4.13: Actual and estimated errors computed by ROR-driven AMR based on (a) one mode and (b) four modes.

4.6.5. IMPACT OF ROR TRUNCATION ON MEMORY EFFICIENCY

We define an efficiency metric, η, to quantify the memory reduction when utilizing a ROR. The memory required to store full-order primal solutions is proportional to $N_{\text{full}} =$

$N_v N_t$ while a ROR needs $N_{ROR} = MN_v + N_v + MN_t$, where N_v and N_t denotes the number of variables and time steps respectively. The efficiency metric is then defined as

$$\eta = 1 - \frac{N_{ROR}}{N_{full}} = 1 - \frac{N_v + N_t}{N_v N_t} M - \frac{1}{N_t}. \tag{4.17}$$

A ROR requires less memory while $\eta > 0$, and vice versa. Furthermore, larger η indicates higher ROR efficiency, with the asymptote of $\eta_{asy} = 1 - \frac{1}{N_t}$. Figure 4.14 shows the variation of this metric for one- and four-mode RORs during AMR. It is observed that the efficiency increases monotonically for both RORs as the mesh is refined, leading to a significant memory reduction. The one-mode ROR, as expected, is more memory efficient than the four-mode ROR while their difference is reduced as the mesh is refined.

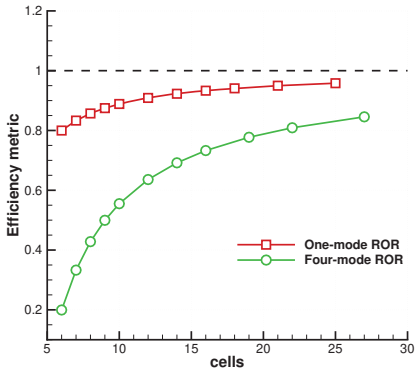
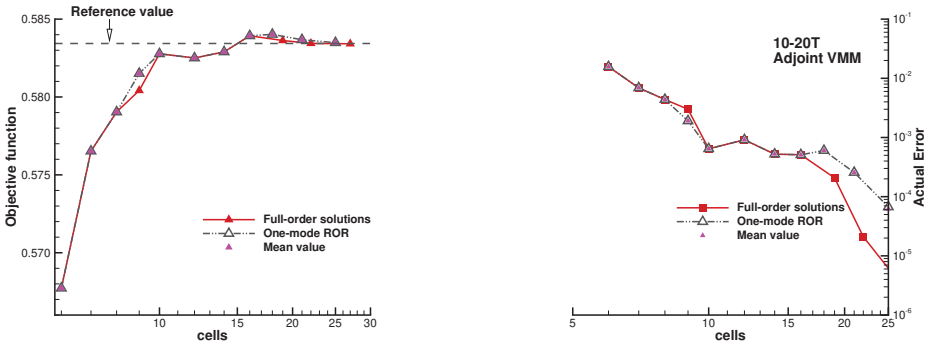


Figure 4.14: Comparisons of the efficiency metric in one- and four-mode RORs during AMR ($N_t = 10001$).

4.6.6. IMPACT OF MEAN VALUE ON ADJOINT-BASED AMR

As the one-mode ROR showed a similar convergence trend for the computation of the QoI, we further investigate the influence of mean values (a zero-mode ROR) on the adjoint-based AMR. There are two ways to do this, solving the unsteady adjoint problems with mean value or solving the steady adjoint problem. We study it in the first way since the second can be achieved by the steady state in the first one. Figure 4.15 show the convergence of QoI and associated errors as the mesh is refined by using adjoint-based AMR with full-order solutions, one-mode ROR, and mean values. We can observe that the results computed based on mean values converge in the same way compared to the one from a one-mode ROR. So do the actual errors. We can see that the mean value plays an important role in the computation of QoI in this case. This also explains why the one-mode ROR can give an accurate evaluation during AMR. This benefit results from the type of chosen QoIs that could be different in realistic LES.

The adapted meshes based on mean-value adjoint-based AMR are presented in Figure 4.16. It is observed that all adapted meshes are the same except for the terminating one (near the right boundary). It actually becomes a pretty fine mesh at the terminating AMR level and the high-frequency spatial information makes a significant difference



(a) Approximation of QoI (b) Actual error

Figure 4.15: The computation of QoI and associated errors during AMR based on full-order solutions, one-mode ROR and mean values.

in computing the QoI. This is consistent with behavior of AMR with a one-mode ROR. The adjoint solution at starting and 7th AMR levels are compared as well, as shown in Figure 4.17. Using the mean value for solving the adjoint problem can maintain major structures of adjoint fields although producing a time-invariant adjoint solution after a short transition. This is because the fluctuations of adjoint solutions based on the full-order solution change locally and the sensitive regions almost keep at a constant range, making it less challenging even for the case with mean values. More realistic turbulent cases should be considered to validate the effectiveness of this proposed method.

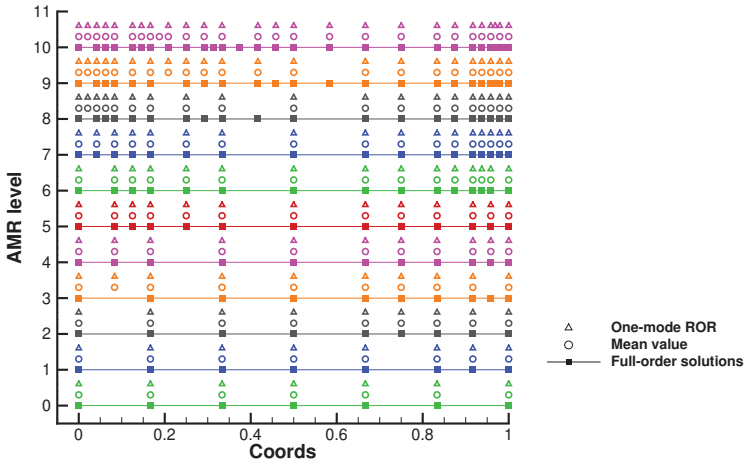
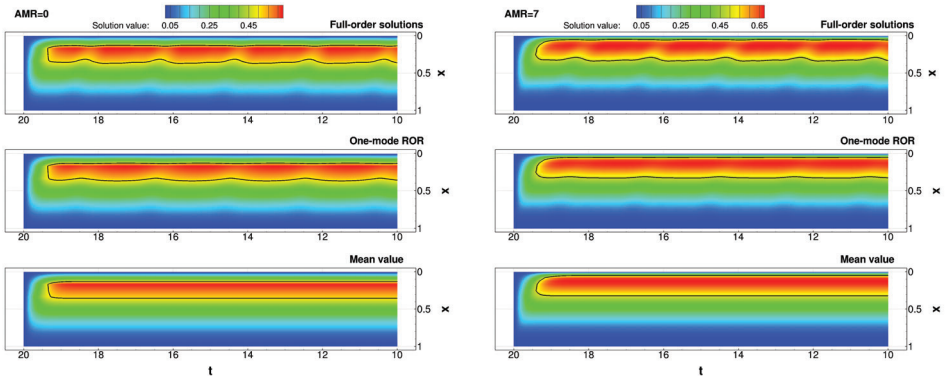


Figure 4.16: Comparisons of mesh refinement patterns based on mean values to the one of full-order solutions and one-mode ROR.



(a) Starting AMR level (b) 7th AMR level

Figure 4.17: Comparison of adjoint solutions computed based on full-order solutions, one-mode ROR and mean values at (a) starting and (b) 7th AMR level.

4.7. ONLINE ROR FOR ADJOINT-BASED AMR

In this section, we investigate the use of the EOA ROR and apply it to the same Burgers problem considered in Section 4.6. The online ROR without the enhanced process is referred to as the standard online ROR herein.

4.7.1. IMPACT OF THE EOA ROR

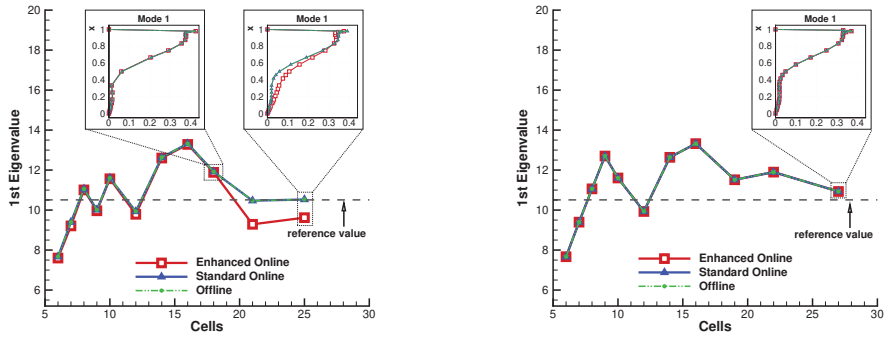
ON POD ANALYSIS

Figure 4.18 compares first eigenvalues from the offline, standard online and EOA RORs, as well as the shapes of their associated POD modes. The standard online ROR can produce results identical to those from an offline ROR using both one and four POD modes, validating the standard incremental algorithm.

The eigenvalues and POD modes from a one-mode EOA ROR agree well with the standard online ROR except for the last two AMR levels. This reflects the characteristics mentioned in region 3, where high-wavenumber features become important on fine meshes. The truncation of higher-order POD modes during the single-mode EOA removes many of these features. Additionally, this truncation also has a notable impact on the distribution of POD modes at the final AMR level as shown in Figure 4.18(a). In contrast, using the four-mode EOA ROR has no detrimental effect on both the eigenvalues and POD modes, as shown in Figure 4.18(b). Therefore, one way to improve the representation of model interactions is to increase the number of POD modes used for the identification of EOA ROR.

ON THE APPROXIMATION OF A QOI

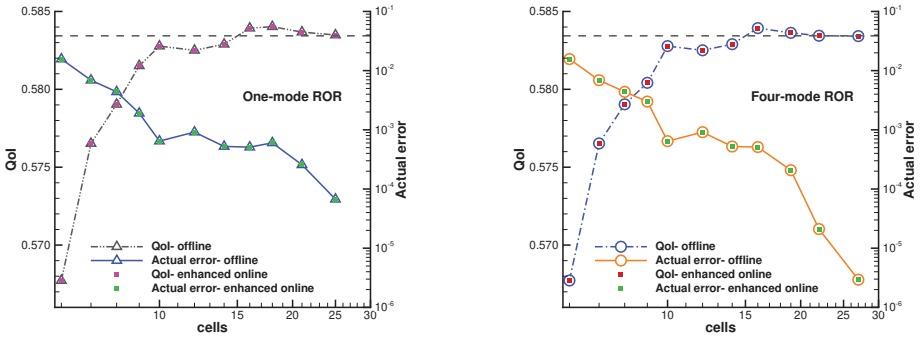
The approximations of the QoI and its error from EOA ROR are presented in Figure 4.19. We can observe that the EOA ROR does not alter the convergence history of the computation, preserving the same accuracy of calculations from the offline ROR-driven AMR



(a) One-mode (b) Four-mode

Figure 4.18: Mesh convergence of the first eigenvalue and first POD mode for enhanced online (red), standard online (blue), and offline (green) ROR with (a) one and (b) four modes.

technique, even for the one-mode case. Although the POD mode obtained from the one-



(a) One-mode (b) Four-mode

Figure 4.19: Comparisons of QoI and actual error during AMR using enhanced online and offline ROR with (a) one and (b) four modes.

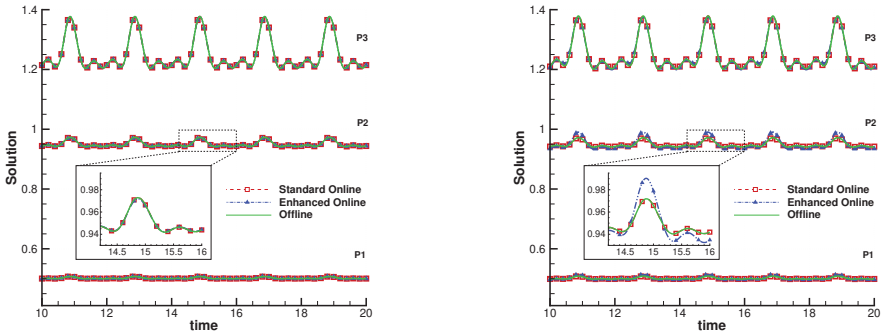
mode EOA ROR differs from that obtained with the offline ROR, the resulting QoI is still reasonably accurate, as shown in Figure 4.19(a). In fact, the mesh refinement patterns are the same as those obtained with the offline SVD.

4.7.2. DISCUSSIONS OF THE EOA ROR

RECONSTRUCTED PRIMAL SOLUTION

In order to analyze the mechanism for this phenomenon, we choose three spatial positions with small (P1), medium (P2) and large (P3) mean values, as shown in Figure 4.20. The QoI's good agreement is partly because the reconstructions of primal solutions from the offline ROR and EOA ROR have similar trends over time. This is true even at the 10th

AMR level, as presented in Figure 4.20(b), although there are differences in terms of their amplitudes at P2 and P1. The discrepancies result from only using one POD mode, which by design attempts to approximate instantaneous primal solutions with large dominant amplitudes and thus needs to sacrifice some accuracy of medium- and small-amplitude oscillations. Note that since the QoI is a statistical value over time, its calculation can benefit from the cancellation of temporal fluctuations. In addition, the EOA ROR provides the same accuracy as the standard online ROR until the 8th AMR level, as shown in Figure 4.20(a).



(a) 8th AMR level (b) 10th AMR level

Figure 4.20: Reconstructed primal solutions from standard online (red), enhanced online (blue), and offline (green) ROR with one POD mode at the 8th and 10th AMR level. P1, P2 and P3 denote three different spatial positions, i.e. P1 ($x = 1/6$), P2 ($x = 1/2$), P3 ($x = 5/6$).

THE STATISTICAL VALUE OF A QOI

Taking the QoI used here for an example, we can calculate it as $\int_\Omega \sin(\pi x)\,\bar{u}_i\,\mathrm{d}\Omega$, where $\bar{u}_i$ denotes the time-averaged solution. The QoI will be identical as long as the time-averaged value $\bar{u}_i$ stays the same. Since the one-mode ROM preserves the dominant large-scale features, the mean value can be predicted reliably as shown in Table 4.1.

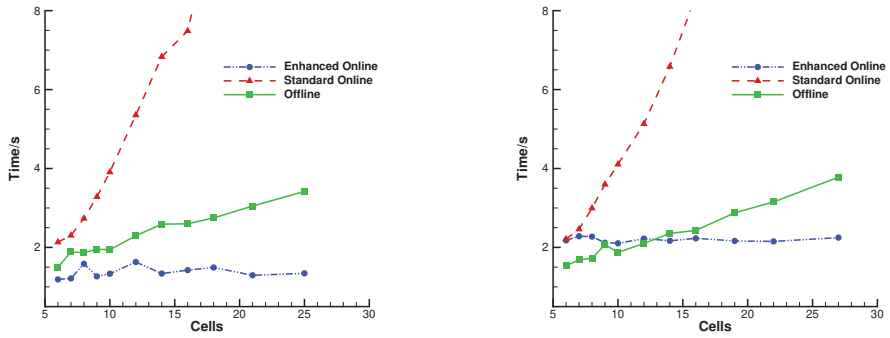
Table 4.1: The comparison of mean values at three spatial positions between standard ROM and enhanced online ROM

Spatial Positions	Standard ROM	Enhanced Online ROM
P1	0.5012696	0.5012696
P2	0.9489279	0.9489279
P3	1.249128	1.249128

4.7.3. COMPARISON OF COMPUTING TIME

Figure 4.21 compares the computing time to build a ROR by the offline, standard online and enhanced online algorithms during AMR. Figure 4.21(a) shows this computing time

for a one-mode ROR. The standard online ROR requires more computing time than the offline ROR. In contrast, the EOA ROR is much faster, and the advantage becomes more and more apparent as the mesh is refined. This property is also observed for the four-mode ROR, as shown in Figure 4.21(b), although it is less efficient at the starting AMR levels. Generally, the computing time of the offline and standard online ROR increases as the mesh is refined. However, the computing time of the EOA ROR grows much more slowly. The results demonstrate that the EOA ROR could be promising for realistic LES applications.



(a) One-mode (b) Four-mode

Figure 4.21: Computing time of constructing offline (green), standard online (red) and enhanced online (blue) ROR with (a) one and (b) four POD modes during AMR.

4.8. SUMMARY

We investigated the performance of the adjoint-based method on a 1D LES model problem based on the Burgers equation with a multi-frequency forcing term, and studied the impact of using offline and online RORs on the proposed method. The main findings are summarized as follows.

- The effectiveness of combining adjoint-based error estimation with VMM has been verified using an unsteady Burgers problem with a manufactured solution.
- A POD-based ROR is introduced to represent the primal solution from the nonlinear unsteady simulation so as to remove the storage bottleneck that would otherwise occur when we solve the adjoint problem backward in time. The procedure is evaluated using a Burgers problem with a multi-frequency forcing term. From numerical experiments, adjoint-based AMR is shown to be more efficient than traditional uniform refinement.
- The results show that AMR can start from very coarse meshes. The QoI estimate converges reliably during AMR, and reaches a high level of accuracy at moderate levels of refinement. For the cases considered here, using the ROR for primal solutions does not significantly affect the performance of AMR. Specifically, the adaptive results from a four-mode ROR, which can capture 99.9% of total energy, have

good agreement with the results from a full-order solution-driven AMR. Using a single-mode ROR leads to suboptimal meshes since the first POD mode accounts for only 65-75% of total energy, but the AMR procedure still considerably outperforms uniform refinement.

- Incremental SVD is introduced to build a ROR on the fly and an enhanced online algorithm is proposed to further improve the efficiency of building a ROR so that it can be applied for realistic LES. The results show that the enhanced algorithm does not affect the quick convergence of the QoI computed by ROR-driven adjoint-based AMR. The time consumption of building a ROR is significantly reduced by using the EOA.

5

APPLICATIONS OF ROR FOR UNSTEADY ADJOINT METHOD IN 2D/3D PROBLEMS

Using the enhanced online algorithm, a reduced-order representation of the primal solution is effectively built for high-dimensional 2D/3D problems here. The impact of RORs on solving the adjoint problem is studied for unsteady flows past a 2D/3D circular cylinder, in terms of accuracy, efficiency and stability.

5.1. INTRODUCTION

In Chapter 4, we demonstrated the effectiveness of using reduced-order representations (RORs) for solving the unsteady adjoint problem of the 1D forced Burgers problem. Since POD techniques are not affected by the dimension of the physical domain, it can be expected that the efficiency of the enhanced online algorithm will translate to higher-dimensional problems. Nevertheless, the applicability of the proposed method to multidimensional applications needs to be investigated since the physical characteristics become more complicated and challenging in 2D and 3D flow problems.

In this chapter, we examine the effects of using a primal ROR on the accuracy of adjoint solutions and adjoint-based error estimates for unsteady 2D and 3D flow problems. The development of the adjoint system for 2D/3D cases is presented in Section 5.2, including the associated boundary conditions. The enhanced online algorithm is coupled with OpenFOAM in Section 5.3, which provides the order-reduction technique for solving the unsteady adjoint problem. The impact of RORs on the adjoint problem is studied for 2D circular cylinder flows, at $Re = 100$ and $Re = 500$, and a 3D circular cylinder flow at $Re = 500$. We examine the impacts on the adjoint field in Section 5.4, the computing accuracy and efficiency in Section 5.5, and on the adjoint-based error estimation of the drag in Section 5.6. The impact of using RORs on the adjoint stability of turbulent flows is also discussed.

5.2. ADJOINT FORMULATION FOR UNSTEADY 2D/3D FLOWS

Although the continuous and discrete adjoint methods are studied for error estimation in the literature, the continuous adjoint method is considered here. This is because the continuous adjoint method has been used widely in OpenFOAM [139–141] and it allows us to implement the algorithm efficiently. The adjoint system is derived for incompressible flows with different boundary conditions. Although the derivation is demonstrated for cylinder flows, the procedure can be applied to other applications starting from the general expressions used here.

5.2.1. PRIMAL FLOW MODEL

We consider the unsteady Navier–Stokes equations for incompressible flows,

$$\begin{aligned} R^p &= -\frac{\partial u_j}{\partial x_j} = 0 \\ R_i^u &= \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right) = 0\,, \end{aligned} \tag{5.1}$$

where $i, j = 1,2$ for 2D problems or $i, j = 1,2,3$ for 3D problems. ν represents the dynamic viscosity coefficient and u_i and p are flow velocity and pressure.

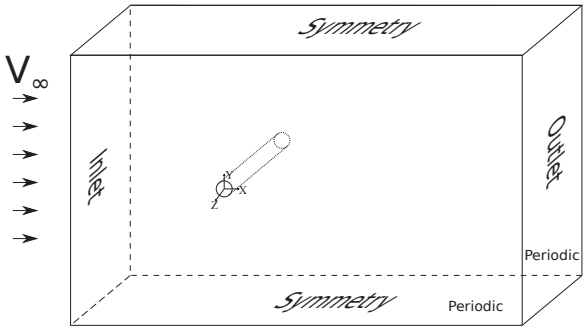


Figure 5.1: The computational geometry and domain of a cylinder.

We consider flows past a cylinder as shown in Figure 5.1. On the upper and lower sides of the domain, symmetry conditions are applied. Periodic boundary conditions are applied in spanwise direction. A constant free stream velocity is imposed at the inlet while a pressure boundary condition is imposed at the outlet. On the surface of the cylinder, a no-slip boundary condition is prescribed. The values used for these conditions are summarized in Table 5.1.

Table 5.1: Velocity (u_i) and pressure values for boundary conditions used in a cylinder flow problem

Primal variables	Inlet	Outlet	No-slip	Free-slip	Periodic
Velocity ($u_i, i = 1,2,3$)	$u_i = u_i^0$	$\frac{\partial u_i}{\partial n} = 0$	$u_i = 0$	$u_j n_j = 0$	$u_i = u_i^p$
Pressure (p)	$\frac{\partial p}{\partial n} = 0$	$p = 0$	$\frac{\partial p}{\partial n} = 0$	$\frac{\partial p}{\partial n} = 0$	$p = p^p$

5.2.2. ADJOINT SYSTEM

In unsteady flow problems, we are often interested in time-averaged QoIs, such as lift and drag. In general, such a QoI can be expressed as

$$J = \frac{1}{T}\int_I\int_\Omega J_\Omega \,\mathrm{d}\Omega\mathrm{d}t + \frac{1}{T}\int_I\int_{\partial\Gamma} J_\Gamma \,\mathrm{d}\Gamma\,\mathrm{d}t, \tag{5.2}$$

where the J_Ω and J_Γ denote the contribution to the QoI per volume and surface area respectively. $\partial\Gamma$ denotes the boundary, which can be just part of the boundary of the fluid domain Ω. T represents the time length for averaging the QoI over a temporal space I. This formula can be used for different types of QoIs although their specific expressions depend on the problems considered.

The Lagrange equation is introduced as

$$L = L(v_i, p, u_i, q; \beta) = J + \frac{1}{T}\int_I\int_\Omega (qR^p + v_i R_i^u)\,\mathrm{d}\Omega\,\mathrm{d}t, \tag{5.3}$$

where v_i, q are Lagrange (adjoint, co-state [142]) variables, and β denotes design variables that can be changed by users. Design variables can be parameters for changing geometry in optimization problems or mesh size during AMR, for instance.

Given that the flow variables (i.e. u_i, p) are controlled by the design variables, there are three categories of independent variables in the Lagrange function, i.e. β, v_i, q. Thus the variation of Lagrange function, δL, is expressed as

$$\delta L = \frac{\delta L}{\delta \beta}\delta\beta + \frac{\delta L}{\delta v_i}\delta v_i + \frac{\delta L}{\delta q}\delta q\,. \tag{5.4}$$

The solution of the Lagrange function requires

$$\delta L = 0, \quad \forall \delta\beta, \delta v_i, \delta q\,. \tag{5.5}$$

This is equivalent to

$$\frac{\delta L}{\delta \beta}\delta\beta = 0\,, \tag{5.6a}$$

$$\frac{\delta L}{\delta v_i}\delta v_i = 0\,, \tag{5.6b}$$

$$\frac{\delta L}{\delta q}\delta q = 0\,, \tag{5.6c}$$

$\forall \delta\beta, \delta v_i, \delta q$. Equations (5.6b) and (5.6c) are satisfied because of the conservation of mass and momentum (i.e. Equation (5.1)). Therefore, the condition Equation (5.6a) is what needs to be solved, which defines the associated adjoint problem. If the L only involves algebraic operations, this extrema condition is stated as

$$\frac{\delta L}{\delta \beta} = \frac{\partial L}{\partial \beta} = 0\,. \tag{5.7}$$

Otherwise, we need to consider the variation for PDEs rather than simply computing derivatives. Considering Equation (5.3), we can derive the adjoint problem by

$$\begin{aligned}\frac{\delta L}{\delta \beta} &= \frac{\delta J}{\delta \beta} + \frac{1}{T}\int_I\int_\Omega \frac{\delta(qR^p + v_i R_i^u)}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t && \triangleright \frac{\delta \mathrm{d}\Omega}{\delta\beta} = 0 \\ &= \underbrace{\frac{\delta J}{\delta\beta}}_{\mathrm{I}} + \underbrace{\frac{1}{T}\int_I\left[\int_\Omega q\frac{\delta R^p}{\delta\beta}\,\mathrm{d}\Omega + \int_\Omega v_i\frac{\delta R_i^u}{\delta\beta}\,\mathrm{d}\Omega\right]\mathrm{d}t}_{\mathrm{II}} && \triangleright q, v_i \text{ independent of } \beta\,.\end{aligned} \tag{5.8}$$

The order of variation and integration were exchanged above since the integration can be regarded as a summation and the domain is not changed when considering error estimations (i.e. $\frac{\delta \mathrm{d}\Omega}{\delta\beta} = 0$). The terms I denotes the component from the QoI and II is that from physical constraints. These will be derived separately in the subsequent subsections.

QoI's contribution Term I is expressed as

$$\mathrm{I} = \frac{\delta J}{\delta\beta} = \frac{1}{T}\int_I\left[\int_\Omega \frac{\delta J_\Omega}{\delta\beta}\,\mathrm{d}\Omega + \int_{\partial\Gamma}\frac{\delta J_\Gamma}{\delta\beta}\,\mathrm{d}\Gamma\right]\mathrm{d}t\,, \tag{5.9}$$

where

$$\begin{aligned}\int_\Omega \frac{\delta J_\Omega}{\delta\beta}\,\mathrm{d}\Omega &= \int_{\partial\Omega}\Big(\frac{\partial J_\Omega}{\partial\beta}+\frac{\partial J_\Omega}{\partial u_i}\frac{\delta u_i}{\delta\beta}+\frac{\partial J_\Omega}{\partial p}\frac{\delta p}{\delta\beta}\Big)\,\mathrm{d}\Omega\\ \int_{\partial\Gamma} \frac{\delta J_\Gamma}{\delta\beta}\,\mathrm{d}\Gamma &= \int_{\partial\Gamma}\left(\frac{\partial J_\Gamma}{\partial\beta}+\frac{\partial J_\Gamma}{\partial u_i}\frac{\delta u_i}{\delta\beta}+\frac{\partial J_\Gamma}{\partial p}\frac{\delta p}{\delta\beta}+\frac{\partial J_\Gamma}{\partial u_{i,j}}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\right)\mathrm{d}\Gamma\,.\end{aligned} \tag{5.10}$$

$\frac{\partial J_\Gamma}{\partial u_{i,j}}$ denotes the contribution from velocity gradients, such as the stress tensor of velocity. $\frac{\delta}{\delta\beta}$ is regarded as differential when operating on derivatives. The variation ($\frac{\delta}{\delta\beta}$) is treated normally for algebraic operations using the chain rule while the variation is applied directly to the whole term involving derivatives. The chain rule is applied to the velocity u_i rather than gradient operators or tensors. Taking the drag for an instance, the drag is expressed by the gradient of velocity, where the $\frac{\delta}{\delta\beta}$ is applied inside the gradient. Therefore, $\frac{\partial J_\Gamma}{\partial u_{i,j}}$ denotes the counterpart to the velocity tensor during the derivation rather than an operator for computing derivatives.

Volume integration The term II includes volume integration which we split into two components for convenience

$$\mathrm{II} = \underbrace{\frac{1}{T}\int_I\int_\Omega q\frac{\delta R^p}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t}_{\mathrm{II}_A} + \underbrace{\frac{1}{T}\int_I\int_\Omega v_i\frac{\delta R_i^u}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t}_{\mathrm{II}_B}\,. \tag{5.11}$$

Integration-by-parts is applied, treating each index as an independent variable. The term II_A involves in the continuity equation and is re-expressed as

$$\begin{aligned}\mathrm{II}_A &= \frac{1}{T}\int_I\int_\Omega q\frac{\delta R^p}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t = \frac{1}{T}\int_I\int_\Omega q\frac{\delta}{\delta\beta}\Big(-\frac{\partial u_i}{\partial x_i}\Big)\,\mathrm{d}\Omega\,\mathrm{d}t && \triangleright \text{ Equation (5.1)}\\ &= \frac{1}{T}\int_I\int_\Omega -q\frac{\partial}{\partial x_i}\Big(\frac{\delta u_i}{\delta\beta}\Big)\,\mathrm{d}\Omega\,\mathrm{d}t\\ &= \frac{1}{T}\int_I\left[\int_{\partial\Omega} -q\frac{\delta u_i}{\delta\beta}n_i\,\mathrm{d}\Gamma+\int_\Omega\frac{\partial q}{\partial x_i}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega\right]\mathrm{d}t && \triangleright \text{ Integration-by-parts}.\end{aligned} \tag{5.12}$$

Term II_B involves the momentum equations and is stated as

$$\begin{aligned}\mathrm{II}_B &= \frac{1}{T}\int_I\int_\Omega v_i\frac{\delta R_i^u}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t\\ &= \frac{1}{T}\int_I\int_\Omega v_i\frac{\delta}{\delta\beta}\left(\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}+\frac{\partial p}{\partial x_i}-\frac{\partial\tau_{ij}}{\partial x_j}\right)\mathrm{d}\Omega\,\mathrm{d}t\\ &= \frac{1}{T}\int_I\int_\Omega \underbrace{v_i\frac{\partial}{\partial t}\left(\frac{\delta u_i}{\delta\beta}\right)}_{\mathrm{II}_{B,\text{temporal}}}\mathrm{d}\Omega\,\mathrm{d}t\\ &+\frac{1}{T}\int_I\int_\Omega\Big(\underbrace{v_j\frac{\partial u_j}{\partial x_i}\frac{\delta u_i}{\delta\beta}}_{\mathrm{II}_{B.0}}+\underbrace{v_iu_j\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)}_{\mathrm{II}_{B.1}}+\underbrace{v_i\frac{\partial}{\partial x_i}\left(\frac{\delta p}{\delta\beta}\right)}_{\mathrm{II}_{B.2}}\underbrace{-v_i\frac{\partial}{\partial x_j}\left(\frac{\delta\tau_{ij}}{\delta\beta}\right)}_{\mathrm{II}_{B.3}}\Big)\mathrm{d}\Omega\,\mathrm{d}t\,.\end{aligned} \tag{5.13}$$

It is noted that the index is switched in $\mathrm{II}_{B.0}$ so that we have a consistent expression of velocity variation. The term II_B is further split into a temporal part, $\mathrm{II}_{B,\text{temporal}}$, and a spatial part, $\mathrm{II}_{B,\text{spatial}}$, with $\mathrm{II}_{B,\text{spatial}} = \mathrm{II}_{B.0} + \mathrm{II}_{B.1} + \mathrm{II}_{B.2} + \mathrm{II}_{B.3}$, which will be discussed separately.

After integration-by-parts is applied in temporal direction for $\mathrm{II}_{B,\text{temporal}}$, we obtain

$$\begin{aligned}\mathrm{II}_{B,\text{temporal}} &= \frac{1}{T}\int_I\int_\Omega v_i \frac{\partial}{\partial t}\left(\frac{\delta u_i}{\delta\beta}\right)\mathrm{d}\Omega\,\mathrm{d}t \\ &= \frac{1}{T}\int_\Omega \left(v_i\frac{\delta u_i}{\delta\beta}\Big|_{t_\text{start}}^{t_\text{end}} - \int_I \frac{\partial v_i}{\partial t}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}t\right)\mathrm{d}\Omega \\ &= \frac{1}{T}\int_\Omega v_i\frac{\delta u_i}{\delta\beta}\Big|_{t_\text{start}}^{t_\text{end}}\mathrm{d}\Omega - \frac{1}{T}\int_I\int_\Omega \frac{\partial v_i}{\partial t}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t.\end{aligned} \tag{5.14}$$

Likewise, integration-by-parts is applied in the spatial directions for $\mathrm{II}_{B,\text{spatial}}$ while the temporal integration is neglected. The term $(\mathrm{II}_{B.0})$ can be used directly since the $\frac{\delta u_j}{\delta\beta}$ is outside of derivatives. The remaining terms are derived using the integration-by-parts as below,

$$\mathrm{II}_{B.1} = \int_\Omega v_i u_j \frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\mathrm{d}\Omega = \int_{\partial\Omega} v_i u_j \frac{\delta u_i}{\delta\beta} n_j\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial(v_i u_j)}{\partial x_j}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega \tag{5.15}$$

$$\mathrm{II}_{B.2} = \int_\Omega v_i \frac{\partial}{\partial x_i}\left(\frac{\delta p}{\delta\beta}\right)\mathrm{d}\Omega = \int_{\partial\Omega} v_i \frac{\delta p}{\delta\beta} n_i\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial v_i}{\partial x_i}\frac{\delta p}{\delta\beta}\,\mathrm{d}\Omega \tag{5.16}$$

$$\begin{aligned}\mathrm{II}_{B.3} &= \int_\Omega -v_i\frac{\partial}{\partial x_j}\left(\frac{\delta\tau_{ij}}{\delta\beta}\right)\mathrm{d}\Omega \\ &= \int_{\partial\Omega} -v_i\frac{\delta\tau_{ij}}{\delta\beta}n_j\,\mathrm{d}\Gamma + \int_\Omega \frac{\partial v_i}{\partial x_j}\frac{\delta\tau_{ij}}{\delta\beta}\,\mathrm{d}\Omega \\ &= \int_{\partial\Omega} -v_i n_j \nu\left[\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)+\frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta\beta}\right)\right]\mathrm{d}\Gamma + \int_\Omega \frac{\partial v_i}{\partial x_j}\nu\left[\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)+\frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta\beta}\right)\right]\mathrm{d}\Omega \\ &= \int_{\partial\Omega} -v_i n_j \nu\left[\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)+\frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta\beta}\right)\right]\mathrm{d}\Gamma \\ &\quad + \int_{\partial\Omega}\nu n_j\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Gamma - \int_\Omega \frac{\partial}{\partial x_j}\left(\nu\frac{\partial v_i}{\partial x_j}+\nu\frac{\partial v_j}{\partial x_i}\right)\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega.\end{aligned} \tag{5.17}$$

The sensitivities of primal variables $\left(\frac{\delta u_i}{\delta\beta}, \frac{\delta u_j}{\delta\beta}\right)$ are isolated in these terms. Finally, the

expression of the term II_B is expressed as

$$\begin{aligned}
\mathrm{II}_B &= \mathrm{II}_{B,\text{temporal}} + \mathrm{II}_{B.0} + \mathrm{II}_{B.1} + \mathrm{II}_{B.2} + \mathrm{II}_{B.3} \\
&= \frac{1}{T}\int_\Omega v_i \frac{\delta u_i}{\delta\beta}\Big|_{t_{\text{start}}}^{t_{\text{end}}} \mathrm{d}\Omega - \underline{\frac{1}{T}\int_I\int_\Omega \frac{\partial v_i}{\partial t}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega\,\mathrm{d}t} \\
&+ \frac{1}{T}\int_I \Bigg\{ \underline{\int_\Omega v_j \frac{\partial u_j}{\partial x_i}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega} + \int_{\partial\Omega} v_i u_j \frac{\delta u_i}{\delta\beta} n_j\,\mathrm{d}\Gamma - \underline{\int_\Omega \frac{\partial(v_i u_j)}{\partial x_j}\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega} \\
&+ \int_{\partial\Omega} v_i \frac{\delta p}{\delta\beta} n_i\,\mathrm{d}\Gamma - \underline{\int_\Omega \frac{\partial v_i}{\partial x_i}\frac{\delta p}{\delta\beta}\,\mathrm{d}\Omega} + \int_{\partial\Omega} -v_i n_j \nu\left[\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta\beta}\right)\right]\mathrm{d}\Gamma \\
&+ \int_{\partial\Omega} \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Gamma - \underline{\int_\Omega \frac{\partial}{\partial x_j}\left(\nu\frac{\partial v_i}{\partial x_j} + \nu\frac{\partial v_j}{\partial x_i}\right)\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Omega}\Bigg\}\mathrm{d}t,
\end{aligned} \tag{5.18}$$

where the underlined terms involve volume integrations while the remaining terms involve boundary integrations. All above-mentioned terms are summarized in Figure 5.2.

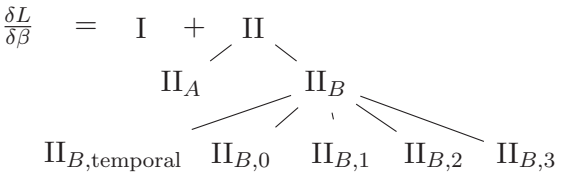


Figure 5.2: The expansion of the variation of the Lagrange function.

Then we first expand the term II as

$$\mathrm{II} = \mathrm{II}_A + \mathrm{II}_B = \mathrm{II}^\Omega + \mathrm{II}^{\partial\Omega}, \tag{5.19}$$

where II^Ω and $\mathrm{II}^{\partial\Omega}$ denote those terms involving volume and boundary integrations, respectively. They are defined as

$$\begin{aligned}
\mathrm{II}^\Omega &= \frac{1}{T}\int_I\int_\Omega \left(-\frac{\partial v_i}{\partial x_i}\frac{\delta p}{\delta\beta} + \left[-\frac{\partial v_i}{\partial t} + v_j\frac{\partial u_j}{\partial x_i} - \frac{\partial(v_i u_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) + \frac{\partial q}{\partial x_i}\right]\frac{\delta u_i}{\delta\beta}\right)\mathrm{d}\Omega\,\mathrm{d}t \\
&= \frac{1}{T}\int_I\int_\Omega \left(R^q\frac{\delta p}{\delta\beta} + R_i^v\frac{\delta u_i}{\delta\beta}\right)\mathrm{d}\Omega\,\mathrm{d}t,
\end{aligned} \tag{5.20}$$

$$\begin{aligned}
\mathrm{II}^{\partial\Omega} &= \frac{1}{T}\int_I\Bigg[\int_{\partial\Omega} v_i n_i\frac{\delta p}{\delta\beta}\,\mathrm{d}\Gamma + \int_{\partial\Omega}\left(-q n_i + v_i u_j n_j + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right)\frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Gamma \\
&\quad + \int_{\partial\Omega} -\nu\left(v_i n_j + v_j n_i\right)\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\mathrm{d}\Gamma t\Bigg]\mathrm{d}t + \frac{1}{T}\int_\Omega v_i\frac{\delta u_i}{\delta\beta}\Big|_{t_{\text{start}}}^{t_{\text{end}}}\mathrm{d}\Omega \\
&= \frac{1}{T}\int_I\int_{\partial\Omega}\left(D^q\frac{\delta p}{\delta\beta} + D_i^v\frac{\delta u_i}{\delta\beta} + P_{ij}^v\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\right)\mathrm{d}\Gamma\,\mathrm{d}t + \frac{1}{T}\int_\Omega v_i\frac{\delta u_i}{\delta\beta}\Big|_{t_{\text{start}}}^{t_{\text{end}}}\mathrm{d}\Omega,
\end{aligned} \tag{5.21}$$

where the notations of $R^q, R_i^v, D^q, D_i^v, P_{ij}^v$ are defined by

$$R^q = -\frac{\partial v_i}{\partial x_i} \tag{5.22a}$$

$$R_i^v = -\frac{\partial v_i}{\partial t} + v_j \frac{\partial u_j}{\partial x_i} - \frac{\partial (v_i u_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) + \frac{\partial q}{\partial x_i} \tag{5.22b}$$

$$D^q = v_i n_i \tag{5.22c}$$

$$D_i^v = -q n_i + v_i u_j n_j + \nu n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{5.22d}$$

$$P_{ij}^v = -\nu(v_i n_j + v_j n_i). \tag{5.22e}$$

It is noted that the adjoint systems including the adjoint equations and boundary conditions, can be different if we derive them using a conservative expression of the convective term in Equation (5.1). In fact, this expression can be obtained by applying integration-by-parts on $\mathrm{II}_{B.0}$ in space. Specifically,

$$\begin{aligned}
\int_\Omega v_j \frac{\partial u_j}{\partial x_i}\frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Omega &= \int_{\partial\Omega} v_j u_j \frac{\delta u_i}{\delta \beta} n_i \,\mathrm{d}\Gamma - \int_\Omega u_j \frac{\partial}{\partial x_i}\left(v_j \frac{\delta u_i}{\delta \beta}\right)\mathrm{d}\Omega \\
&= \int_{\partial\Omega} v_j u_j \frac{\delta u_i}{\delta \beta} n_i \,\mathrm{d}\Gamma - \int_\Omega u_j \frac{\partial v_j}{\partial x_i}\frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Omega - \int_\Omega u_j v_j \frac{\partial}{\partial x_i}\left(\frac{\delta u_i}{\delta \beta}\right)\mathrm{d}\Omega \\
&= \int_{\partial\Omega} v_j u_j \frac{\delta u_i}{\delta \beta} n_i \,\mathrm{d}\Gamma - \int_\Omega u_j \frac{\partial v_j}{\partial x_i}\frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Omega - \int_\Omega u_j v_j \frac{\delta}{\delta \beta}\left(\frac{\partial u_i}{\partial x_i}\right)\mathrm{d}\Omega \\
&= \int_{\partial\Omega} v_j u_j n_i \frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Gamma - \int_\Omega u_j \frac{\partial v_j}{\partial x_i}\frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Omega,
\end{aligned} \tag{5.23}$$

where the integration in time is left out for convenience. This results in the same expressions shown in Equation (5.22), except for R_i^u and D_i^u,

$$R_i^v = -\frac{\partial v_i}{\partial t} - u_j \frac{\partial v_j}{\partial x_i} - \frac{\partial (v_i u_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right) + \frac{\partial q}{\partial x_i} \tag{5.24a}$$

$$D_i^v = -q n_i + u_j(v_j n_i + v_i n_j) + \nu n_j \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{5.24b}$$

The notation (R_i^v, D_i^v) will be used to generalize the derivation from this point. We use the conservative expressions for the numerical experiments in the following sections.

Recalling Figure 5.2, the sensitivity of the Lagrange function is ultimately expressed as

$$\begin{aligned}
\frac{\delta L}{\delta \beta} &= \mathrm{I} + \mathrm{II} \\
&= \frac{1}{T}\int_\Omega v_i \frac{\delta u_i}{\delta \beta}\Big|_{t_{\text{start}}}^{t_{\text{end}}}\mathrm{d}\Omega + \frac{1}{T}\int_I\int_\Omega \left(\frac{\partial J_\Omega}{\partial \beta} + \hat{R}^q \frac{\delta p}{\delta \beta} + \hat{R}^v \frac{\delta u_i}{\delta \beta}\right)\mathrm{d}\Omega\,\mathrm{d}t \\
&\quad + \frac{1}{T}\int_I\int_{\partial\Omega}\left(\frac{\partial J_\Gamma}{\partial \beta} + \hat{D}^q \frac{\delta p}{\delta \beta} + \hat{D}_i^v \frac{\delta u_i}{\delta \beta} + \hat{P}_{ij}^v \frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right)\right)\mathrm{d}\Gamma\,\mathrm{d}t,
\end{aligned} \tag{5.25}$$

where the following expressions are employed

$$\begin{aligned}
\hat{R}^q &= \frac{\partial J_\Omega}{\partial p} + R^q \\
\hat{R}^v &= \frac{\partial J_\Omega}{\partial u_i} + R_i^v \\
\hat{D}^q &= \frac{\partial J_\Gamma}{\partial p} + D^q \\
\hat{D}_i^v &= \frac{\partial J_\Gamma}{\partial u_i} + D_i^v \\
\hat{P}_{ij}^v &= \frac{\partial J_\Gamma}{\partial u_{i,j}} + P_{ij}^v \,.
\end{aligned} \tag{5.26}$$

These expressions consider the contribution of the QoI while Equations (5.22a) to (5.22e) only consider the constraints.

Hence, the adjoint system, including adjoint equations and boundary conditions, is formulated by satisfying the Equation (5.6a). More specifically, the adjoint equations are obtained from the domain integral by setting terms with the sensitivity of the primal variables to zero, i.e.

$$\begin{aligned}
\hat{R}^q &= \frac{\partial J_\Omega}{\partial p} + R^q = 0 \\
\hat{R}^v &= \frac{\partial J_\Omega}{\partial u_i} + R_i^v = 0 \,.
\end{aligned} \tag{5.27}$$

The associated boundary conditions for the adjoint problem are determined by the boundary integral that should be zero, as follows

$$\begin{aligned}
\hat{D}^q &= \frac{\partial J_\Gamma}{\partial p} + D^q = 0 \\
\hat{D}_i^v &= \frac{\partial J_\Gamma}{\partial u_i} + D_i^v = 0 \\
\hat{P}_{ij}^v &= \frac{\partial J_\Gamma}{\partial u_{i,j}} + P_{ij}^v = 0 \,.
\end{aligned} \tag{5.28}$$

The QoI will change the final expression of adjoint equations and boundary conditions. Therefore, the specific expressions of Equations (5.27) and (5.28) need to be determined with respect to a particular problem.

ADJOINT EQUATIONS FOR THE DRAG

When the QoI is chosen as the drag for cylinder flows, it is expressed as

$$J = \frac{1}{T} \int_I \oint_S (-p n_j + \tau_{ij} n_i) r_j \, \mathrm{d}S \, \mathrm{d}t \,, \tag{5.29}$$

where $\tau_{ij} = \nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ and r_j denotes the drag direction. In other words, $J_\Gamma = (-pn_j + \tau_{ij}n_i)r_j$ and $J_\Omega = 0$. The adjoint equations are then

$$\begin{aligned} R^q &= -\frac{\partial v_i}{\partial t} - u_j\frac{\partial v_j}{\partial x_i} - \frac{\partial (u_j v_i)}{\partial x_j} - \frac{\partial}{\partial x_j}\big(\nu(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i})\big) + \frac{\partial q}{\partial x_i} = 0 \\ R_i^u &= -\frac{\partial v_i}{\partial x_i} = 0\,. \end{aligned} \tag{5.30}$$

The pressure term is positive in the flow problem, while the adjoint pressure term is also positive. This is because the adjoint pressure is derived based on the continuity equation that is given with a minus sign. Meanwhile, the influence of the pressure term is reflected in the adjoint continuity equation.

The initial condition for the unsteady adjoint is determined by

$$\frac{1}{T}\int_\Omega v_i \frac{\delta u_i}{\delta \beta}\Big|_{t_{\text{start}}}^{t_{\text{end}}} \mathrm{d}\Omega = 0\,. \tag{5.31}$$

Since the flow velocity is known at $t = t_{\text{start}}$, it cannot be altered and thus its variation $(\frac{\delta u_i}{\delta \beta})$ is zero. Given that there is no contribution to the drag at the ending time, t_{end}, and the flow solutions are not known yet, the velocity variation $(\frac{\delta u_i}{\delta \beta})$ can be arbitrary values when the design variables are changed. Therefore, the adjoint velocity v_i has to be zero in order to satisfy the condition Equation (5.31), i.e. Equation (5.6a). In that sense, we have the initial condition at $t = t_{\text{end}}$ for the adjoint problem,

$$v_i = 0\,, \tag{5.32}$$

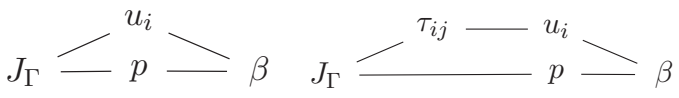
and thus the adjoint problem should be solved backward in time.

5.2.3. ADJOINT BOUNDARY CONDITIONS

The boundary conditions for the adjoint problem are derived as follows. When the drag is considered as the QoI, the term I is re-expressed as

$$\mathrm{I} = \frac{1}{T}\int_I\int_{\partial\Gamma} \frac{\delta J_\Gamma}{\delta \beta}\,\mathrm{d}\Gamma\,\mathrm{d}t = \frac{1}{T}\int_I\int_{\partial\Gamma} \Big(\frac{\partial J_\Gamma}{\partial u_i}\frac{\delta u_i}{\delta \beta} + \frac{\partial J_\Gamma}{\partial p}\frac{\delta p}{\delta \beta}\Big)\,\mathrm{d}\Gamma\,\mathrm{d}t\,, \tag{5.33}$$

since the drag relies on the u_i, p as shown in Figure 5.3(a). Given that the drag is defined



(a) General expression (b) With stress tensor

Figure 5.3: The sketch of a drag function and the flow and adjoint variables.

by the stress tensor as shown in Figure 5.3(b), the term I can be also stated as

$$\begin{aligned}
\mathrm{I} &= \frac{1}{T}\int_I\int_{\partial\Gamma}\left(\frac{\partial J_\Gamma}{\partial \tau_{kj}}\frac{\delta\tau_{ij}}{\partial\beta}+\frac{\partial J_\Gamma}{\partial p}\frac{\delta p}{\delta\beta}\right)\mathrm{d}\Gamma\,\mathrm{d}t \\
&= \frac{1}{T}\int_I\oint_S\left(n_i r_j\frac{\delta\tau_{ij}}{\delta\beta}-n_j r_j\frac{\delta p}{\delta\beta}\right)\mathrm{d}S\,\mathrm{d}t && \triangleright\text{Equation (5.29)} \\
&= \frac{1}{T}\int_I\oint_S\left(n_i r_j \nu\frac{\delta}{\delta\beta}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-n_j r_j\frac{\delta p}{\delta\beta}\right)\mathrm{d}S\,\mathrm{d}t && \triangleright\ \nu\text{ is a constant} \\
&= \frac{1}{T}\int_I\oint_S\left(n_i r_j \nu\left[\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)+\frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta\beta}\right)\right]-n_j r_j\frac{\delta p}{\delta\beta}\right)\mathrm{d}S\,\mathrm{d}t && \triangleright\text{ Order exchange} \\
&= \frac{1}{T}\int_I\oint_S\left(\nu\left(n_i r_j+n_j r_i\right)\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)-n_i r_i\frac{\delta p}{\delta\beta}\right)\mathrm{d}S\,\mathrm{d}t,
\end{aligned} \tag{5.34}$$

where S is the cylinder wall surface, part of the whole boundary $\partial\Omega$. Therefore, $\frac{\partial J_\Gamma}{\partial p}$, $\frac{\partial J_\Gamma}{\partial u_i}$ and $\frac{\partial J_\Gamma}{\partial u_{i,j}}$ will vanish in all boundaries except the cylinder wall surface, in which they are read as

$$\begin{aligned}
\frac{\partial J_\Gamma}{\partial p} &= -n_i r_i \\
\frac{\partial J_\Gamma}{\partial u_i} &= 0 \\
\frac{\partial J_\Gamma}{\partial u_{i,j}} &= \nu(n_i r_j + n_j r_i)\,.
\end{aligned} \tag{5.35}$$

Each boundary of the computational domain Ω in Figure 5.1 is considered in detail below.

INLET

At the inlet, Dirichlet boundary conditions are applied to the primal velocity so that

$$\frac{\delta u_i}{\delta\beta} = 0\,. \tag{5.36}$$

The boundary-integral component in Equation (5.25) is then stated as

$$\frac{\delta L}{\delta\beta}(\partial\Omega) = \frac{1}{T}\int_I\int_{\partial\Omega}\left(D^q\frac{\delta p}{\delta\beta}+P^v_{ij}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\right)\mathrm{d}\Gamma\,\mathrm{d}t\,. \tag{5.37}$$

The Lagrange condition requires this component to be zero, i.e.

$$\begin{aligned}
D^q &= 0 \\
P^v_{ij} &= 0\,.
\end{aligned} \tag{5.38}$$

Thus we have

$$D^q = v_i n_i = v_n = 0\,, \tag{5.39}$$

which indicates the normal component of adjoint velocity is zero. There are two ways to derive the boundary condition from P^v_{ij}. Firstly we can project P^v_{ij} onto the normal direction n_i

$$\begin{aligned} n_i P^v_{ij} &= -\nu(n_i v_i n_j + n_i v_j n_i) = 0 \\ \Longrightarrow & n_i v_i n_j + n_i v_j n_i = v_n n_j + n_i n_i v_j = v_j = 0\,. \end{aligned} \tag{5.40}$$

Therefore, the adjoint velocity u_j satisfies the zero Dirichlet boundary condition at the inlet. Alternatively, we can expand the p^v_{ij} based on the summation indices i, j as

$$\begin{aligned} &P^v_{11} = 0 \rightarrow v_1 n_1 + v_1 n_1 = 0 \Longrightarrow v_1 = 0 \\ &P^v_{12} = 0 \rightarrow v_1 n_2 + v_2 n_1 = 0 \Longrightarrow v_2 = 0 \quad \triangleright \text{ Due to } v_1 = 0 \\ &P^v_{13} = 0 \rightarrow v_1 n_3 + v_3 n_1 = 0 \Longrightarrow v_3 = 0\,. \quad \triangleright \text{ Due to } v_1 = 0 \end{aligned} \tag{5.41}$$

Consequently, the components of adjoint velocity are zero. The adjoint pressure is not determined from those conditions. Therefore, the zero Neumann boundary condition is applied [141].

CYLINDER SURFACE

At the cylinder wall surface, zero Dirichlet and zero Neumann boundary conditions are applied to the primal velocity and pressure, respectively. Therefore, we have

$$\frac{\delta u_i}{\delta \beta} = 0,\ n_j \frac{\partial}{\partial x_j}\frac{\delta p}{\delta \beta} = 0\,. \tag{5.42}$$

In order to satisfy the Lagrange condition, the surface integrals should be set to zero, viz.

$$\begin{aligned} \hat{D}^q &= \frac{\partial J_\Gamma}{\partial p} + D^q = -n_i r_i + v_i n_i = 0 \\ \hat{P}^v_{ij} &= \frac{\partial J_\Gamma}{\partial u_{i,j}} + P^v_{ij} = \nu(n_i r_j + n_j r_i) - \nu(v_i n_j + v_j n_i) = 0\,. \end{aligned} \tag{5.43}$$

The condition of $\hat{D}^q$ implies that the normal component of the adjoint velocity is identified as

$$v_n = r_n\,. \tag{5.44}$$

The other condition of P^v_{ij} can be simplified using the projection as follows

$$\begin{aligned} n_i \hat{P}^v_{ij} &= n_i \nu(n_i r_j + n_j r_i) - n_i \nu(v_i n_j + v_j n_i) \\ &= 2\nu\left[r_j - v_j + n_j(r_n - v_n)\right] \qquad \triangleright \text{ Equation (5.44)} \\ &= 2\nu(r_j - v_j) = 0 \\ \Rightarrow v_j &= r_j\,. \end{aligned} \tag{5.45}$$

This can be derived using the expansion method as well. Consequently, the adjoint velocity is a constant vector, i.e. r_i, which is consistent with in literature [39]. A zero Neumann boundary condition is chosen for the adjoint pressure.

OUTLET

At the outlet, the primal boundary conditions are given by $\frac{\partial u_i}{\partial n} = 0$, p = constant, and thus we have $\frac{\delta p}{\delta \beta} = 0, n_j \frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right) = 0$. The stress τ_{ij} at the outlet is physically independent of the design variables [140] when the outlet is far away from the object. Thus we have $\frac{\delta \tau_{ij}}{\delta \beta} = 0$,

$$\frac{\delta \tau_{ij}}{\delta \beta} = \nu \frac{\delta}{\delta \beta}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \nu\left(\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta \beta}\right)\right) = 0\,. \tag{5.46}$$

Projecting Equation (5.46) onto v_i and n_j, we have

$$v_i n_j \frac{\delta \tau_{ij}}{\delta \beta} = 0 \tag{5.47}$$

Therefore, we have

$$\begin{aligned} P^v_{ij}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right) &= -\nu\left(v_i n_j + v_j n_i\right)\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right) \\ &= -v_i n_j \nu\left(\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta \beta}\right) + \frac{\partial}{\partial x_i}\left(\frac{\delta u_j}{\delta \beta}\right)\right) = -v_i n_j \frac{\delta \tau_{ij}}{\delta \beta} = 0\,. \end{aligned} \tag{5.48}$$

Thus the last part of the boundary-integral component in Equation (5.25) vanishes and we have the Lagrange condition

$$\frac{\delta L}{\delta \beta}(\partial\Omega) = \frac{1}{T}\int_I \int_{\partial\Omega} D^v_i \frac{\delta u_i}{\delta \beta}\,\mathrm{d}\Gamma\,\mathrm{d}t\,. \tag{5.49}$$

D^v_i should be set to zero, i.e. $D^v_i = 0$, to satisfy the Lagrange condition. Consequently, we have

$$D^v_i = -q n_i + u_j(v_j n_i + v_i n_j) + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) = 0\,. \tag{5.50}$$

The adjoint boundary conditions are derived by projecting the Equation (5.50) onto normal (n_i) and tangential directions. When using the conservative variables, the boundary condition for adjoint pressure is derived as

$$\begin{aligned} n_i D^v_i &= -q + u_j(v_j + v_n n_j) + \nu n_i n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \\ &= -q + v_j u_j + v_n u_n + 2\nu n_j \frac{\partial v_n}{\partial x_j} = 0 \\ \Rightarrow \quad q &= v_j u_j + v_n u_n + 2\nu n_j \frac{\partial v_n}{\partial x_j}\,. \end{aligned} \tag{5.51}$$

The boundary condition for the adjoint velocity is determined by

$$\begin{aligned} D_i^v - n_k D_k^v n_i &= -qn_i + u_j(v_j n_i + v_i n_j) + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \\ &\quad - \left[-qn_i + v_j u_j n_i + v_n u_n n_i + 2\nu n_i n_j \frac{\partial v_n}{\partial x_j}\right] \\ &= u_n(v_i - v_n n_i) + \nu n_j\left(\frac{\partial (v_i - v_k n_k n_i)}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - n_i n_k \frac{\partial v_j}{\partial x_k}\right) = 0 \\ \Rightarrow \quad & u_n v_{\parallel i} + \nu n_j\left(\frac{\partial v_{\parallel i}}{\partial x_j} + \frac{\partial v_j}{\partial x_{\parallel i}}\right) = 0\,. \end{aligned} \tag{5.52}$$

where $v_{\parallel i}$ denotes the tangent velocity and $x_{\parallel i}$ stands for the tangent direction.

Discretization for cylinder flows The adjoint pressure can be computed by

$$q = v_j u_j + v_n u_n + 2\nu n_j \frac{\partial v_n}{\partial x_j}\,, \tag{5.53}$$

where the primal and adjoint velocities can be interpolated at the boundary and the gradient term is discretized by

$$\nu n_j \frac{\partial v_n}{\partial x_j} = \nu \frac{\partial v_n}{\partial n} \approx \nu \frac{v_{n,\mathrm{P}} - v_{n,\mathrm{N}}}{\Delta h}\,, \tag{5.54}$$

as long as the adjoint velocity is known. Equation (5.52) states that the velocity is an implicit function. Equation (5.54) is thus solved with the discrete system for the domain. However, this will change the matrix structure and make implementation more complicated in OpenFOAM. Rather than using a coupled approach, the boundary condition for the adjoint velocity is evaluated using the solution at the last time step. The adjoint velocity is decomposed into normal and tangential components for evaluation. The normal component is constructed from the face flux (F_f), i.e. $\boldsymbol{u}_n = F_f/\|\boldsymbol{S}\|$. The tangential component ($v_{\parallel i}$) is governed by

$$u_n v_{\parallel i} + \nu n_j\left(\frac{\partial v_{\parallel i}}{\partial x_j} + \frac{\partial v_j}{\partial x_{\parallel i}}\right) = 0\,. \tag{5.55}$$

The viscous terms in this equation are approximated as follows

$$\begin{aligned} \nu n_j \frac{\partial v_{\parallel i}}{\partial x_j} &= \nu \frac{\partial v_{\parallel i}}{\partial n} \approx \nu \frac{v_{\parallel i,\mathrm{P}} - v_{\parallel i,\mathrm{N}}}{\Delta h} \\ \nu n_j \frac{\partial v_j}{\partial x_{\parallel i}} &= \nu \frac{\partial v_n}{\partial x_{\parallel i}} = \nu n_j\left(\frac{\partial v_j}{\partial x_i} - \frac{\partial v_j}{\partial x_k} n_k n_i\right) = \nu n_j \frac{\partial v_j}{\partial x_i} - \nu \frac{\partial v_n}{\partial n} n_i\,. \end{aligned} \tag{5.56}$$

Since $\nu n_j \frac{\partial v_j}{\partial x_{\parallel i}}$ is independent of the tangential component of the adjoint velocity, it is treated as a known value when we solve this equation for the tangential component.

$\nu n_j \frac{\partial v_j}{\partial x_i}$ is evaluated by the velocity gradient and surface normal vector while $\nu \frac{\partial v_n}{\partial n}$ is computed by Equation (5.54). The discretization of Equation (5.55) is expressed as

$$u_n v_{\parallel i,\mathrm{P}} + \nu \frac{v_{\parallel i,\mathrm{P}} - v_{\parallel i,\mathrm{N}}}{\Delta h} + \nu n_j \frac{\partial v_j}{\partial x_i} - \nu \frac{\partial v_n}{\partial n} n_i = 0 . \tag{5.57}$$

Here we have two ways to solve this equation. One way is

$$v_{\parallel i,\mathrm{P}} = -\frac{1}{u_n} \left[\frac{\nu}{\Delta h} \left(v^o_{\parallel i,\mathrm{P}} - v_{\parallel i,\mathrm{N}} \right) + \nu n_j \frac{\partial v_j}{\partial x_i} - \nu \frac{\partial v_n}{\partial n} n_i \right] , \tag{5.58}$$

where v^o denotes the solution at the previous time step. The other way is

$$v_{\parallel i,\mathrm{P}} = -\frac{1}{u_n + \frac{\nu}{\Delta h}} \left[-\frac{\nu}{\Delta h} v_{\parallel i,\mathrm{N}} + \nu n_j \frac{\partial v_j}{\partial x_i} - \nu \frac{\partial v_n}{\partial n} n_i \right] . \tag{5.59}$$

Although the first way has been applied in the OpenFOAM tutorial, the second one provides a more accurate computation, which is used in the current study.

FAR FIELD

For the far-field boundary, Dirichlet boundary conditions are used for the primal velocity and pressure, viz. p = constant, u_i = constant. Therefore we have $\frac{\delta u_i}{\delta \beta} = 0, \frac{\delta p}{\delta \beta} = 0$. Thus the boundary integral in Equation (5.25) becomes

$$\frac{\delta L}{\delta \beta}(\partial\Omega) = \frac{1}{T} \int_T \int_{\partial\Omega} P^v_{ij} \frac{\partial}{\partial x_j} \left(\frac{\delta u_i}{\delta \beta} \right) \mathrm{d}\Gamma \, \mathrm{d}t . \tag{5.60}$$

The Lagrange condition requires $P^v_{ij} = 0$ and thus the adjoint velocity is zero, i.e. $v_i = 0$, because of Equation (5.41). As there is no constraint for the adjoint pressure, a zero Neumann boundary condition is used for the adjoint pressure.

PERIODICITY

For arbitrary periodic boundaries, their wall normal unit vectors are reversed to each other, viz. $\boldsymbol{n}^{\Gamma_1} = -\boldsymbol{n}^{\Gamma_2}$, as shown in Figure 5.4. The primal boundary conditions are given as $u_i^{\Gamma_1} = u_i^{\Gamma_2}, p^{\Gamma_1} = p^{\Gamma_2}$. The boundary conditions for the adjoint velocity and pressure are also periodic, i.e. $v_i^{\Gamma_1} = v_i^{\Gamma_2}, q^{\Gamma_1} = q^{\Gamma_2}$. This can be proved as follows.

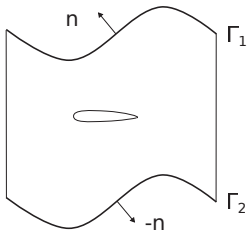


Figure 5.4: The sketch of the periodic boundary condition.

The flow states are identical at periodic boundaries. In other words, the geometry of periodic boundaries is identical and the gradient of scalar variables is also the same. In that sense, we have

$$\begin{aligned} & u_i^{\Gamma_1} = u_i^{\Gamma_2}, p^{\Gamma_1} = p^{\Gamma_2}, \frac{\partial u_i^{\Gamma_1}}{\partial x_j} = \frac{\partial u_i^{\Gamma_2}}{\partial x_j} \\ \Rightarrow \quad & \frac{\delta u_i^{\Gamma_1}}{\delta \beta} = \frac{\delta u_i^{\Gamma_2}}{\delta \beta}, \quad \frac{\delta p^{\Gamma_1}}{\delta \beta} = \frac{\delta p^{\Gamma_2}}{\delta \beta}, \quad \frac{\partial}{\partial x_j}\left(\frac{\delta u_i^{\Gamma_1}}{\delta \beta}\right) = \frac{\partial}{\partial x_j}\left(\frac{\delta u_i^{\Gamma_2}}{\delta \beta}\right). \end{aligned} \tag{5.61}$$

The boundary integral in Equation (5.25) becomes

$$\begin{aligned} & \frac{\delta L}{\delta \beta}(\Gamma_1 + \Gamma_2) \\ & = \frac{1}{T}\int_I \int_{\Gamma_1} \left(D^{q,\Gamma_1}\frac{\delta p^{\Gamma_1}}{\delta \beta} + D_i^{v,\Gamma_1}\frac{\delta u_i^{\Gamma_1}}{\delta \beta} + P_{ij}^{v,\Gamma_1}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i^{\Gamma_1}}{\delta \beta}\right)\right) \mathrm{d}\Gamma\, \mathrm{d}t \\ & + \frac{1}{T}\int_I \int_{\Gamma_2} \left(D^{q,\Gamma_2}\frac{\delta p^{\Gamma_2}}{\delta \beta} + D_i^{v,\Gamma_2}\frac{\delta u_i^{\Gamma_2}}{\delta \beta} + P_{ij}^{v,\Gamma_2}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i^{\Gamma_2}}{\delta \beta}\right)\right) \mathrm{d}\Gamma\, \mathrm{d}t \\ & = \frac{1}{T}\int_I \int_{\Gamma_1} \left(\left(D^{q,\Gamma_1} + D^{q,\Gamma_2}\right)\frac{\delta p^{\Gamma_1}}{\delta \beta} + \left(D_i^{v,\Gamma_1} + D_i^{v,\Gamma_2}\right)\frac{\delta u_i^{\Gamma_1}}{\delta \beta} + \left(P_{ij}^{v,\Gamma_1} + P_{ij}^{v,\Gamma_2}\right)\frac{\partial}{\partial x_j}\left(\frac{\delta u_i^{\Gamma_1}}{\delta \beta}\right)\right) \mathrm{d}\Gamma\, \mathrm{d}t. \end{aligned} \tag{5.62}$$

This is because Γ_1 has an integral path that is the same as for Γ_2. The Lagrange condition will lead to

$$D^{q,\Gamma_1} + D^{q,\Gamma_2} = 0 \tag{5.63a}$$

$$D_i^{v,\Gamma_1} + D_i^{v,\Gamma_2} = 0 \tag{5.63b}$$

$$P_{ij}^{v,\Gamma_1} + P_{ij}^{v,\Gamma_2} = 0. \tag{5.63c}$$

Equation (5.63a) is stated as $v_i^{\Gamma_1} n_i^{\Gamma_1} + v_i^{\Gamma_2} n_i^{\Gamma_2} = 0$ or $v_n^{\Gamma_1} = -v_n^{\Gamma_2}$. Furthermore, we can project Equation (5.63c) onto $n_i^{\Gamma_1}$ as

$$\begin{aligned} n_i^{\Gamma_1}\left(P_{ij}^{v,\Gamma_1} + P_{ij}^{v,\Gamma_2}\right) &= -\nu n_i^{\Gamma_1}\left(v_i^{\Gamma_1} n_j^{\Gamma_1} + v_j^{\Gamma_1} n_i^{\Gamma_1}\right) - \nu n_i^{\Gamma_1}(v_i^{\Gamma_2} n_j^{\Gamma_2} + v_j^{\Gamma_2} n_i^{\Gamma_2}) \\ &= -\nu\left(v_n^{\Gamma_1} n_j^{\Gamma_1} - v_n^{\Gamma_2} n_j^{\Gamma_2}\right) - \nu\left(v_j^{\Gamma_1} - v_j^{\Gamma_2}\right) \\ &= -\nu\left(v_n^{\Gamma_1} + v_n^{\Gamma_2}\right) n_j^{\Gamma_1} - \nu\left(v_j^{\Gamma_1} - v_j^{\Gamma_2}\right) \\ &= -\nu\left(v_j^{\Gamma_1} - v_j^{\Gamma_2}\right) = 0 \quad \Rightarrow \quad v_j^{\Gamma_1} = v_j^{\Gamma_2}. \end{aligned} \tag{5.64}$$

This gives proof of the periodicity for adjoint velocity, viz. $v_i^{\Gamma_1} = v_i^{\Gamma_2}$.

The boundary of adjoint pressure is derived by projecting Equation (5.63b) onto $n_i^{\Gamma_1}$

as

$$
\begin{aligned}
n_i^{\Gamma_1}\left(D_i^{\nu,\Gamma_1}+D_i^{\nu,\Gamma_2}\right) &= -q^{\Gamma_1}+u_j^{\Gamma_1}v_j^{\Gamma_1}+u_j^{\Gamma_1}n_j^{\Gamma_1}v_i^{\Gamma_1}n_i^{\Gamma_1}+\nu n_i^{\Gamma_1}n_j^{\Gamma_1}\left(\frac{\partial v_i^{\Gamma_1}}{\partial x_j}+\frac{\partial v_j^{\Gamma_1}}{\partial x_i}\right)\\
&\quad +q^{\Gamma_2}-u_j^{\Gamma_2}v_j^{\Gamma_2}-u_j^{\Gamma_2}n_j^{\Gamma_1}v_i^{\Gamma_2}n_i^{\Gamma_1}-\nu n_i^{\Gamma_1}n_j^{\Gamma_1}\left(\frac{\partial v_i^{\Gamma_2}}{\partial x_j}+\frac{\partial v_j^{\Gamma_2}}{\partial x_i}\right)\\
&= q^{\Gamma_2}-q^{\Gamma_1}=0 \quad \Rightarrow \quad q^{\Gamma_1}=q^{\Gamma_2}.
\end{aligned}
\tag{5.65}
$$

FREE-SLIP WALL

On a free-slip boundary, the normal component of the primal velocity is zero, $u_n = u_j n_j = 0$, and the primal pressure satisfies a zero Neumann boundary condition in the normal direction. We thus have $\frac{\delta(u_j n_j)}{\delta\beta}=0, \frac{\partial}{\partial n}\left(\frac{\delta p}{\delta\beta}\right)=0$. The normal component of primal velocity variation vanishes, but the tangential part can change freely, leading to the requirement for zeroing $\frac{\delta u_i}{\delta\beta}$. Given that there is no contribution from the QoI, the Lagrange conditions give

$$
D^q=0, \quad D_i^v=0, \quad P_{ij}^v=0. \tag{5.66}
$$

Based on the derivation for the inlet boundary, $D^q=0$ and $P_{ij}^v=0$ are equivalent to $v_i=0$. The boundary condition for adjoint pressure is given by projecting $D_i^v=0$ onto n_i. When using the conservative variables, this leads to

$$
\begin{aligned}
n_i D_i^v &= -q+u_j(v_j+v_n n_j)+\nu n_i n_j\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)\\
&= -q+2\nu n_j\frac{\partial v_n}{\partial x_j}=0 \quad \Rightarrow \quad q=2\nu n_j\frac{\partial v_n}{\partial x_j}.
\end{aligned}
\tag{5.67}
$$

SYMMETRY PLANE

Symmetry boundary conditions will be used for primal flow variables, i.e. the primal flow variables are symmetric with respect to a plane. Thus the normal gradient of the velocity at this plane is zero, i.e. zero Neumann boundary condition, $u_n=0, \frac{\partial p}{\partial n}=0, \frac{\partial u_\parallel}{\partial n}=0$. Therefore, we have

$$
\frac{\delta u_n}{\delta\beta}=0, \frac{\partial}{\partial n}\left(\frac{\delta p}{\delta\beta}\right)=0, \frac{\partial}{\partial n}\left(\frac{\delta u_\parallel}{\delta\beta}\right)=0. \tag{5.68}
$$

The symmetry boundary condition should be also applied for the adjoint pressure and velocity, i.e. $v_n=0, \frac{\partial q}{\partial n}=0, \frac{\partial v_\parallel}{\partial n}=0$, which is done as follows.

At the symmetry boundary, $\frac{\partial u_\parallel}{\partial n}=0$ is the constraint in the tangential direction. We use $\boldsymbol{\iota}$ (i.e. ι_i) to describe the unit vector in this direction and $\boldsymbol{n}\cdot\boldsymbol{\iota}=0$. Actually, this condition denotes two boundary conditions inside the symmetry plane, and their directions are orthogonal to each other, viz. $\boldsymbol{\iota}=\mathbf{t}+\mathbf{s}, \mathbf{t}\cdot\mathbf{s}=0$.

Given that there is no contribution from the QoI, the surface integral in Equation (5.25) is simplified as

$$
\frac{\delta L}{\delta\beta}(\partial\Omega)=\frac{1}{T}\int_I\int_{\partial\Omega}\left(D^q\frac{\delta p}{\delta\beta}+D_i^v\frac{\delta u_i}{\delta\beta}+P_{i,j}^v\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right)\right)\mathrm{d}\Gamma\,\mathrm{d}t. \tag{5.69}
$$

The term involving $P^v_{i,j}$ is expanded as

$$\begin{aligned} P^v_{i,j}\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right) &= -\nu(v_i n_j + v_j n_i)\frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right) \\ &= -\nu v_i n_j \frac{\partial}{\partial x_j}\left(\frac{\delta u_i}{\delta\beta}\right) - \nu v_j \frac{\partial}{\partial x_j}\left(\cancel{\frac{\delta u_n}{\delta\beta}}\right) \\ &= -\nu v_i n_j \frac{\partial}{\partial x_j}\left(\frac{\delta u_n n_i + u_\parallel \iota_i}{\delta\beta}\right) \\ &= -\nu v_i n_j \frac{\partial}{\partial x_j}\left(n_i \cancel{\frac{\delta u_n}{\delta\beta}} + \iota_i \frac{\delta u_\parallel}{\delta\beta}\right) \\ &= -\nu v_i \iota_i \cancel{\frac{\partial}{\partial n}\left(\frac{\delta u_\parallel}{\delta\beta}\right)} = 0\,. \end{aligned} \tag{5.70}$$

As a consequence, the integral term involving $P^v_{i,j}$ is eliminated automatically. The Lagrange conditions require

$$D^q = 0, \quad D^v_i = 0\,. \tag{5.71}$$

For $D^q = 0$, this reads as

$$D^q = v_i n_i = v_n = 0\,. \tag{5.72}$$

For $D^v_i = 0$, we have

$$D^v_i = -q n_i + u_j v_i n_j + u_j v_j n_i + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{5.73}$$

It is noted that the term involved $-qn_i$ is expressed as

$$\frac{1}{T}\int_I\int_{\partial\Omega} -q n_i \frac{\delta u_i}{\delta\beta}\,\mathrm{d}\Gamma\,\mathrm{d}t = \frac{1}{T}\int_I\int_{\partial\Omega} -q\cancel{\frac{\delta u_n}{\delta\beta}}\,\mathrm{d}\Gamma\,\mathrm{d}t = 0\,. \tag{5.74}$$

This means there is no specific requirement for the adjoint pressure q. The zero gradient condition is applied to q here as is commonly recommended. The remaining terms are

$$\begin{aligned} D^v_i &= u_j v_i n_j + u_j v_j n_i + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) = \cancel{u_n} v_i + u_j v_j n_i + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \\ &= u_j v_j n_i + \nu n_j\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) = u_j v_j n_i + \nu\left(\frac{\partial v_i}{\partial n} + \frac{\partial v_n}{\partial x_i}\right) = 0\,. \end{aligned} \tag{5.75}$$

Projecting it onto $\boldsymbol{\iota}$ results in

$$u_j v_j \cancel{n_i \iota_i} + \iota_i\left(\frac{\partial v_i}{\partial n} + \frac{\partial v_n}{\partial x_i}\right) = \frac{\partial(v_i\iota_i)}{\partial n} + \frac{\partial v_n}{\partial x_i}\iota_i = \frac{\partial v_\parallel}{\partial n} + \frac{\partial v_n}{\partial x_i}\iota_i = 0\,. \tag{5.76}$$

The gradient of v_n can be split into directional derivatives as

$$\begin{aligned} &\frac{\partial v_n}{\partial x_i} = \frac{\partial v_n}{\partial x_j} n_j n_i + \frac{\partial v_n}{\partial x_j}\iota_j\iota_i \\ \Rightarrow\quad &\frac{\partial v_n}{\partial x_i}\iota_i = \frac{\partial v_n}{\partial x_j} n_j \cancel{n_i \iota_i} + \frac{\partial v_n}{\partial x_j}\iota_j\iota_i\iota_i = \frac{\partial v_n}{\partial x_j}\iota_j = \frac{\partial v_n}{\partial \iota}\,. \end{aligned} \tag{5.77}$$

Since v_n is always equal to zero as shown in Equation (5.72), the directional derivative of v_n in this plane is zero, leading to $\frac{\partial v_n}{\partial t} = 0$. Therefore, we have

$$\begin{aligned} \frac{\partial v_n}{\partial x_i} t_i &= \frac{\partial v_n}{\partial t} = 0 \\ \Rightarrow \quad \frac{\partial v_\parallel}{\partial n} + \frac{\partial v_n}{\partial x_i} t_i &= \frac{\partial v_\parallel}{\partial n} = 0, \end{aligned} \tag{5.78}$$

which gives the symmetry boundary condition for the adjoint velocity.

5.2.4. NUMERICAL DISCRETIZATION

The primal and adjoint problems are discretized using the finite-volume method (FVM). The semi-discrete system is solved with the PIMPLE algorithm. We demonstrate this method for both primal and adjoint problems in this section.

FINITE VOLUME METHOD

Gauss's theorem is applied in FVM to convert the volume integral to surface integrals as

$$\int_\Omega \nabla \star \phi \, \mathrm{d}\Omega = \int_{\partial\Omega} \phi \star \, \mathrm{d}\boldsymbol{S}, \tag{5.79}$$

where ϕ denotes any fields (e.g. pressure, velocity), and $\star$ represents any operators, e.g. inner ($\nabla \cdot \phi$), outer ($\nabla\phi$), and cross ($\nabla \times \phi$). Thus we have

$$\begin{aligned} \int_\Omega \nabla \cdot \phi \, \mathrm{d}\Omega &= \int_{\partial\Omega} \phi \cdot \boldsymbol{n} \, \mathrm{d}S \\ \int_\Omega \nabla \phi \, \mathrm{d}\Omega &= \int_{\partial\Omega} \phi \boldsymbol{n} \, \mathrm{d}S, \end{aligned} \tag{5.80}$$

The values at cell faces need to be evaluated using interpolation schemes. Figure 5.5 shows the face ($\boldsymbol{S}_f$) where we need to evaluate the variables. Essentially, all values will be computed based on solutions from the present (P) and neighbor (N) cells. The neighbor cells can be generally extended to outer layers for high-order schemes. Here linear interpolation is used for the numerical discretization. Geometric information is determined by the computational mesh, such as the distance vector ($\boldsymbol{d}$), surface normal vector ($\boldsymbol{n}$), surface vector ($\boldsymbol{S}_f$), the length ratio of face center to present cell center ($\lambda = \frac{\|\boldsymbol{d}_{\mathrm{P,Cf}}\|}{\|\boldsymbol{d}\|}$) and neighbor cell center ($1-\lambda$). Then the face value (ϕ_{Cf}) can be computed by interpolating the values from cell centers ($\phi_{\mathrm{P}}, \phi_{\mathrm{N}}$) as

$$\phi_{\mathrm{Cf}} = (1-\lambda)\phi_{\mathrm{P}} + \lambda\phi_{\mathrm{N}} \overset{\text{or}}{=} \lambda_{\mathrm{P}}\phi_{\mathrm{P}} + \lambda_{\mathrm{N}}\phi_{\mathrm{N}}, \tag{5.81}$$

where $\lambda_{\mathrm{P}} + \lambda_{\mathrm{N}} = 1$. This technique can be applied for both primal and adjoint fields at cell faces.

PIMPLE ALGORITHM FOR PRIMAL PROBLEM

The unsteady incompressible Navier-Stoke equations (viz. Equation (5.1)) is re-stated in a vector format as

$$\begin{aligned} R &= \nabla \cdot \boldsymbol{u} = 0 \\ R_i &= \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}) + \nabla p = 0, \end{aligned} \tag{5.82}$$

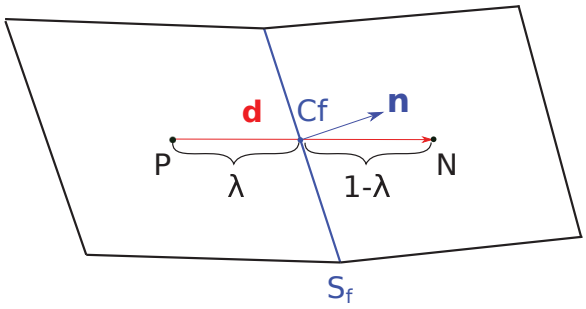


Figure 5.5: Interpolation for the face variables at the face center (Cf) between present (P) and neighbor (N) cells.

where $\boldsymbol{u}$ denotes the velocity vector, i.e. $u_i, i = 1,2,3$ in 3D problems. It is worth mentioning that $\nabla\cdot(\nu\nabla\boldsymbol{u}) = \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right)$ is an expression for the viscous term. However, $\nabla\boldsymbol{u}$ by definition is $\frac{\partial u_j}{\partial x_i}$. Thus, $(\nabla\boldsymbol{u})^{\mathsf{T}} = \frac{\partial u_i}{\partial x_j}$ should be used here. For convenience, we replace $\nabla\cdot\left[\nu(\nabla\boldsymbol{u})^{\mathsf{T}}\right]$ with an expression $\nabla\cdot(\nu\nabla\boldsymbol{u}) = \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right)$. Let us postulate that the finite volume is given at $t \in I : [t^n, t^{n+1}]$, in which the flow solution at $t = t^n$ is known. The nonlinear convection term is linearized about $\boldsymbol{u}^n$ so that the system to be solved is

$$\begin{aligned} R &= \nabla\cdot\boldsymbol{u} = 0 \\ R_i &= \frac{\partial \boldsymbol{u}}{\partial t} + \nabla\cdot(\boldsymbol{u}^n\boldsymbol{u}) - \nabla\cdot(\nu\nabla\boldsymbol{u}) + \nabla p = 0\,. \end{aligned} \tag{5.83}$$

The nonlinearity is then treated by the outer correctors in the PIMPLE algorithm.

The FVM is applied by integrating Equation (5.83) on a spatial volume (ΔV). Using Gauss theorem with a linear interpolation scheme, the volume integration is converted into surface integrals as

$$\begin{aligned} \int_V \nabla\cdot(\boldsymbol{u}^n\boldsymbol{u})\,\mathrm{d}V &= \int_S \boldsymbol{n}\cdot\boldsymbol{u}^n\boldsymbol{u}\,\mathrm{d}S = \sum_{S_f}\int_{S_f}\boldsymbol{n}\cdot\boldsymbol{u}^n\boldsymbol{u}\,\mathrm{d}S \\ &\approx \sum_{S_f}\boldsymbol{S}_f\cdot\boldsymbol{u}_f^n\boldsymbol{u}_f = \sum_{S_f}F_f\boldsymbol{u}_f \\ &= \sum_{S_{f_i}}F_{f_i}\left[(1-\lambda_i)\boldsymbol{u}_{\mathrm{P}} + \lambda_i\boldsymbol{u}_{\mathrm{N}_i}\right] \quad \triangleright \text{Linear interpolation} \\ -\int_V \nabla\cdot(\nu\nabla\boldsymbol{u})\,\mathrm{d}V &= -\sum_{S_f}\int_{S_f}\boldsymbol{n}\cdot\nu\nabla\boldsymbol{u}\mathrm{d}S \approx -\sum_{S_f}\nu\boldsymbol{S}_f\cdot\nabla\boldsymbol{u} \\ &= -\sum_{S_{f_i}}\nu\frac{\boldsymbol{u}_{\mathrm{N}_i}-\boldsymbol{u}_{\mathrm{P}}}{\|\boldsymbol{d}\|}\|\boldsymbol{S}_{f_i}\| = \sum_{S_{f_i}}\nu\frac{\|\boldsymbol{S}_{f_i}\|}{\|\boldsymbol{d}\|}\left(\boldsymbol{u}_{\mathrm{P}}-\boldsymbol{u}_{\mathrm{N}_i}\right) \\ \int_V \nabla p\,\mathrm{d}V &\approx \Delta V\nabla p\,. \end{aligned} \tag{5.84}$$

Here $\boldsymbol{u}_{\mathrm{P}}$ denotes the flow solution at the current cell and $\boldsymbol{u}_{\mathrm{N}_i}$ is the solution in neighbor cells. $\boldsymbol{u}_{\mathrm{N}_i}$ is introduced for the splitting operation used in the PIMPLE algorithm. $\boldsymbol{n}$

stands for the surface normal vector. $F_f = \boldsymbol{S}_f \cdot \boldsymbol{u}_f^n$ represents face flux and $\boldsymbol{S}_f$ is the area vector of face f. λ_i denotes the interpolation weight, which is 0.5 for linear interpolation. $\boldsymbol{d}$ is the distance vector from the current cell center and neighbor cell center. The time gradient of velocity is approximated at the cell center, resulting in a semi-discretization system

$$\frac{\partial \boldsymbol{u}_\mathrm{P}}{\partial t} = -f(\boldsymbol{u}, p), \tag{5.85}$$

where

$$f(\boldsymbol{u}, p) = \frac{1}{\Delta V}\sum_{S_{f_i}} F_{f_i}\left[(1-\lambda_i)\boldsymbol{u}_\mathrm{P} + \lambda_i \boldsymbol{u}_{\mathrm{N}_i}\right] + \frac{1}{\Delta V}\sum_{S_{f_i}} \nu \frac{\|\boldsymbol{S}_{f_i}\|}{\|\boldsymbol{d}\|}(\boldsymbol{u}_\mathrm{P} - \boldsymbol{u}_{\mathrm{N}_i}) + \nabla p. \tag{5.86}$$

The backward Euler time scheme is considered here to demonstrate the PIMPLE algorithm. The full-discretization system is then obtained as

$$\frac{\boldsymbol{u}_\mathrm{P}^{n+1} - \boldsymbol{u}_\mathrm{P}^n}{\Delta t} = -f(\boldsymbol{u}^{n+1}, p^{n+1}), \tag{5.87}$$

where $\Delta t = t^{n+1} - t^n$. This system can be expressed in a matrix format as follows

$$\boldsymbol{A}_\mathrm{P}\boldsymbol{u}_\mathrm{P}^{n+1} + \sum \boldsymbol{A}_\mathrm{N}\boldsymbol{u}_\mathrm{N}^{n+1} = \boldsymbol{S}_\mathrm{P} - \nabla p^{n+1}, \tag{5.88}$$

where

$$\begin{aligned}
\boldsymbol{A}_\mathrm{P} &= \frac{1}{\Delta t} + \frac{1}{\Delta V}\sum_{S_{f_i}} F_{f_i}(1-\lambda_i) + \frac{1}{\Delta V}\sum_{S_{f_i}} \nu \frac{\|\boldsymbol{S}_{f_i}\|}{\|\boldsymbol{d}\|} \\
\boldsymbol{A}_\mathrm{N} &= \frac{1}{\Delta V}\left(F_{f_i}\lambda_i - \nu\frac{\|\boldsymbol{S}_{f_i}\|}{\|\boldsymbol{d}\|}\right) \\
\boldsymbol{S}_\mathrm{P} &= \frac{1}{\Delta t}\boldsymbol{u}_\mathrm{P}^n.
\end{aligned} \tag{5.89}$$

Here, $\boldsymbol{A}_\mathrm{P}$ represents the diagonal matrix coefficients while $\boldsymbol{A}_\mathrm{N}$ denotes off-diagonal matrix coefficients. This notation allows for compact descriptions of the algorithm and its implementations in OpenFOAM. Combined with the continuity equation for $\boldsymbol{u}^{n+1}$, viz.

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0. \tag{5.90}$$

we have the complete system for $\boldsymbol{u}^{n+1}, p^{n+1}$.

In the PIMPLE algorithm, the velocity and pressure are solved separately and iteratively to obtain the approximated solution $(\boldsymbol{u}_\mathrm{P}^{\divideontimes}, \boldsymbol{u}_\mathrm{N}^{\circledast}, p^\star)$ satisfying

$$\boldsymbol{A}_\mathrm{P}\boldsymbol{u}_\mathrm{P}^{\divideontimes} + \sum \boldsymbol{A}_\mathrm{N}\boldsymbol{u}_\mathrm{N}^{\circledast} = \boldsymbol{S}_\mathrm{P} - \nabla p^\star. \tag{5.91}$$

$\boldsymbol{u}_\mathrm{P}^{\divideontimes}, \boldsymbol{u}_\mathrm{N}^{\circledast}$ and $p^\star$ can be obtained at different levels, and $\divideontimes, \circledast, \star$ stand for where the solution is solved; but they need to be solved iteratively to obtain correct values for $\boldsymbol{u}_\mathrm{P}^{n+1}, p^{n+1}$. Subtracting Equation (5.91) from Equation (5.88), the error equation is derived as

$$\boldsymbol{A}_\mathrm{P}\boldsymbol{u}_\mathrm{P}^{'} + \sum \boldsymbol{A}_\mathrm{N}\boldsymbol{u}_\mathrm{N}^{'} = -\nabla p^{'}. \tag{5.92}$$

where

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{'} &= \boldsymbol{u}_{\mathrm{P}}^{n+1} - \boldsymbol{u}_{\mathrm{P}}^{\divideontimes} \\ \boldsymbol{u}_{\mathrm{N}}^{'} &= \boldsymbol{u}_{\mathrm{N}}^{n+1} - \boldsymbol{u}_{\mathrm{N}}^{\circledast} \\ p^{'} &= p^{n+1} - p^{\star}. \end{aligned} \tag{5.93}$$

For any solution from Equation (5.91), the solution error is governed by Equation (5.92). The solution correction can be obtained by solving this error equation. There are two different approaches to dealing with Equation (5.92) as follows.

Approach without correction The PIMPLE algorithm combines the SIMPLE and PISO algorithm [143]. For the predictor step, the velocity field $\boldsymbol{u}_{\mathrm{P}}^{*}$ is solved implicitly by

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{*} + \sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*} = \boldsymbol{S}_{\mathrm{P}} - \nabla p^{n}. \tag{5.94}$$

For first correction step, subtracting Equation (5.94) from Equation (5.88), the we have the correction equation

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{'} + \sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{'} = -\nabla p^{'}. \tag{5.95}$$

where

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{'} &= \boldsymbol{u}_{\mathrm{P}}^{n+1} - \boldsymbol{u}_{\mathrm{P}}^{*} \\ \boldsymbol{u}_{\mathrm{N}}^{'} &= \boldsymbol{u}_{\mathrm{N}}^{n+1} - \boldsymbol{u}_{\mathrm{N}}^{*} \\ p^{'} &= p^{n+1} - p^{n}. \end{aligned} \tag{5.96}$$

The error in current cell ($\boldsymbol{u}_{\mathrm{P}}^{'}$) is treated as the dominant factor here. This is to say that the error contributions from neighbor cells ($\boldsymbol{u}_{\mathrm{N}}^{'}$) are ignored. Then we have the approximation

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{'} \approx -\nabla p^{'}. \tag{5.97}$$

Considering Equations (5.94) and (5.95), the corrected velocity ($\boldsymbol{u}_{\mathrm{P}}^{**}$) is calculated by

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{**} &= \boldsymbol{u}_{\mathrm{P}}^{*} + \boldsymbol{u}_{\mathrm{P}}^{'} \\ &= \boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}}\right) - \boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{n} - \boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{'} \\ &= \mathrm{HbyA}^{*} - \boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{*}, \end{aligned} \tag{5.98}$$

where

$$\mathrm{HbyA}^{*} = \boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}}\right). \tag{5.99}$$

The notation of HbyA^{*} is widely used as a variable in OpenFOAM since it is easy to understand and program the algorithm. From continuity of velocity, i.e. $\nabla \cdot \boldsymbol{u}_{\mathrm{P}}^{**} = 0$, the Poisson for the corrected pressure is

$$\nabla \cdot \left(\boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{*}\right) = \nabla \cdot \mathrm{HbyA}^{*}. \tag{5.100}$$

Then the pressure (p^{*}) is obtained by solving Equation (5.100). After that, the velocity is computed by Equation (5.98). In fact, the corrected velocity and pressure satisfy

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{**} + \sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*} = \boldsymbol{S}_{\mathrm{P}} - \nabla p^{*}. \tag{5.101}$$

For the second corrector step, the new error equation is formulated by subtracting Equation (5.101) from Equation (5.88), as shown below

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{'}+\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{'}=-\nabla p^{'}. \tag{5.102}$$

where

$$\begin{aligned}\boldsymbol{u}_{\mathrm{P}}^{'}&=\boldsymbol{u}_{\mathrm{P}}^{n+1}-\boldsymbol{u}_{\mathrm{P}}^{**}\\ \boldsymbol{u}_{\mathrm{N}}^{'}&=\boldsymbol{u}_{\mathrm{N}}^{n+1}-\boldsymbol{u}_{\mathrm{N}}^{*}\\ p^{'}&=p^{n+1}-p^{*}.\end{aligned} \tag{5.103}$$

This is simplified again into Equation (5.97). The second corrected velocity ($\boldsymbol{u}_{\mathrm{P}}^{**}$) is calculated using Equations (5.97) and (5.101) as

$$\begin{aligned}\boldsymbol{u}_{\mathrm{P}}^{***}&=\boldsymbol{u}_{\mathrm{P}}^{**}+\boldsymbol{u}_{\mathrm{P}}^{'}\\ &=\boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{**}+\boldsymbol{S}_{\mathrm{P}}\right)-\boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{*}-\boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{'}\\ &=\mathrm{HbyA}^{**}-\boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{**},\end{aligned} \tag{5.104}$$

where

$$\mathrm{HbyA}^{**}=\boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{**}+\boldsymbol{S}_{\mathrm{P}}\right). \tag{5.105}$$

The continuity of velocity $\nabla\cdot\boldsymbol{u}_{\mathrm{P}}^{***}=0$ leads to the Poisson for the second corrected pressure as

$$\nabla\cdot\left(\boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{**}\right)=\nabla\cdot\mathrm{HbyA}^{**}. \tag{5.106}$$

Then the pressure (p^{**}) is obtained by solving Equation (5.106). After that, the velocity is updated by Equation (5.104). In fact, this corrected velocity and pressure satisfy

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{***}+\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{**}=\boldsymbol{S}_{\mathrm{P}}-\nabla p^{**}. \tag{5.107}$$

We consider two correction steps in the current study although more correction steps can be used. Within the two corrector steps, there are similarities in the computations, such as updating HbyA*, solving the pressure Poisson equation. The corrector step can be generalized for multiple corrector steps. Consequently, the PIMPLE algorithm is summarized as follows

- Outer loop for PIMPLE (pimple loop)
  - Solve the $\boldsymbol{u}^{*}$ using the Equation (5.94) with known p^{n}.
  - Inner loop for PIMPLE (pimple corrector)
    - Update HbyA* using $\boldsymbol{u}_{\mathrm{N}}^{*}$, i.e. $\mathrm{HbyA}^{*}=\boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*}+\boldsymbol{S}_{\mathrm{P}}\right)$.
    - Solve the pressure Poisson equation, Equation (5.100), for p^{*}.
    - Calculate the $\boldsymbol{u}_{\mathrm{P}}^{**}$ using Equation (5.98).
    - Save the $\boldsymbol{u}_{\mathrm{P}}^{**}$, p^{*} for next corrector.

Approach with correction This approach utilizes a different way to approximate the correction. The same predictor step shown in Equation (5.94) is used for solving the velocity field $\boldsymbol{u}_{\mathrm{P}}^{*}$. Therefore, the solution error satisfies Equation (5.95). For a more accurate computation, the velocity error of the current cells is assumed to be the weighted average of the neighbor cells, i.e.

$$\boldsymbol{u}_{\mathrm{P}}^{'} \approx \frac{\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{'}}{\sum \boldsymbol{A}_{\mathrm{N}}}. \tag{5.108}$$

Then we have $\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{'} = (\sum \boldsymbol{A}_{\mathrm{N}}) \boldsymbol{u}_{\mathrm{P}}^{'}$, which is used to replace all $\boldsymbol{u}_{\mathrm{N}}^{'}$ with $\boldsymbol{u}_{\mathrm{P}}^{'}$. Substituting into Equation (5.95), the correction equation is expressed as

$$(\boldsymbol{A}_{\mathrm{P}} + \sum \boldsymbol{A}_{\mathrm{N}}) \boldsymbol{u}_{\mathrm{P}}^{'} = -\nabla p^{'}. \tag{5.109}$$

The velocity correction is expressed as

$$\boldsymbol{u}_{\mathrm{P}}^{'} = -\boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{'}, \tag{5.110}$$

where

$$\boldsymbol{A}_{\mathrm{t}} = \boldsymbol{A}_{\mathrm{P}} + \sum \boldsymbol{A}_{\mathrm{N}}. \tag{5.111}$$

Adding up Equations (5.94) and (5.110), the corrected velocity ($\boldsymbol{u}_{\mathrm{P}}^{**}$) is calculated by

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{**} &= \boldsymbol{u}_{\mathrm{P}}^{*} + \boldsymbol{u}_{\mathrm{P}}^{'} \\ &= \boldsymbol{A}_{\mathrm{P}}^{-1} \left(-\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}} \right) - \boldsymbol{A}_{\mathrm{P}}^{-1} \nabla p^{n} - \boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{'} \\ &= \mathrm{HbyA}^{*} - \left(\boldsymbol{A}_{\mathrm{P}}^{-1} - \boldsymbol{A}_{\mathrm{t}}^{-1} \right) \nabla p^{n} - \boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{*}, \end{aligned} \tag{5.112}$$

where HbyA^{*} is defined by Equation (5.99). From continuity of $\boldsymbol{u}_{\mathrm{P}}^{**}$, i.e. $\nabla \cdot \boldsymbol{u}_{\mathrm{P}}^{**} = 0$, so the Poisson equation for corrected pressure is

$$\nabla \cdot \left(\boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{*} \right) = \nabla \cdot \mathrm{HbyA}^{*} - \left(\boldsymbol{A}_{\mathrm{P}}^{-1} - \boldsymbol{A}_{\mathrm{t}}^{-1} \right) \nabla p^{n}. \tag{5.113}$$

Then the pressure (p^{*}) is obtained by solving Equation (5.113). After that, the velocity is computed by Equation (5.112). In fact, the corrected velocity and pressure satisfy

$$\boldsymbol{A}_{\mathrm{P}} \boldsymbol{u}_{\mathrm{P}}^{**} + \sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{**} = \boldsymbol{S}_{\mathrm{P}} - \nabla p^{*}. \tag{5.114}$$

This can be proved as follows. Equation (5.112) is re-arranged as

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{**} &= \boldsymbol{A}_{\mathrm{P}}^{-1} \left(-\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}} \right) - \boldsymbol{A}_{\mathrm{P}}^{-1} \nabla p^{n} - \boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{'} \\ &\Rightarrow \\ \boldsymbol{A}_{\mathrm{P}} \boldsymbol{u}_{\mathrm{P}}^{**} &= -\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}} - \nabla p^{n} - \boldsymbol{A}_{\mathrm{P}} \boldsymbol{A}_{\mathrm{t}}^{-1} \nabla p^{'} \\ &= -\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}} - \nabla p^{n} + \boldsymbol{A}_{\mathrm{P}} \boldsymbol{u}_{\mathrm{P}}^{'} && \triangleright \text{Equation (5.110)} \\ &= -\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}} - \nabla p^{n} - \nabla p^{'} - \sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{'} && \triangleright \text{Equation (5.109)} \\ &= -\sum \boldsymbol{A}_{\mathrm{N}} \boldsymbol{u}_{\mathrm{N}}^{**} + \boldsymbol{S}_{\mathrm{P}} - \nabla p^{*}, \end{aligned} \tag{5.115}$$

where

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{**} &= \boldsymbol{u}_{\mathrm{P}}^{*} + \boldsymbol{u}_{\mathrm{P}}^{'} \\ \boldsymbol{u}_{\mathrm{N}}^{**} &= \boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{u}_{\mathrm{N}}^{'} \\ p^{*} &= p^{n} + p^{'}. \end{aligned} \tag{5.116}$$

For the second corrector step, the velocity correction is computed by Equation (5.110). The corrected velocity ($\boldsymbol{u}_{\mathrm{P}}^{***}$) is stated as

$$\begin{aligned} \boldsymbol{u}_{\mathrm{P}}^{***} &= \boldsymbol{u}_{\mathrm{P}}^{**} + \boldsymbol{u}_{\mathrm{P}}^{'} \\ &= \boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{**} + \boldsymbol{S}_{\mathrm{P}}\right) - \boldsymbol{A}_{\mathrm{P}}^{-1}\nabla p^{*} - \boldsymbol{A}_{\mathrm{t}}^{-1}\nabla p^{'} \\ &= \mathrm{HbyA}^{**} - \left(\boldsymbol{A}_{\mathrm{P}}^{-1} - \boldsymbol{A}_{\mathrm{t}}^{-1}\right)\nabla p^{*} - \boldsymbol{A}_{\mathrm{t}}^{-1}\nabla p^{**}, \end{aligned} \tag{5.117}$$

where

$$p^{**} = p^{*} + p^{'}. \tag{5.118}$$

The continuity of velocity $\boldsymbol{u}_{\mathrm{P}}^{**}$ will induce the Poisson for the corrected pressure

$$\nabla\cdot\left(\boldsymbol{A}_{\mathrm{t}}^{-1}\nabla p^{**}\right) = \nabla\cdot\mathrm{HbyA}^{**} - \left(\boldsymbol{A}_{\mathrm{P}}^{-1} - \boldsymbol{A}_{\mathrm{t}}^{-1}\right)\nabla p^{*}. \tag{5.119}$$

Then the pressure (p^{**}) is obtained by solving Equation (5.119). After that, the velocity is computed by Equation (5.117). Likewise, the corrected velocity and pressure satisfy

$$\boldsymbol{A}_{\mathrm{P}}\boldsymbol{u}_{\mathrm{P}}^{***} + \sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{***} = \boldsymbol{S}_{\mathrm{P}} - \nabla p^{**}. \tag{5.120}$$

The above-mentioned corrector steps can be generalized for multiple corrector steps. Thus, the PIMPLE algorithm with correction is summarized as follows

- Outer loop for PIMPLE (pimple loop)
  - Solve the $\boldsymbol{u}^{*}$ using the Equation (5.94) with known p^{n}.
  - Inner loop for PIMPLE (pimple corrector)
    - Update HbyA^{*} using $\boldsymbol{u}_{\mathrm{N}}^{*}$, i.e. $\mathrm{HbyA}^{*} = \boldsymbol{A}_{\mathrm{P}}^{-1}\left(-\sum \boldsymbol{A}_{\mathrm{N}}\boldsymbol{u}_{\mathrm{N}}^{*} + \boldsymbol{S}_{\mathrm{P}}\right)$.
    - Solve the pressure Poisson equation, Equation (5.113), for p^{*}.
    - Calculate the $\boldsymbol{u}_{\mathrm{P}}^{**}$ using Equation (5.112).
    - Save the $\boldsymbol{u}_{\mathrm{P}}^{**}$, p^{*} for next loop.

The matrix coefficients are reconstructed when the velocity object is constructed in each outer loop of the PIMPLE algorithm. Thus the nonlinearity is addressed.

PIMPLE ALGORITHM FOR ADJOINT PROBLEM

The adjoint problem governed by Equation (5.30) is described in a vector format as

$$\begin{aligned} R^{q} &= -\nabla\cdot\boldsymbol{u}_{a} = 0 \\ R_{i}^{u} &= -\frac{\partial \boldsymbol{u}_{a}}{\partial t} \underbrace{-\nabla\boldsymbol{u}_{a}\cdot\boldsymbol{u}}_{\mathrm{ATC}} - \nabla\cdot(\boldsymbol{u}\boldsymbol{u}_{a}) - \nabla\cdot(\nu\nabla\boldsymbol{u}_{a}) + \nabla p_{a} = 0, \end{aligned} \tag{5.121}$$

where the adjoint transported convection (ATC) term distinguishes the adjoint equation from the primal problem. $\boldsymbol{u}_a, p_a$ denote the adjoint velocity vector and adjoint pressure.

The FVM is applied on an arbitrary volume, ΔV, with $t \in I : [t^n, t^{n+1}]$, in which the solution at $t = t^{n+1}$ is known for the adjoint problem since it is solved backward in time. Similar to the flow problem, the time-dependent adjoint problem in Equation (5.121) is solved with a semi-discretized form as follows,

$$\frac{\partial \boldsymbol{u}_{a,\mathrm{P}}}{\partial t} = \frac{1}{\Delta V}\int_V \left(-\nabla \boldsymbol{u}_a \cdot \boldsymbol{u} - \nabla\cdot(\boldsymbol{u}\boldsymbol{u}_a) - \nabla\cdot(\nu\nabla\boldsymbol{u}_a) + \nabla p_a\right)\mathrm{d}V\,, \tag{5.122}$$

where V represents the control volume and $\boldsymbol{u}_{a,\mathrm{P}}$ represents the adjoint velocity at current cell. Applying this on each mesh cell with the linear interpolation scheme at face S_{f_i}, we obtain the expression in the discrete space. The adjoint transported convection (ATC) term is expressed as

$$\begin{aligned}
\int_V -\nabla \boldsymbol{u}_a \cdot \boldsymbol{u}\,\mathrm{d}V &\approx -\int_V \nabla \boldsymbol{u}_a\,\mathrm{d}V \cdot \boldsymbol{u}_{\mathrm{P}} \\
&= -\sum_{S_f}\int_{S_f} \boldsymbol{n}(\boldsymbol{u}_a\cdot\boldsymbol{u}_{\mathrm{P}})\,\mathrm{d}S \\
&\approx -\sum_{S_f}(\boldsymbol{u}_{a,\mathrm{Cf}}\cdot\boldsymbol{u}_{\mathrm{P}})\,\boldsymbol{S}_f \\
&= -\sum_{S_{f_i}}\left[(1-\lambda_{f_i})\boldsymbol{u}_{a,\mathrm{P}} + \lambda_{f_i}\boldsymbol{u}_{a,\mathrm{N}_i}\right]\cdot\boldsymbol{u}_{\mathrm{P}}\,\boldsymbol{S}_{f_i} \qquad \triangleright\text{Equation (5.81)} \\
&= -\sum_{S_{f_i}}\left[(1-\lambda_{f_i})\boldsymbol{S}_{f_i}\boldsymbol{u}_{a,\mathrm{P}}\cdot\boldsymbol{u}_{\mathrm{P}} + \lambda_{f_i}\boldsymbol{S}_{f_i}\boldsymbol{u}_{a,\mathrm{N}_i}\cdot\boldsymbol{u}_{\mathrm{P}}\right],
\end{aligned} \tag{5.123}$$

where the subscripts $_{\mathrm{P}}$ and $_{\mathrm{Cf}}$ denote where the variables are defined, i.e. at the cell center of the present cell and the face center, respectively. The subscript $_{\mathrm{N}}$ denotes that adjoint variables are defined at the cell center of neighbour cells. The outer product of $\boldsymbol{S}_{f_i}\boldsymbol{u}_{a,\mathrm{P}} \equiv \boldsymbol{S}_{f_i}\otimes\boldsymbol{u}_{a,\mathrm{P}}$ can be computed by the vector/matrix multiplications as $\boldsymbol{S}_{f_i}\boldsymbol{u}_{a,\mathrm{P}}^{\mathsf{T}}$.

Likewise, the convection and viscous terms are evaluated as

$$\begin{aligned}
\int_V -\nabla\cdot(\boldsymbol{u}\boldsymbol{u}_a)\,\mathrm{d}V &= -\int_S (\boldsymbol{n}\cdot\boldsymbol{u})\boldsymbol{u}_a\,\mathrm{d}S \\
&= -\sum_{S_f}\int_{S_f}(\boldsymbol{n}\cdot\boldsymbol{u})\boldsymbol{u}_a\,\mathrm{d}S \\
&\approx -\sum_{S_f}\underbrace{\left(\boldsymbol{S}_f\cdot\boldsymbol{u}_{\mathrm{Cf}}\right)}_{F_f}\boldsymbol{u}_{a,\mathrm{Cf}} \\
&= -\sum_{S_{f_i}}\boldsymbol{S}_f\cdot[(1-\lambda_i)\boldsymbol{u}_{\mathrm{P}} + \lambda_i\boldsymbol{u}_{\mathrm{N}}]\left[(1-\lambda_{f_i})\boldsymbol{u}_{a,\mathrm{P}} + \lambda_{f_i}\boldsymbol{u}_{a,\mathrm{N}_i}\right] \\
&= -\sum_{S_{f_i}}F_{f_i}\left[(1-\lambda_{f_i})\boldsymbol{u}_{a,\mathrm{P}} + \lambda_{f_i}\boldsymbol{u}_{a,\mathrm{N}_i}\right],
\end{aligned} \tag{5.124}$$

$$\begin{aligned}\int_V -\nabla\cdot(\nu\nabla\boldsymbol{u}_a)\,\mathrm{d}V &= -\int_S \boldsymbol{n}\cdot(\nu\nabla\boldsymbol{u}_a)\,\mathrm{d}S \\ &= -\sum_{S_f}\int_{S_f}\boldsymbol{n}\cdot(\nu\nabla\boldsymbol{u}_a)\,\mathrm{d}S \\ &\approx -\sum_{S_f}\boldsymbol{S}_f\cdot(\nu\nabla\boldsymbol{u}_a)_{\mathrm{Cf}} \\ &\approx -\sum_{S_{f_i}}\|\boldsymbol{S}_{f_i,d}\|\frac{(\nu\boldsymbol{u}_a)_{\mathrm{N}_i}-(\nu\boldsymbol{u}_a)_{\mathrm{P}}}{\|\boldsymbol{d}_{f_i}\|} \qquad \triangleright \text{Equation (5.126)} \\ &= \sum_{S_{f_i}}\nu\frac{\|\boldsymbol{S}_{f_i,d}\|}{\|\boldsymbol{d}_{f_i}\|}(\boldsymbol{u}_{a,\mathrm{P}}-\boldsymbol{u}_{a,\mathrm{N}_i})\,.\end{aligned} \tag{5.125}$$

F_f is the convection face flux. The face gradient on an orthogonal mesh can be calculated by

$$\boldsymbol{S}_f\cdot(\nu\nabla\boldsymbol{u}_a)_{\mathrm{Cf}} = |\boldsymbol{S}_f|\,\boldsymbol{n}\cdot(\nu\nabla\boldsymbol{u}_a)_{\mathrm{Cf}} = |\boldsymbol{S}_f|\,(\nu\frac{\partial\boldsymbol{u}_a}{\partial n})_{\mathrm{Cf}} = \|\boldsymbol{S}_f\|\frac{(\nu\boldsymbol{u}_a)_{\mathrm{N}}-(\nu\boldsymbol{u}_a)_{\mathrm{P}}}{\|\boldsymbol{d}\|}\,, \tag{5.126}$$

where $\boldsymbol{d}$ represents the distance vector from the present-cell center to the neighbor-cell center (or face center at boundaries). This gradient flux needs to be corrected on a mesh with non-orthogonal cells.

The term involving ATC will be evaluated explicitly in the PIMPLE algorithm and is stated as

$$\boldsymbol{S}_{\mathrm{ATC}} = \frac{1}{\Delta V}\int_V -\nabla\boldsymbol{u}_a\cdot\boldsymbol{u}\,\mathrm{d}V\,. \tag{5.127}$$

Then the system is given as

$$\frac{\partial\boldsymbol{u}_{a,\mathrm{P}}}{\partial t} = \frac{1}{\Delta V}\int_V\{-\nabla\cdot(\boldsymbol{u}\boldsymbol{u}_a)-\nabla\cdot(\nu\nabla\boldsymbol{u}_a)+\nabla p_a\}\,\mathrm{d}V+\boldsymbol{S}_{\mathrm{ATC}}\,, \tag{5.128}$$

Substituting Equations (5.124) and (5.125) into Equation (5.128), we obtain the following system of ordinary equations

$$\frac{\partial\boldsymbol{u}_{a,\mathrm{P}}}{\partial t} = f(\boldsymbol{u}_a,p_a) = \boldsymbol{A}_{\mathrm{S,P}}\boldsymbol{u}_{a,\mathrm{P}}+\sum_{S_{f_i}}\boldsymbol{A}_{\mathrm{S,N}_i}\boldsymbol{u}_{a,\mathrm{N}_i}+\nabla p_a+\boldsymbol{S}_{\mathrm{ATC}}\,, \tag{5.129}$$

where the $\boldsymbol{A}_{\mathrm{S,P}}$ and $\boldsymbol{A}_{\mathrm{S,N}_i}$ denote the coefficient matrices for velocity values of at the present and neighbor cells, and they are expressed as

$$\begin{aligned}\boldsymbol{A}_{\mathrm{S,P}} &= \frac{1}{\Delta V}\left(-\sum_{S_{f_i}}F_{f_i}(1-\lambda_{f_i})\mathrm{I}+\sum_{S_{f_i}}\nu\frac{\|\boldsymbol{S}_{f_i,d}\|}{\|\boldsymbol{d}_{f_i}\|}\mathrm{I}\right) \\ \boldsymbol{A}_{\mathrm{S,N}_i} &= -\frac{1}{\Delta V}\left(F_{f_i}\lambda_{f_i}+\nu\frac{\|\boldsymbol{S}_{f_i,d}\|}{\|\boldsymbol{d}_{f_i}\|}\right)\mathrm{I},\end{aligned} \tag{5.130}$$

where I denotes the identity matrix.

Since we know actual solutions at t^{n+1}, viz. $\boldsymbol{u}_a^{n+1}$ and p_a^{n+1}, we need to compute $\boldsymbol{u}_a^n, p_a^n$ that satisfy Equation (5.129). The Euler backward scheme is used to demonstrate how to solve Equation (5.129) but other time integration schemes can be used as well. Then we have

$$\frac{\boldsymbol{u}_{a,\mathrm{P}}^n - \boldsymbol{u}_{a,\mathrm{P}}^{n+1}}{\Delta t} = -f(\boldsymbol{u}_a^n, p_a^n), \tag{5.131}$$

where

$$f(\boldsymbol{u}_a^n, p_a^n) = \boldsymbol{A}_{\mathrm{S,P}}^n \boldsymbol{u}_{a,\mathrm{P}}^n + \sum_{S_{f_i}} \boldsymbol{A}_{\mathrm{S,N}_i}^n \boldsymbol{u}_{a,\mathrm{N}_i}^n + \boldsymbol{S}_{\mathrm{ATC}} + \nabla p_a^n. \tag{5.132}$$

The superscript in $\boldsymbol{A}_{\mathrm{S,P}}^n$ denotes that the $\boldsymbol{u}_{\mathrm{P}}$ should be used at t^n, independent of adjoint velocity $\boldsymbol{u}_a^n$. These superscripts are ignored in the following content if there is no conflict in understanding. The ATC term is computed explicitly by

$$\begin{aligned} \boldsymbol{S}_{\mathrm{ATC}}^{n+1} &= \frac{1}{\Delta V}\int_V -\nabla \boldsymbol{u}_a^{n+1} \cdot \boldsymbol{u}^n \,\mathrm{d}V \\ &\approx -\frac{1}{\Delta V}\sum_{S_{f_i}} \left[(1-\lambda_{f_i}) \boldsymbol{S}_{f_i} \boldsymbol{u}_{a,\mathrm{P}}^{n+1} \cdot \boldsymbol{u}_{\mathrm{P}}^n + \lambda_{f_i} \boldsymbol{S}_{f_i} \boldsymbol{u}_{a,\mathrm{N}_i}^{n+1} \cdot \boldsymbol{u}_{\mathrm{P}}^n \right]. \end{aligned} \tag{5.133}$$

We should use $\boldsymbol{u}_{a,\mathrm{P}}^n$ rather than $\boldsymbol{u}_{a,\mathrm{P}}^{n+1}$ in Equation (5.133), but it affects the left-hand matrix. Instead, through this substitute, the ATC term will be evaluated explicitly at $t = t^{n+1}$ in the PIMPLE algorithm and becomes a source term in the discrete system. This is easy to implement in OpenFOAM although it could reduce the convergence rate. In order to guarantee the solution accuracy, the ATC term will be updated in each outer loop of the PIMPLE algorithm, which is similar to the idea of corrections in the PISO algorithm.

Then the $\boldsymbol{u}_a^n, p_a^n$ satisfy

$$\boldsymbol{A}_{a,\mathrm{P}} \boldsymbol{u}_{a,\mathrm{P}}^n + \sum_{S_{f_i}} \boldsymbol{A}_{a,\mathrm{N}_i} \boldsymbol{u}_{a,\mathrm{N}_i}^n = \boldsymbol{S}_{a,\mathrm{P}} - \nabla p_a^n, \tag{5.134}$$

where

$$\begin{aligned} \boldsymbol{A}_{a,\mathrm{P}} &= \frac{1}{\Delta t}\mathrm{I} + \boldsymbol{A}_{\mathrm{S,P}} = \left(\frac{1}{\Delta t} - \frac{1}{\Delta V}\sum_{S_{f_i}} F_{f_i}(1-\lambda_{f_i}) + \frac{1}{\Delta V}\sum_{S_{f_i}} \nu \frac{\|\boldsymbol{S}_{f_i,d}\|}{\|\boldsymbol{d}_{f_i}\|} \right)\mathrm{I} \\ \boldsymbol{A}_{a,\mathrm{N}_i} &= \boldsymbol{A}_{\mathrm{S,N}_i} = -\frac{1}{\Delta V}\left(F_{f_i}\lambda_{f_i} + \nu \frac{\|\boldsymbol{S}_{f_i,d}\|}{\|\boldsymbol{d}_{f_i}\|} \right)\mathrm{I} \\ \boldsymbol{S}_{a,\mathrm{P}} &= \frac{1}{\Delta t}\boldsymbol{u}_{a,\mathrm{P}}^{n+1} - \boldsymbol{S}_{\mathrm{ATC}}^{n+1}. \end{aligned} \tag{5.135}$$

At the same time, the adjoint velocity should satisfy the mass conservation,

$$\nabla \cdot \boldsymbol{u}_{a,\mathrm{P}}^n = 0. \tag{5.136}$$

Since Equations (5.134) and (5.136) have forms similar to those of Equations (5.88) and (5.90), the adjoint problem can be solved with approaches mentioned above with and without corrections. The PIMPLE algorithm for the adjoint problem with correction is presented

here. In each outer loop of the PIMPLE algorithm, we first solve $\boldsymbol{u}^*_{a,\mathrm{P}}$ in the predictor step,

$$\boldsymbol{A}_{a,\mathrm{P}}\boldsymbol{u}^*_{a,\mathrm{P}} + \sum_{S_{f_i}} \boldsymbol{A}_{a,\mathrm{N}_i}\boldsymbol{u}^*_{a,\mathrm{N}_i} = \boldsymbol{S}_{a,\mathrm{P}} - \nabla p_a^{n+1}, \tag{5.137}$$

where p_a^{n+1} is the known value at t^{n+1}. Then the correction equation is obtained by subtracting Equation (5.137) from Equation (5.134),

$$\boldsymbol{A}_{a,\mathrm{P}}\boldsymbol{u}^{'}_{a,\mathrm{P}} + \sum_{S_{f_i}} \boldsymbol{A}_{a,\mathrm{N}_i}\boldsymbol{u}^{'}_{a,\mathrm{N}_i} = -\nabla p^{'}_a, \tag{5.138}$$

where

$$\begin{aligned} \boldsymbol{u}^{'}_{a,\mathrm{P}} &= \boldsymbol{u}^{n}_{a,\mathrm{P}} - \boldsymbol{u}^*_{a,\mathrm{P}} \\ \boldsymbol{u}^{'}_{a,\mathrm{N}} &= \boldsymbol{u}^{n}_{a,\mathrm{N}} - \boldsymbol{u}^*_{a,\mathrm{N}} \\ p^{'}_a &= p^n_a - p^{n+1}_a. \end{aligned} \tag{5.139}$$

Considering the assumption used in Equation (5.108), the correction equation Equation (5.138) is simplified into

$$(\boldsymbol{A}_{a,\mathrm{P}} + \sum \boldsymbol{A}_{a,\mathrm{N}})\boldsymbol{u}^{'}_{a,\mathrm{P}} = -\nabla p^{'}_a. \tag{5.140}$$

The adjoint velocity correction is approximated by

$$\boldsymbol{u}^{'}_{a,\mathrm{P}} = -\boldsymbol{A}^{-1}_{a,\mathrm{t}}\nabla p^{'}_a, \tag{5.141}$$

where

$$\boldsymbol{A}_{a,\mathrm{t}} = \boldsymbol{A}_{a,\mathrm{P}} + \sum \boldsymbol{A}_{a,\mathrm{N}}. \tag{5.142}$$

Considering Equations (5.137) and (5.141), the corrected velocity ($\boldsymbol{u}^{**}_{a,\mathrm{P}}$) is calculated by

$$\begin{aligned} \boldsymbol{u}^{**}_{a,\mathrm{P}} &= \boldsymbol{u}^*_{a,\mathrm{P}} + \boldsymbol{u}^{'}_{a,\mathrm{P}} \\ &= \boldsymbol{A}^{-1}_{a,\mathrm{P}}\left(-\sum \boldsymbol{A}_{a,\mathrm{N}}\boldsymbol{u}^*_{a,\mathrm{N}} + \boldsymbol{S}_{a,\mathrm{P}}\right) - \boldsymbol{A}^{-1}_{a,\mathrm{P}}\nabla p^{n+1}_a - \boldsymbol{A}^{-1}_{a,\mathrm{t}}\nabla p^{'}_a \\ &= \mathrm{HbyA}^*_a - \left(\boldsymbol{A}^{-1}_{a,\mathrm{P}} - \boldsymbol{A}^{-1}_{a,\mathrm{t}}\right)\nabla p^{n+1}_s - \boldsymbol{A}^{-1}_{a,\mathrm{t}}\nabla p^*_a, \end{aligned} \tag{5.143}$$

where HbyA^*_a is defined

$$\mathrm{HbyA}^*_a = \boldsymbol{A}^{-1}_{a,\mathrm{P}}\left(-\sum \boldsymbol{A}_{a,\mathrm{N}}\boldsymbol{u}^*_{a,\mathrm{N}} + \boldsymbol{S}_{a,\mathrm{P}}\right). \tag{5.144}$$

Due to $\nabla\cdot\boldsymbol{u}^{**}_{a,\mathrm{P}} = 0$, the Poisson equation for corrected pressure is read as

$$\nabla\cdot\left(\boldsymbol{A}^{-1}_{a,\mathrm{P}}\nabla p^*_a\right) = \nabla\cdot\mathrm{HbyA}^*_a - \left(\boldsymbol{A}^{-1}_{a,\mathrm{P}} - \boldsymbol{A}^{-1}_{a,\mathrm{t}}\right)\nabla p^{n+1}_a. \tag{5.145}$$

Then the adjoint pressure (p^*) is obtained by solving Equation (5.145). After that, the velocity is computed by Equation (5.143). In fact, the corrected velocity and pressure should satisfy

$$\boldsymbol{A}_{a,\mathrm{P}}\boldsymbol{u}^{**}_{a,\mathrm{P}} + \sum \boldsymbol{A}_{a,\mathrm{N}}\boldsymbol{u}^{**}_{a,\mathrm{N}} = \boldsymbol{S}_{a,\mathrm{P}} - \nabla p^*_a. \tag{5.146}$$

This is similar to Equation (5.137), and thus the second/later corrector steps can be applied in this way. The PIMPLE algorithm with correction is summarized for the adjoint problem as follows

- Outer loop for PIMPLE (pimple loop)
  - Solve the $\boldsymbol{u}_a^*$ using the Equation (5.137) with known p^{n+1}.
  - Inner loop for PIMPLE (pimple corrector)
    - Update HbyA_a^* using $\boldsymbol{u}_{a,\text{N}}^*$, i.e. $\text{HbyA}_a^* = \boldsymbol{A}_{a,\text{P}}^{-1}\left(-\sum \boldsymbol{A}_{a,\text{N}}\boldsymbol{u}_{a,\text{N}}^* + \boldsymbol{S}_{a,\text{P}}\right)$.
    - Solve the pressure Poisson equation, Equation (5.145), for p_a^*.
    - Calculate the $\boldsymbol{u}_{\text{P}}^{**}$ using Equation (5.143).
    - Save the $\boldsymbol{u}_{a,\text{P}}^{**}$, p_a^* for next loop.

It is noted that $\boldsymbol{S}_{a,\text{P}}$ is only updated in outer loops.

COUPLING BETWEEN ADJOINT AND FLOW SOLVERS

The pointer list (`PtrList<T>`) in OpenFOAM is used to store the primal solution and reuse it when we solve the adjoint problem backward in time, as shown in Figure 5.6. Since the primal solution is stored in a pointer list, we need to know the solution index so that we can access consistent values for the adjoint solver. This can be computed based on physical time. It is assumed that the primal and adjoint problems are solved using the same time step, avoiding interpolation errors. Alternatively, we can use the index counter for the primal solution. The primal solution requires a huge amount of storage for the many time steps used in unsteady flow problems. This storage issue will be circumvented when we apply the online SVD that will be discussed in the next section.

5.2.5. VALIDATION ON STEADY CYLINDER FLOW AT $Re = 40$

We consider the flow past a 2D circular cylinder at $Re = 40$ and first solve the unsteady adjoint problem. The cylinder diameter (L_d) is 2 m and the inlet velocity is 1 m/s. The kinematic viscosity of fluid $\nu = \mu/\rho$ is chosen as 0.025.

The second-order backward method is used for the time-marching scheme. The second-order linear upwind method is used for the convective term while the Gauss theorem is used for the viscous term. The linear interpolation scheme is considered for the computation of face flux or variables. The gradient is computed by the least-squares scheme.

The velocity is solved by Preconditioned bi-conjugate gradient (PBiCG) with DILU preconditioner while the pressure equation is solved by the geometric agglomerated algebraic multigrid solver (GAMG). The parameters for the PIMPLE algorithm are set as shown in Table 5.2.

Table 5.2: Parameters in PIMPLE algorithm

Parameters	Values
momentumPredictor	yes
consistent	true
nOuterCorrector	3
nCorrectors	2
nNonOrthogonalCorrectors	2

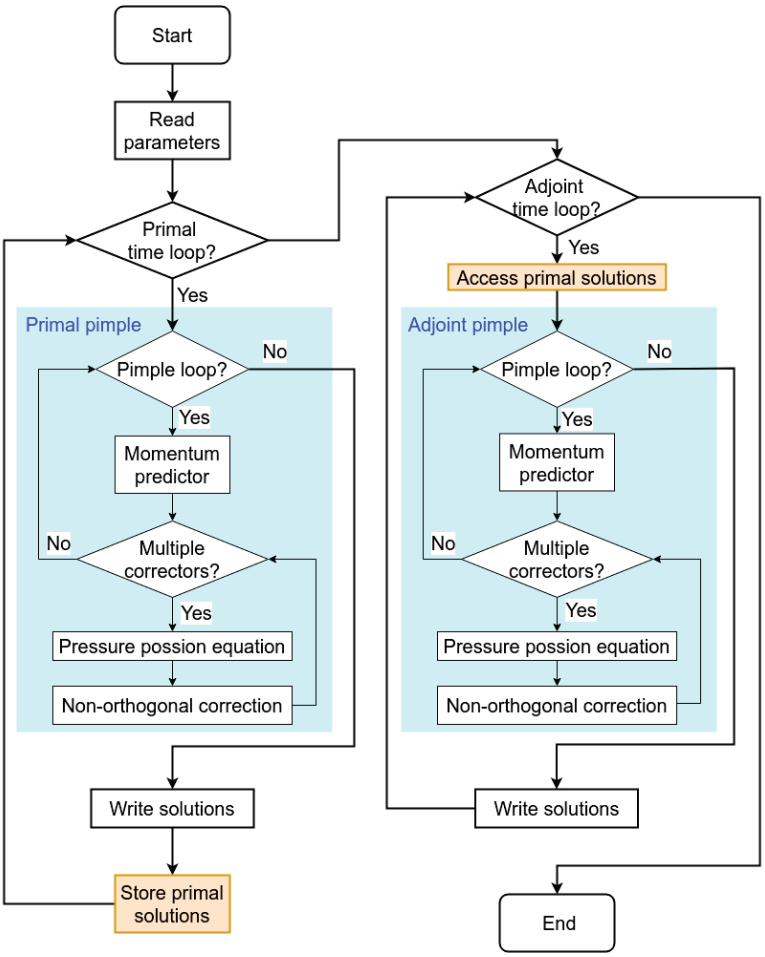


Figure 5.6: Coupling the primal and adjoint solver with access to the primal solutions.

GRID CONVERGENCE AND DOMAIN CONVERGENCE

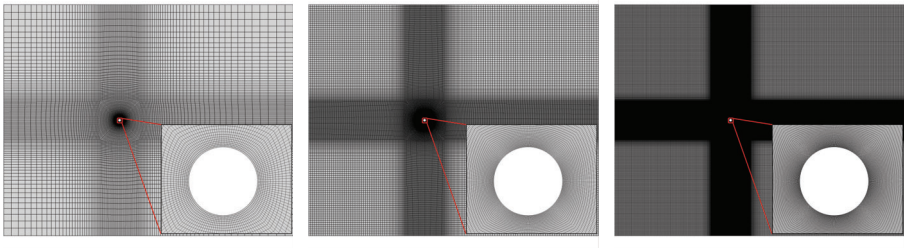
The computational grid was designed to have the first layer satisfying $y^+ \leq 1$, i.e. the cell height of 0.05. There is an O-shape structure around the cylinder and H-shape outside to the boundaries, as shown in Figure 5.7(a). This kind of mesh was used to capture the boundary layer and control the mesh density of the wake.

A convergence study on the size of the computational domain was carried out first. The cylinder is located at center of the domain. For the small computational domain, the inlet is imposed $10L_d$ before the cylinder while the outlet is at $15L_d$ after the cylinder. The upper and lower boundaries are chosen at $10L_d$ from the cylinder. The medium and large domains are doubled and quadrupled relative to the smallest one. For the 2D case, there is only one cell in the z direction with a length of 1 m. The calculation for the drag converges when the computational domain is enlarged, as shown in Table 5.3.

Table 5.3: Numerical parameters for the grid sensitivity study Computation grids for a 2D circular cylinder at Re = 40.

Grid	$\mathscr{G}^1$	$\mathscr{G}^2$	$\mathscr{G}^3$
Domain size			
$L_x \times L_y$	$25L_d \times 20L_d$	$50L_d \times 40L_d$	$100L_d \times 80L_d$
Grid parameters			
Number of mesh cells	12800	16800	15800
C_D	1.61159	1.54277	1.51569
Grid convergence	$\mathscr{G}_1^3$	$\mathscr{G}_2^3$	$\mathscr{G}_3^3$
Number of mesh cells	15800	63200	252800
C_D	1.51569	1.51693	1.51667

The grid convergence was studied using the largest computational domain, i.e. $\mathscr{G}^3$, with the computational meshes shown in Figure 5.7. In fact, the variation of drag computation C_D is less than 0.01% (see Table 5.3).



(a) Coarse (b) Medium (c) Finest

Figure 5.7: Computational meshes used in grid convergence study for the flow past a cylinder at $Re = 40$.

VALIDATION FOR FLOW SOLVER

A steady state is obtained after 700 non-dimensional time units. Table 5.4 compares the computations from the coarse mesh ($\mathscr{G}_1^3$) with the results in literature. The computation for separation angle and recirculation bubble length agrees well with the reference data.

The pressure coefficient is computed by

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2 S_{\text{REF}}}, \tag{5.147}$$

where $S_{\text{REF}} = L_d L_z$ denotes the reference area. The pressure coefficient over the cylinder surface agrees well with the literature, as shown in Figure 5.8. Thus we use this mesh for the analysis of the adjoint problem in the following study.

Table 5.4: Computation of drag coefficient (C_D), separation angle (θ_{sep}), and recirculation bubble length (L_r/d) for a 2D circular cylinder at Re = 40.

	C_D	θ_{sep} (°)	L_r/L_d
Park et al. [144]	1.51	126.41	2.24
Kim et al. [145]	1.51	–	–
Dennis et al. [146]	1.522	126.2	2.345
Meyer et al. [147]	1.56	134.6	2.28
Present	1.51569	125.93	2.125

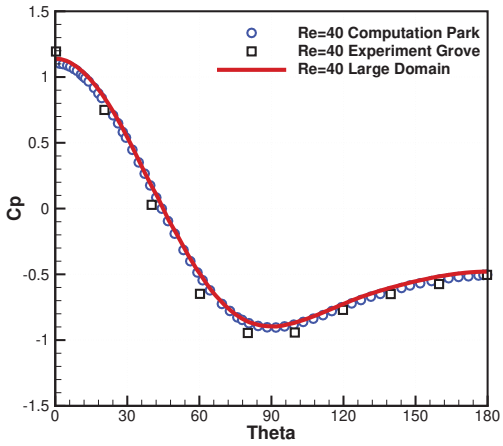


Figure 5.8: Pressure coefficients along the cylinder surface at $Re = 40$, compared to the computational [144] and experimental [148] data in literature.

ADJOINT PROBLEM

The adjoint problem is solved using the PIMPLE algorithm from $t = 900s$ to $t = 700s$, with a time step equivalent to the one used in the primal problem, viz. $\Delta t = 0.1s$. Figure 5.9 shows the evolution of l_2-norm of the adjoint velocity, with time looping right to left. It is observed that the adjoint velocity field gradually arrives at a steady state after a quick transition from the homogeneous initial condition. Figure 5.10 shows the instantaneous distribution for the magnitude of adjoint velocity. It is observed that the adjoint velocity field develops uniformly around the cylinder during a short transition period. The adjoint velocity field progresses upstream from the cylinder with a form similar to a jet. This jet also expands in the y-axis direction before settling down to a steady state. The field remains symmetric during the whole computational process. The drag is sensitive to the regions near the cylinder surface and upstream of it where the adjoint field shows large values. This agrees well with the study of a steady flow problem in [39]. A manufactured solution was used to verify the numerical solver for the adjoint problem and to study the mesh convergence of the adjoint problem. We employ a computational

mesh and outlet boundary condition that are different from what Wang et al. [39] used, but the results bear a strong resemblance, which is observed for the case at $Re = 100$ and 500. Therefore, it is believed that the computation of the adjoint field is accurate and reliable for the current study.

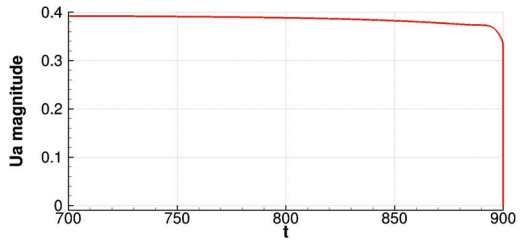


Figure 5.9: The evolution of l_2-norm of adjoint velocity for the steady flow past a 2D cylinder at $Re = 40$.

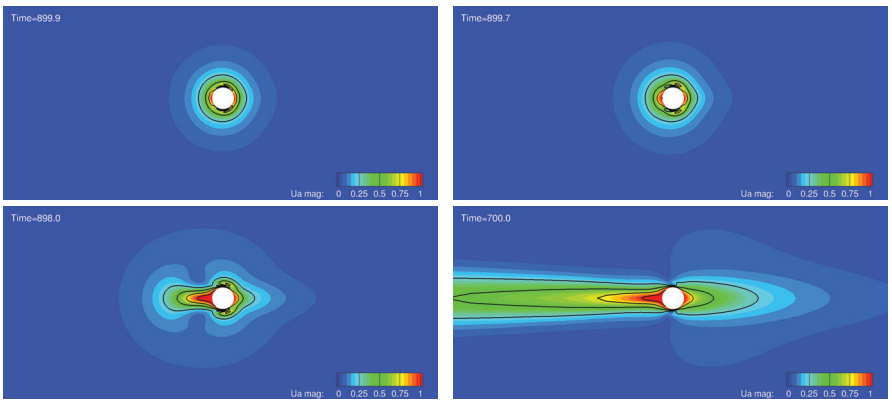


Figure 5.10: The magnitude of the adjoint velocity near the cylinder surface at $t = 899.9, 899.7, 898, 700$ for a 2D cylinder flow at $Re = 40$.

5.3. AMALGAMATION OF ONLINE SVD IN OPENFOAM

The online POD analysis is achieved using libROM as described in Chapter 3 and it is coupled with OpenFOAM here. The interfaces between two libraries are developed and validated for a 2D cylinder flow at $Re = 300$.

5.3.1. INTERFACE DESIGN

There are two interfaces that ought to be designed, i.e. an interface to generate libROM objects which can access the OpenFOAM data and an interface to reconstruct the reduced-order solution for OpenFOAM using libROM. These interfaces have been implemented using pointers for exchanging data between libROM and OpenFOAM, reducing the memory requirement for large-scale problems.

We take a vector field in OpenFOAM (`volVectorField`) to demonstrate the interfaces. Since OpenFOAM does not allow reallocation of memory for objects or variables, we first declare an OpenFOAM object (`volVectorField`), and then initialize the object in libROM (`CAROM::Vector`) with the data address of `volVectorField`. In this way, the memory assigned by OpenFOAM is reused in libROM.

The interface from `volVectorField` to `CAROM::Vector` is described here and shown in Listing 5.1. `UList.data()` denotes the address of the first vector in `volVectorField`. We need to utilize a `Vector` to gain the information (see line 11 in Listing 5.1). The size of the data is computed using the list length and vector size.

Listing 5.1: Creation of `libROM::Vector` using `OpeFOAM::volVectorField`.

```
1  // Declaration of volVectorField
2  volVectorField U_test1("U_test1", Ua);
3
4  // A reference to the first element of the UList
5  volVectorField::value_type & tmpT = U_test1[0];
6
7  // Data length of a VectorField
8  label nRow = U_test1.size()*tmpT.size();
9
10 // Declaration of CAROM::Vector
11 CAROM::Vector  snapVec(U_test1[0].v_, nRow, false, false);
```

The solution reconstruction is done after the SVD/POD analysis. The amplitude at a specific time is computed using the singular values and right singular vectors. Then a `volVectorField` is declared before reconstructing the reduced-order solution. The function in libROM is used to return the calculation by pointer or reference, and thus we can access the data directly. Listing 5.2 shows the interface used to reconstruct the `volVectorField` solution from libROM data. The implementation of the enhanced online algorithm in OpenFOAM is shown in Algorithm 5.

Listing 5.2: Creation of `OpeFOAM::volVectorField` using `libROM::Vector`.

```
1  // Create volVectorField
2  volVectorField U_recon("U_recon", Ua);
3
4  // Use a reference for reconstruction
5  CAROM::Vector snapRom(U_recon[0].v_, nRow, false, false);
6
7  // Create a reference to the amplitude
8  CAROM::Vector & ampVecRef = *ampVec;
9
10 // Access POD modes
11 const CAROM::Matrix* d_out;
12 d_out = inc_basis_generator.getSpatialBasis();
13
14 d_out->mult(ampVecRef, snapRom); // Reconstructed solution
```

Algorithm 5 Implementation of the enhanced online algorithm in OpenFOAM

```
Declaration of parameters and variables for incremental SVD.
Initialize the enhanced online algorithm.
while Time loop for flow solver do                    ▷ Loop for snapshots
    Read the volVectorField.
    Define a CAROM::Vector using the data of volVectorField.
    Take this sample for the enhanced online algorithm.
    Obtain the POD modes, singular values, and right singular vectors.
Output POD modes.
```

5.3.2. VALIDATION OF THE ONLINE SVD

The cylinder flow at $Re = 300$ is used to validate the online SVD/POD analysis. Enhanced incremental SVD is applied to the velocity field. We use 20 snapshots in four vortex-shedding periods. Figure 5.11 shows the x-component of the first four POD modes. They agree well with the results shown in Figure D.2 in [149].

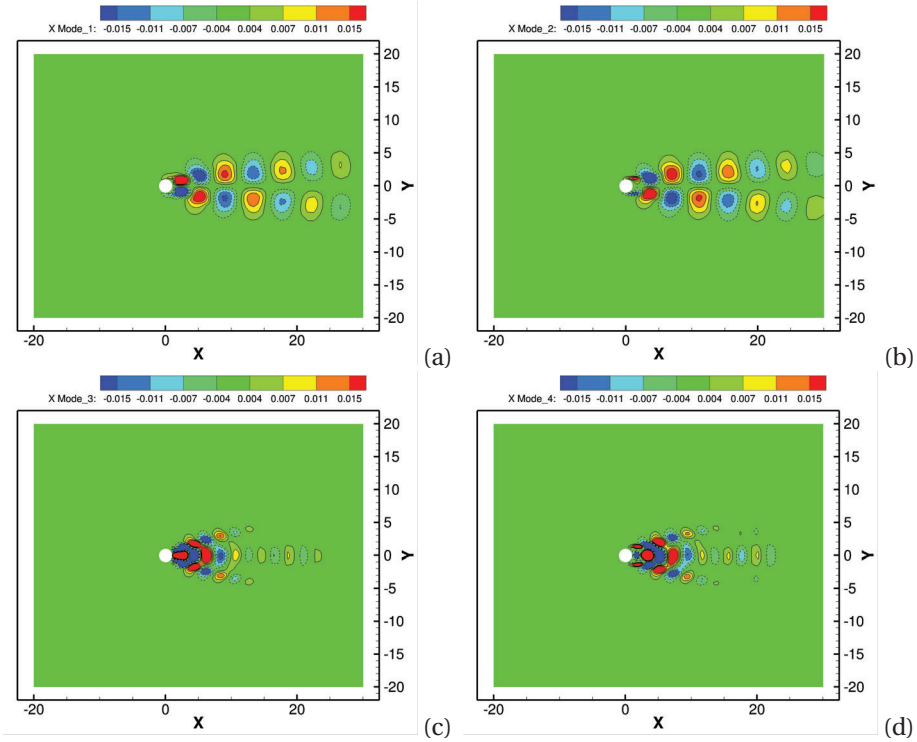


Figure 5.11: X-component of POD modes for a cylinder flow at Re =300 using 20 snapshots in four vortex shedding periods, (a)-(d) first to fourth POD modes.

As shown in Figure 5.12, the first two POD modes have a similar amount of energy

and thus are of equal importance for unsteady motions. They result from periodic vortexes and they exhibit a spatial-phase shift. This phase shift is also observed for the pair of third and fourth POD modes. The first six POD modes capture 99.86% of the total energy.

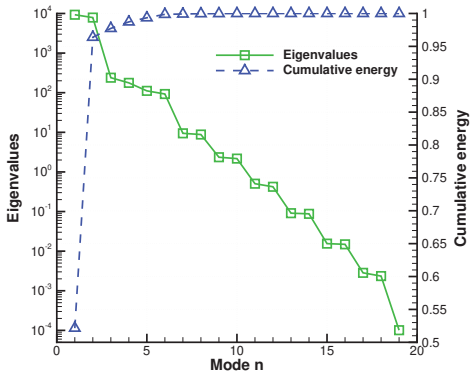


Figure 5.12: The eigenvalues and cumulative energy for a 2D cylinder flow at $Re = 300$.

Figure 5.13 shows the computational cost per incremental step using a mesh with 12800 volume cells. The cost increases significantly when a large truncation number is used. The cost of the standard incremental SVD can be much higher than solving the flow problem per step. The SVD then becomes the bottleneck for flow analyses. For the current study, truncations less than 50 were considered, which did not add significant costs.

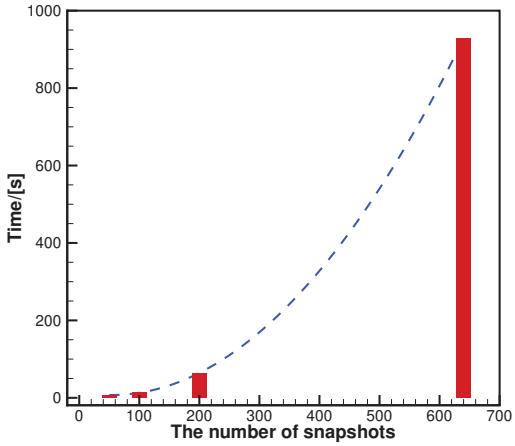


Figure 5.13: The computation time per incremental step with different truncation numbers for the enhanced online algorithm. The number of mesh cells is 12800. The blue dashed line denotes the fitted line for the trend.

5.3.3. USAGE OF ONLINE SVD WITH FLOW AND ADJOINT SOLVERS

Given that simulations for engineering applications can be high dimensional, a prototype is designed for using primal, ROR and adjoint solvers, as shown in Figure 5.14. Route 1 is used to solve the primal solution if we only have limited knowledge of the flow problem. Then we can start the simulation with the primal solver and adjoint solver, as shown in route 2. However, the resulting computational cost can be high. By virtue of using a ROR for large systems, we can instead follow route 3. At this point, it is possible to solve the adjoint with RORs directly (see route 3.B). On the other hand, the POD results can be stored in files without solving the adjoint equations, viz. route 3.A, and used for the computation of adjoint solutions as in route 4.

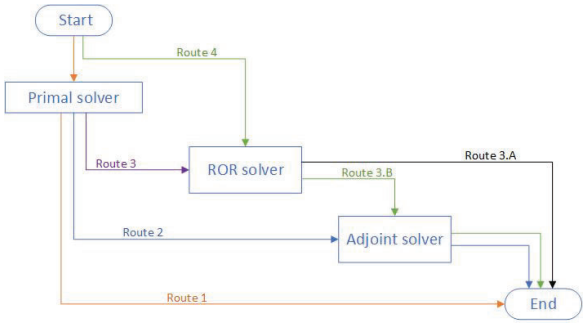


Figure 5.14: Different ways to run primal and adjoint solvers with/without a ROR solver.

5.4. IMPACTS OF RORS ON ADJOINT FIELD

We now employ RORs constructed with the enhanced online algorithm to the following unsteady flows around a cylinder, 2D at $Re = 100,500$ and 3D at $Re = 500$. The 2D flow past a circular cylinder at $Re = 100$ exhibits periodic vortex shedding. It has been reported that the system has a neutral Lyapunov exponent [39]. The 2D cylinder flow at $Re = 500$ plays an important role in understanding vortex-induced vibration [150]. The 3D cylinder flow at $Re = 500$ is turbulent and is used to investigate the effect of RORs on the growth of adjoint solutions in chaotic problems. Furthermore, the impacts of using RORs on the adjoint field are investigated, including the adjoint dynamics, computational accuracy and storage reduction as well as adjoint-based error estimations.

5.4.1. NUMERICAL SETUPS

The cylinder diameter (L_d) is chosen to be 1 meter. The freestream velocity is chosen as 1 m/s and thus the viscosity coeffic'ient ν is determined by the Reynolds number, i.e. 0.01 m^2/s for $Re = 100$ and 0.002 m^2/s for $Re = 500$. The computational mesh uses a domain on $\mathscr{G}^3$. The boundary cells are sized to have y^+ near the cylinder surface that is less than 1. There are 160 grid cells located on the cylinder surface and 25600 volume cells for the 2D cylinder cases. The 3D mesh is generated by extruding the 2D mesh for $4L_d$ in the spanwise direction (the z-axis here), with a uniform discretization using 80 cells. A second-order scheme is used for spatial discretization and a second-order

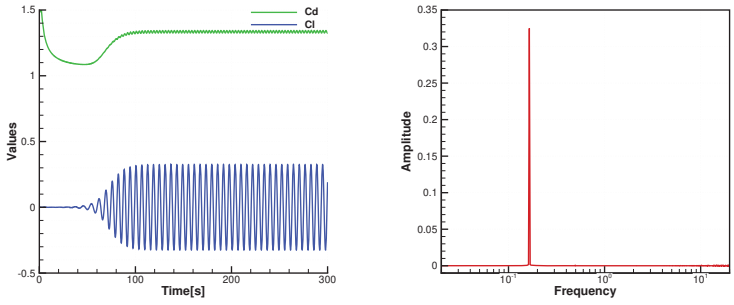
backward time scheme is used for discretization in time. The gradient of adjoint velocity and pressure is solved with the linear interpolation scheme. The flow and adjoint problems are solved using the PIMPLE algorithm as mentioned in Table 5.2. Their time steps are kept the same using a value computed based on the estimated Courant–Friedrichs–Lewy (CFL) number.

5.4.2. PRIMAL FLOW COMPUTATIONS

We first validate the primal flow computation for the three cases. In each case, the primal flow solver starts from a uniform field at the freestream velocity. The flow field is solved for 500 dimensionless time units (10 flow-through time periods) before computing time-averaged values.

2D CYLINDER FLOW AT $Re = 100$

The flow simulation at $Re = 100$ arrives at a statistically steady state after the transition period shown in Figure 5.15(a). The Strouhal number ($St = \frac{fL_d}{U_{\text{ref}}}$) is used to describe shedding vortexes, where f denotes the vortex shedding frequency and U_{ref} is equal to the freestream velocity. The vortex shedding frequency is analyzed by performing a Fast Fourier Transform (FFT) on lift coefficients as shown in Figure 5.15(b). The computed



(a) Time series of C_d and C_l (b) Frequency spectrum

Figure 5.15: The time series of lift and drag coefficients and the frequency spectrum of lift coefficients for the flow past a 2D cylinder at $Re = 100$.

St is equal to 0.16478, which has an excellent agreement with the data in literature as shown in Table 5.5.

The time period for vortex shedding is 6.068 based on the FFT analysis. We consider 30 vortex-shedding periods for averaging the flow fields , e.g. pressure, velocity and drag coefficient. The time-averaged separation angle is determined by the point of zero wall shear stress and the time-averaged recirculation bubble length is obtained by the zero value of the x-component velocity or the minimum velocity magnitude. They compare well with the data in literature as shown in Table 5.5. In addition, a good match with reference data is obtained for the mean drag coefficient and peak lift coefficient while the time-averaged wall pressure coefficients also compare well with the reference, as shown in Figure 5.16.

Table 5.5: Time-averaged drag coefficient ($\overline{C}_D$), peak lift coefficient ($C_{L,\text{peak}}$), separation angle (θ_{sep}), recirculation bubble length (L_r/L_d), and Strouhal number (St) for a 2D circular cylinder at Re = 100.

	$\overline{C}_D$	$C_{L,\text{peak}}$	θ_{sep} (°)	L_r/L_d	St
Park et al. [144] (Body-fitted mesh)	1.33	0.3321	117	1.3878	0.16468
Kim et al. [145] (Cartesian grid)	1.33	0.32	–	–	0.165
Henderson [151] (Unstructured mesh)	1.35	–	–	–	–
Meyer et al. [147] (Cartesian grid)	1.26	0.34	119	–	0.165
Karniadakis et al. [152] (Structured grid)	–	–	–	–	0.168
Present	1.3346	0.325	116.97	1.4	0.16478

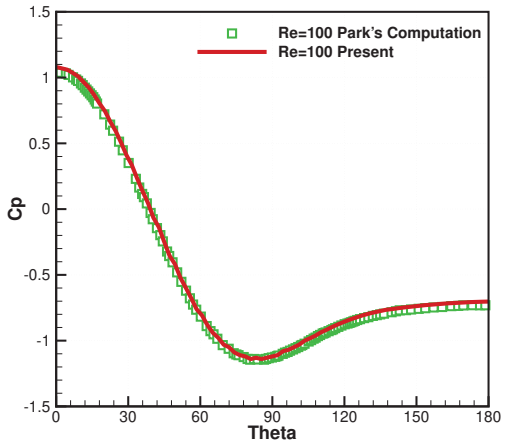


Figure 5.16: Pressure coefficients along the cylinder surface at $Re = 100$ compared to the reference [144].

2D CYLINDER FLOW AT $Re = 500$

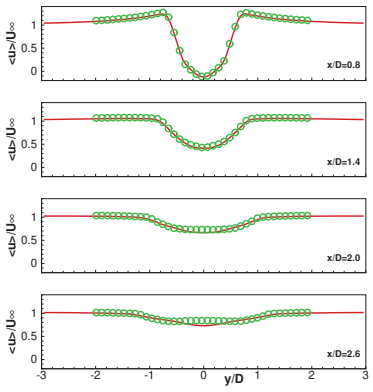
Table 5.6 shows the computation of 2D cylinder flow at $Re = 500$ compared to results from literature. Using frequency analysis, a time period of $T_{\text{vortex}} = 4.485$ was found for the vortex shedding. The St number is in good agreement with the reference data. This is also true for the mean drag coefficient and the peak of the lift coefficient. The averaging time was chosen as $21T_{\text{vortex}} = 94.185 \sim 94$. Figure 5.17(a) shows the profile of time-averaged velocity compared to the data in Ref. [153]. It agrees well in the wake region near the cylinder although there exists a discrepancy at downstream positions. This could be because of the coarse mesh in the far wake region. Figure 5.17(b) shows the vorticity field for the current case from which a rapid dissipation of vorticity can be observed in the far wake region.

3D CYLINDER FLOW AT $Re = 500$

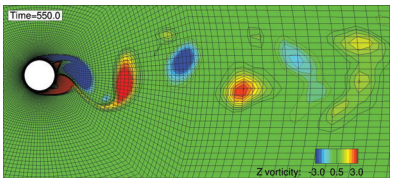
Figure 5.18 shows the time history of C_D and C_L and the frequency spectrum of C_L.The dominant frequency is obtained at $f_1 = 0.2099$ while the secondary tone frequency is

Table 5.6: Time-averaged drag coefficient ($\overline{C}_D$), peak lift coefficient ($C_{L,\text{peak}}$), and Strouhal number (St) for a 2D circular cylinder at Re = 500 with comparison to reference data in literature.

	St	$\overline{C}_D$	$C_{L,\text{peak}}$
Blackburn et al. [150]	0.228	1.46	1.2
Henderson [151]	–	1.445	–
Baek et al. [153]	0.235	1.529	1.247
Present	0.223	1.409	1.14



(a) Mean profile



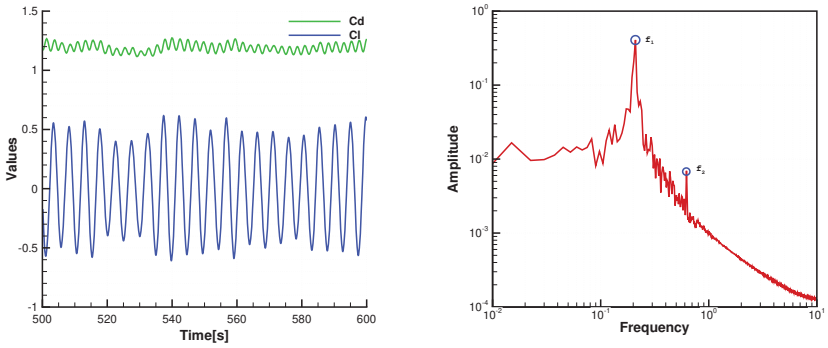
(b) Vorticity distribution

Figure 5.17: Profile of the x-component of the time-averaged velocity at different positions in the wake region for a 2D cylinder flow at $Re = 500$ and the distribution of vorticity. Solid lines denote the present computations while the green cycles are the data from [153].

$f_2 = 0.622$, which is consistent with the calculation in Ref. [154] (see Figure S5). The time period for vortex shedding is $T_{\text{vortes}} = 1/f_1 \approx 4.76$. We consider 21 cycles, i.e. $21T_{\text{vortes}} \approx 100$, for computing the time-averaged quantities. Table 5.7 compares the computation for St, the mean value of drag coefficient, and the root-mean-square of lift coefficients with data from literature. Our computations are well within the range reported in the literature.

Computations using a small and a large time step (the CFL numbers is roughly 1.7 and 4.3) are compared in Table 5.7. The St number from the large time step is similar to that from the small time step (less than 5% deviation). The difference of the mean drag coefficient is within 3.4% although the variation of $C_{L,\text{rms}}$ is more significant. Figure 5.19 shows the time-averaged pressure coefficients. They have a reasonable agreement with the reference before separation. Using a small CFL number results in a more accurate prediction in the rear region of the circular cylinder.

The iso-surface of Q-criterion [161] is used to demonstrate coherent structures in turbulent flows. A large-Q value denotes the region in which the local vorticity is more dominant than the shear strain rate. Figure 5.20 shows the iso-surfaces of $Q = 0.1$ at two time instants using small and large CFL numbers. The one with a small CFL number exhibits more complex structures with small scales. However, they have similar large-scale



(a) Time series (b) Frequency spectrum

Figure 5.18: Time series of lift and drag coefficients and the associated frequency spectrum of lift coefficients for a 3D cylinder flow at $Re = 500$.

Table 5.7: Time-averaged drag coefficient ($\overline{C}_D$), root-mean-square of lift coefficients ($C_{L,\text{rms}}$), and Strouhal number (St) for the 3D turbulent flow past a circular cylinder at $Re = 500$ compared to the data in literature.

	St	$\overline{C}_D$	$C_{L,\text{rms}}$
Zdravkovich [155]	–	1.154	–
Lienhard [156]	–	1.163	–
Roshko [157] (experimental)	0.208	–	–
Sirisup et al. [158]	0.22	–	–
Chen et al. [154] (fine mesh)	0.208	1.176	0.3195
Chen et al. [154] (coarse mesh)	0.206	1.195	–
Norberg [159]	0.2059	–	0.2395
Present (large CFL)	0.2099	1.188	0.3586
Present (small CFL)	0.20	1.149	0.2934

coherent structures both near the cylinder surface and in the wake region. Therefore, the adjoint solution and adjoint-based error of the 3D flow will be computed using the large CFL number.

5.4.3. ADJOINT DYNAMICS

In this section, we examine the dynamics of the adjoint problems of the three cases using their full-order primal solutions.

2D CYLINDER FLOW AT $Re = 100$

The primal flow problem arrived at a statistically-steady state after $t = 500$. The flow problem is then solved from $t = 500$ to 600 while the adjoint problem is solved from $t = 600$ to 500. This time interval includes 16 vortex shedding periods, which is the same time length considered in Ref. [39].

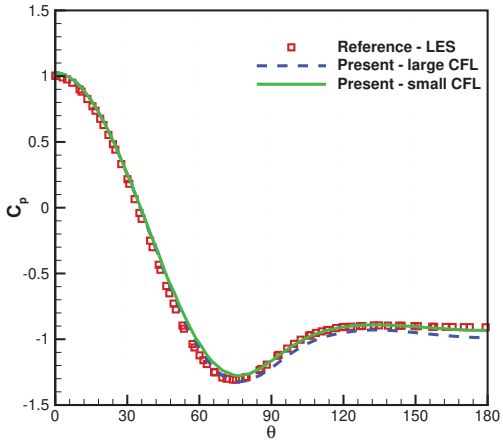


Figure 5.19: Distributions of time-averaged pressure coefficients for the flow past a 3D circular cylinder at $Re = 500$, compared to the data of LES [160].

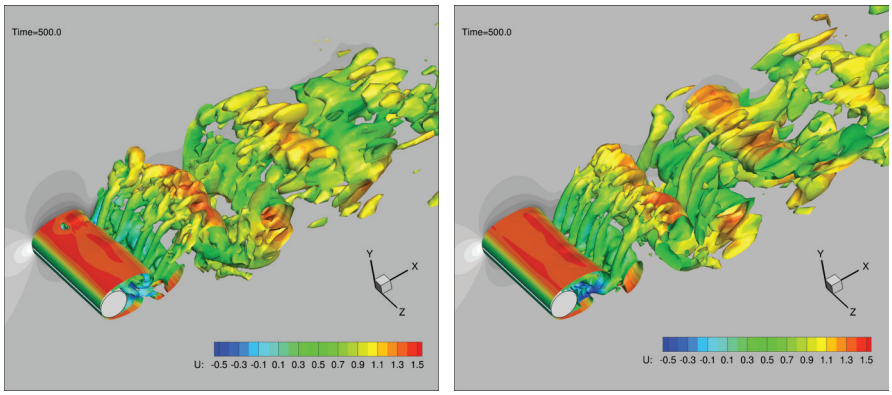


Figure 5.20: Instantaneous vortex shown by iso-surfaces of $Q = 0.1$ for a 3D cylinder flow at $Re = 500$ with small (left) and large (right) CFL numbers, colored by x-component of flow velocity. The pressure is displayed using a grey scale with dark grey representing high pressure and light grey color representing low pressure.

Figure 5.21 shows the l_2 norm of adjoint velocity over the computational domain. The adjoint velocity field increases monotonically for the first several time units and then grows with a small variation to a large oscillation. These are called the initial growth phase (phase I) and transit phase (phase II) here. The phase II lasts for 3 vortex-shedding periods. After that, the adjoint field gradually settles down to a periodic state. This is referred to as the settle-down phase. The oscillatory frequency of the adjoint flow is identical to that of the drag since the drag is the QoI and the adjoint problem is modulated by the change of the instantaneous drag.

Figure 5.22 shows the evolution of the magnitude and stream trace of adjoint velocity

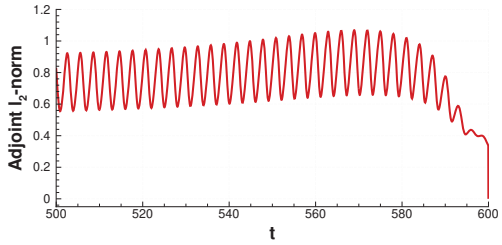


Figure 5.21: The l_2-norm of adjoint velocity for the 2D cylinder flow at $Re = 100$.

backward in time from t=598 to 580, with two contour lines for non-dimensional stream-wise flow velocity. We can observe that the adjoint field becomes more and more violent in the wake region from which the adjoint eddies initiate. These adjoint eddies are induced by periodic vortex shedding and propagated towards the cylinder and upstream. They are developed to larger and more regular eddies into the upstream region.

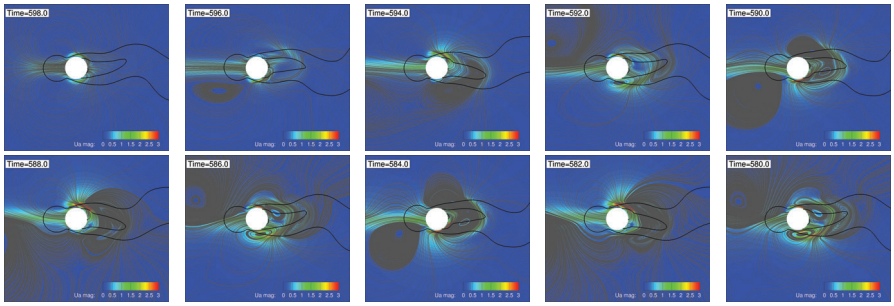


Figure 5.22: Evolution of the magnitude and stream trace (grey lines) of adjoint velocity for the flow around a 2D cylinder at $Re = 100$ from t=598 to 580. The thick black lines denote two contour lines of the non-dimensional stream-wise flow velocity at 0 and 0.8.

After the adjoint field is fully developed, periodic adjoint eddy shedding is observed. A bean-shaped eddy is also seen behind the cylinder, as has been reported by Ref. [39]. Figure 5.23 shows the evolution of the bean-shaped eddy as the adjoint problem is solved backward in time. The bean-shaped adjoint eddy forms in the shear region (i.e. near $u = 0$ in the wake). As observed by Wang et al. [39], this eddy will grow and stretch in the x-direction. This induces a λ-shaped eddy, which is split into two elongated eddies hereafter. One of the daughter eddies (the lower eddy here) is transported upstream and becomes smaller when passing by the cylinder surface. It grows rapidly as it moves into the upstream region before the cylinder, where the adjoint field exhibits high magnitudes and spreads out in the upstream direction afterwards. The other daughter eddy (the upper eddy here) is first captured into the wake just behind the cylinder and then vanishes in the upper region near the cylinder surface. This is because the division of the bean-shaped eddy happens when the vortex shedding is moving downward in the y-axis. Therefore, the upper eddy is shed downward, which forces the upper eddy towards the cylinder surface and leads to its disappearance. In contrast, the lower daughter eddy

is moved away from the cylinder surface and grows in the upstream region. The region where the daughter eddy disappears is then occupied by a subsequent daughter eddy originating from the next bean-shaped adjoint eddy, starting the next cycle of the adjoint field. An observation is that an upper bean-shaped eddy (named based on where the eddy originates) will induce adjoint eddies in lower-upstream regions whereas a lower bean-shaped eddy induces eddies in upper-upstream regions. A jet region is formed upstream after the adjoint field settles down.

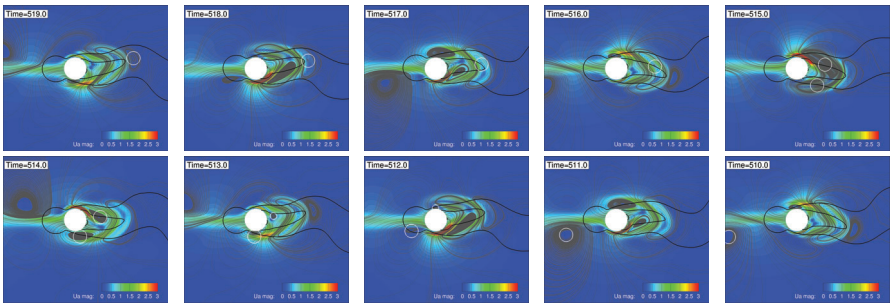


Figure 5.23: Evolution of adjoint eddies (white circles) for the flow around a 2D cylinder at $Re = 100$ after the adjoint flow is settled down. The thick black lines denote two contour lines x-component of the non-dimensional stream-wise flow velocity at 0 and 0.8.

Figure 5.24 compares the x ad y components, and the magnitude of the adjoint velocity in one vortex shedding period. The shear layer of the x ad y components denotes the adjoint eddies mentioned above. The x-component of the adjoint velocity contributes significantly to the magnitude of the adjoint velocity. The y component has an important influence in the wake region behind the cylinder, with a notable contribution in the upper and lower regions during downward and upward vortex shedding, respectively. This is distinct from the steady cylinder flow case.

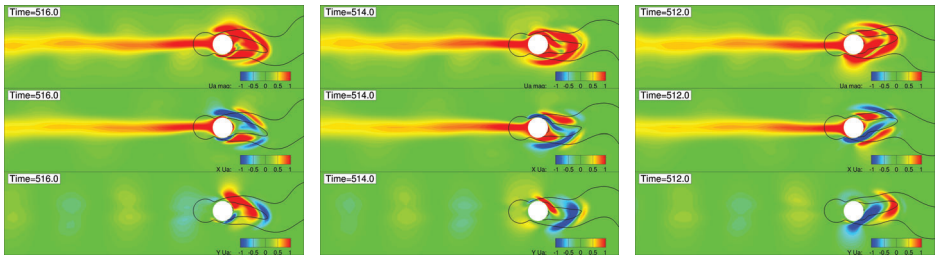


Figure 5.24: Comparison of the magnitude (top), x-component (middle) and y-components (bottom) of adjoint velocity in one vortex shedding period, from $t = 516$ to 511, for a 2D cylinder flow at $Re = 100$.

2D CYLINDER FLOW AT $Re = 500$

Figure 5.25 shows the history of the l_2-norm of the adjoint velocity as the adjoint problem is solved backward in time using full-order primal solutions. It shows that the adjoint

field develops monotonically during phase I and then grows gradually in an oscillatory manner during phase II. It settles down for a long time period afterwards.

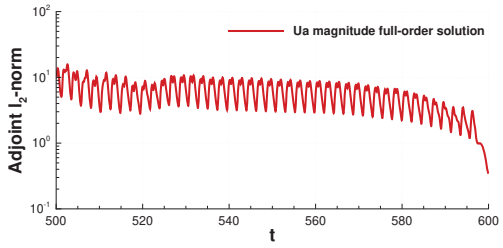


Figure 5.25: The l_2-norm of adjoint velocity for the 2D cylinder flow at $Re = 500$.

As shown in Figure 5.26, the bean-shaped adjoint eddy is also formed near the shear layer as well, but the adjoint eddy structures become more complex after the phase II

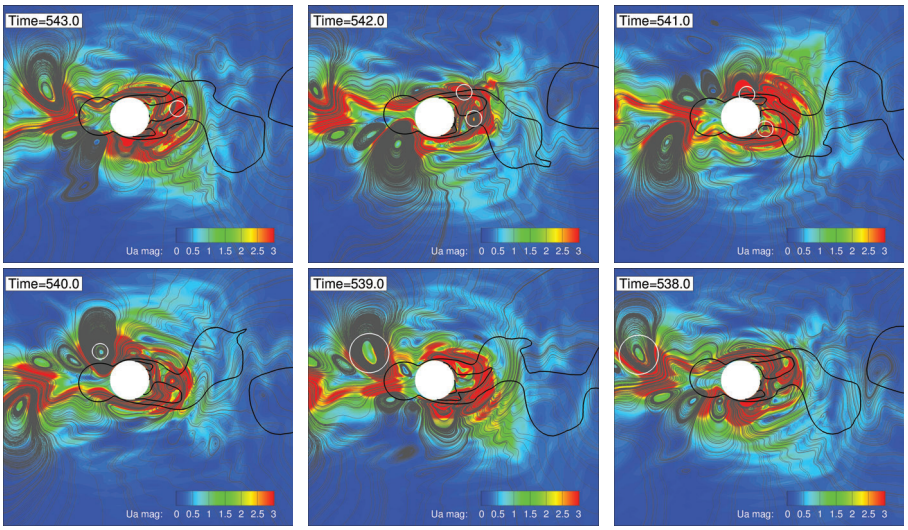


Figure 5.26: Evolution of adjoint eddies (white circles) for a flow around a 2D cylinder at $Re = 500$ after the adjoint flow was settled down ($t = 543 - 538$ in phase III). The thick black lines denote two contour lines of the non-dimensional stream-wise flow velocity at 0 and 0.8.

compared with the case at $Re = 100$. This eddy is split into two daughter eddies. One eddy is propagated upstream and amplified while the other vanishes near the cylinder surface, as in the low-Re case. Noted two other small eddies are generated as the daughter eddy moves past the cylinder. They exhibit an aligned pattern while moving away from the cylinder. These eddies will gradually merge into a large eddy when they move further upstream after $x = -8D$, as shown in Figure 5.27. In general, there are small eddies near the cylinder and large eddies away from the cylinder. Note that in the primal 2D flow vortices tend to merge, resulting in an inverse energy cascade. This is a key difference between the 2D and 3D cases. The adjoint eddies, in this case, are more diverse

in both scale and spatial distribution than those in the low-Re case. The figures show the evolution of an upper-daughter eddy whilst the next lower-daughter eddy emerges in the next cycle during the split of the adjoint bean-shaped eddy.

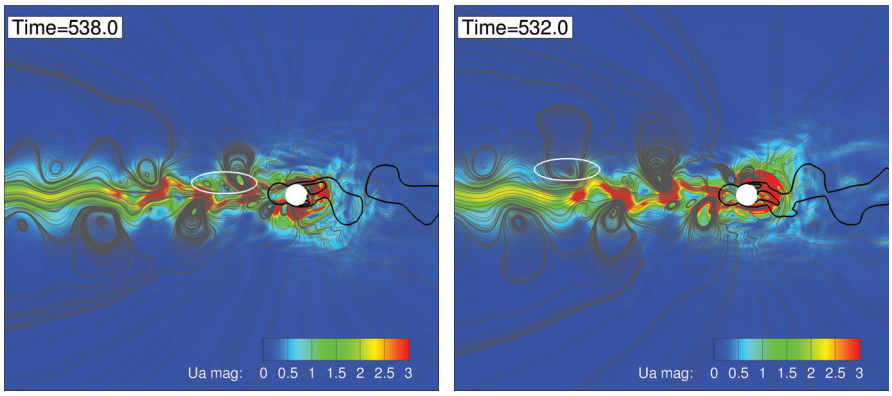


Figure 5.27: The adjoint eddies in upstream regions and the merging of adjoint eddies (see the white cycle) in phase III for the 2D cylinder flow at $Re = 500$.

Figure 5.28 compares the magnitude of the adjoint field with its associated x and y components. The x component primarily attributes to the adjoint field, especially up-

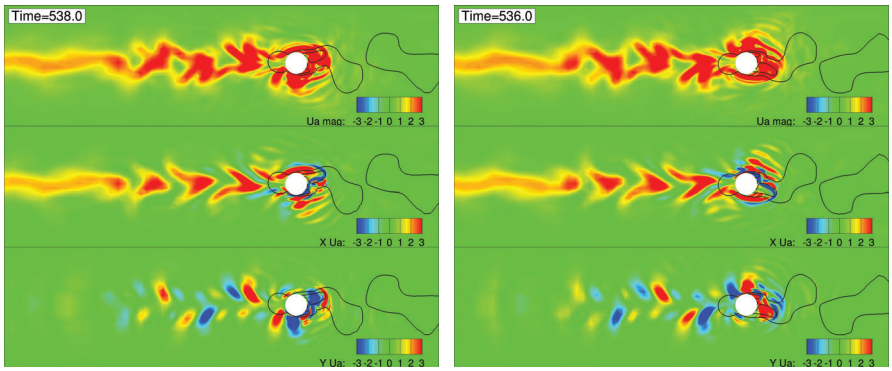


Figure 5.28: The magnitude of adjoint vorticity and associated x/y components for the 2D cylinder flow at $Re = 500$.

stream. More diverse structures are observed downstream of the cylinder surface for both components. It is noted that the triple-eddy pattern appears in the y-component in upstream regions, which can account for the aligned adjoint eddies.

Figure 5.29 compares the contour lines for the magnitude of the adjoint velocity during the entire evolution. It is observed that the adjoint field is developed upstream after phase II (see $t = 580$). A statistically-steady state is reached in phase III (see snapshots at $t = 560, 540, 520$). The adjoint field begins to grow as the high-value regions expand at $t = 500$. There are significant contributions in the vortex street during this phase. How-

ever, this growth is less significant since it is still within the same order of magnitude for the adjoint velocity, as shown in Figure 5.25.

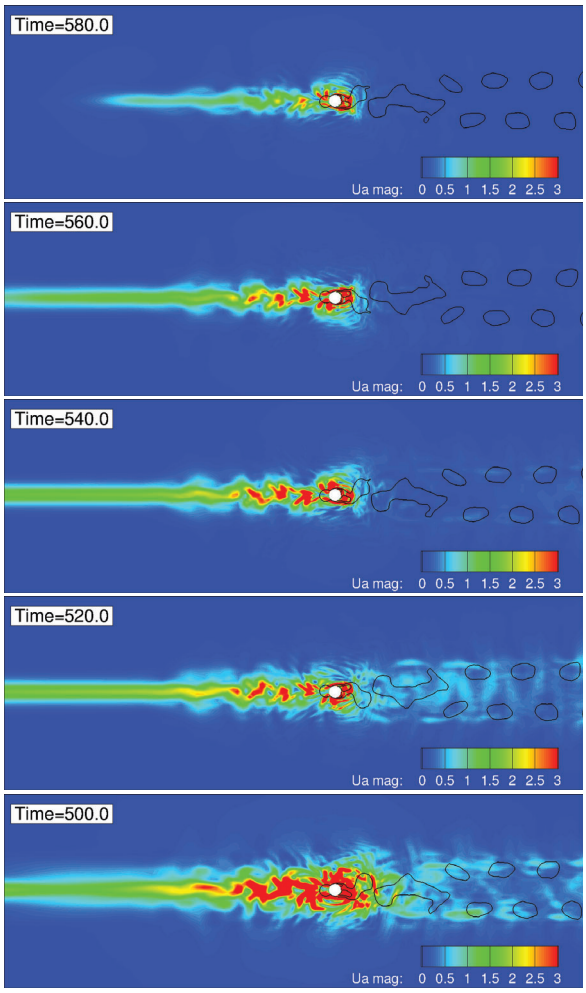


Figure 5.29: The magnitude of adjoint velocity for the 2D cylinder flow at $Re = 500$ at different times. Solid lines denote the x-component of the primal velocity, $u = 0, 0.8$.

3D CYLINDER FLOW AT $Re = 500$

The adjoint problem is solved in $t \in [500, 530]$ during which the turbulent flow is fully developed. Figure 5.34(a) shows the magnitude of the l_2-norm of the adjoint velocity obtained using full-order primal solutions. It exhibits reasonable agreement with the reference slope from [39], as shown in Figure 5.34(a).

The iso-surface of adjoint velocity is shown in Figure 5.30, colored by spanwise adjoint vorticity ($\omega_z^{\text{adjoint}}$) with blue and red representing $\omega_z^{\text{adjoint}} < 0$ and $\omega_z^{\text{adjoint}} > 0$, re-

spectively. Darker colors denote larger vorticity values. Two-dimensional coherent struc-

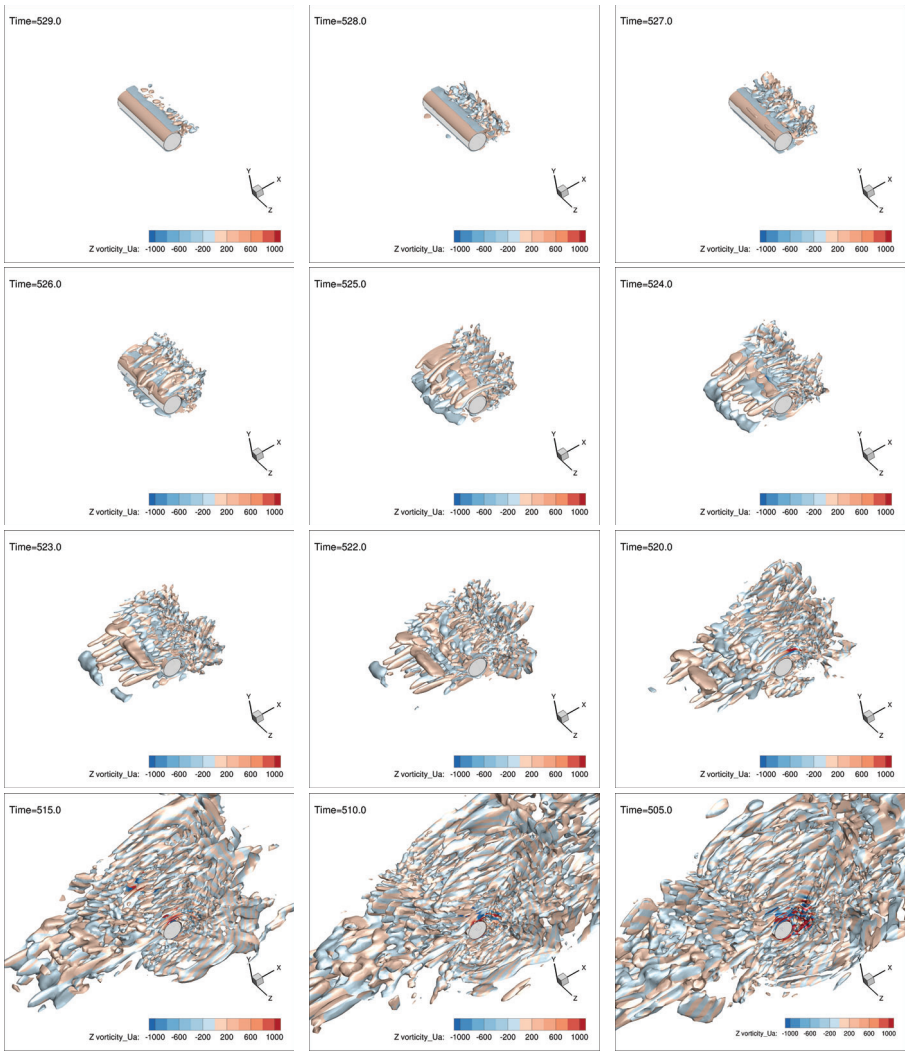


Figure 5.30: Iso-surfaces of $Q = 1$ for the adjoint velocity in a 3D cylinder flow at $Re = 500$, colored by spanwise adjoint vorticity ($\omega_z^{\text{adjoint}}$) using blue representing $\omega_z^{\text{adjoint}} < 0$ and red representing $\omega_z^{\text{adjoint}} > 0$. The darker of color, the larger of vorticity value.

tures are observed during $t = 529 - 527$. They are advected upstream, leading to streamline coherent structures. As the adjoint field is further developed, These structures become more abundant in regions away from the cylinder and more significant in the upstream direction. The change of the adjoint field is similar to the observation in the 2D case. It is noted that high-value adjoint vorticity occurs in the near wake regions upstream of the cylinder, which is similar to the observation in Ref. [39]. In fact, this is

where the flow shear stress rate changes significantly and the adjoint field originates.

5.4.4. INFLUENCE OF RORS ON THE ADJOINT MAGNITUDE

The full-order primal solution is substituted by the POD-based RORs. Now we consider ROR-driven adjoint solutions for which we compare the results to the computation using full-order solutions, referred to as the baseline. Figure 5.31 shows the eigenvalues and associated cumulative energy for the ROR cases. The truncation number is set to 30 for the enhanced online algorithm in all three problems. This captures at least 95% of the total energy. We study the l_2-norm of the adjoint velocity magnitude since it represents the energy of the adjoint system. The impacts of the ROR are investigated on the adjoint l_2-norm when the adjoint problem is solved backward in time.

THE ONLINE POD ANALYSIS FOR THREE CASES

The POD analysis was completed using the enhanced online algorithm. The eigenvalues of the three cases are shown in Figure 5.31. For the 2D cylinder flow at $Re = 100$, the eigenvalues often appear in pairs in terms of their contributions to the total energy. The first and second POD modes account for 96.4% of the total energy. The cumulative energy approaches 100% quickly as the number of POD modes is increased. Using the first 6 POD modes can recover more than 99.9% of the total energy, indicating a good approximation for the primal flow problem. When the Reynolds number is increased to 500, there exist two dominant POD modes which can capture 93.6% of the total energy. However, it requires 10 POD modes to recover 99.9% of the total energy as there are more dynamic features than the low-Re cylinder flow.

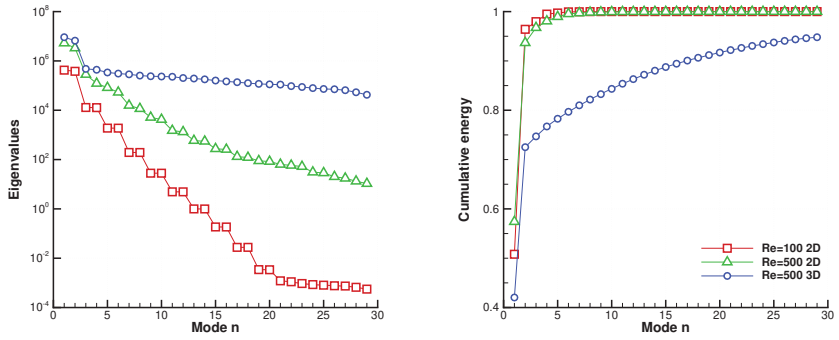


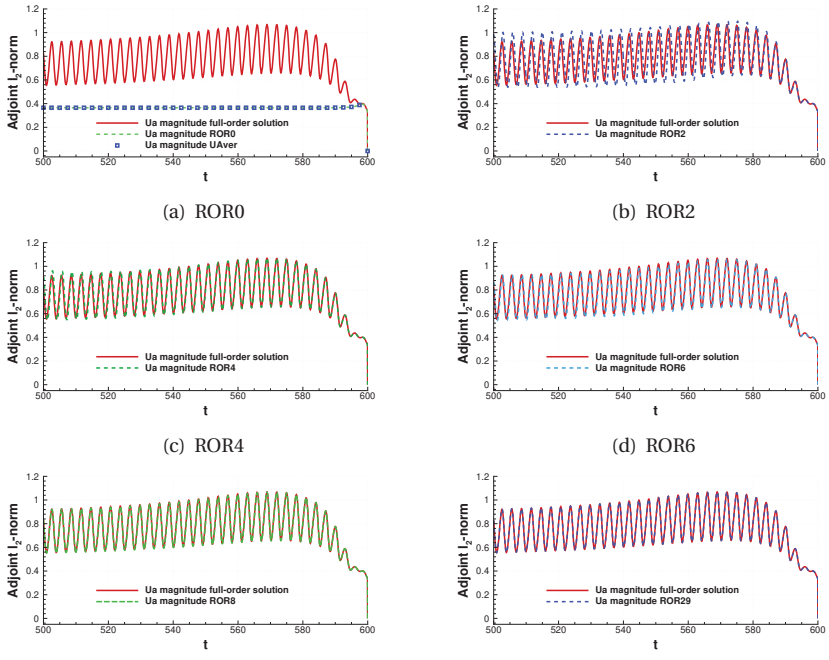
Figure 5.31: Eigenvalues (left) and cumulative energy (right) for the 2D flow past past a 2D cylinder at $Re = 100, 500$ and the 3D cylinder turbulent flow at $Re = 500$.

For the 3D cylinder flow at $Re = 500$, the first and second POD modes play an important role in capturing the total energy while the later decay of eigenvalues becomes much more smooth than the 2D flow problem. The pair of dominant POD modes results from the periodic vortex shedding which is the main feature of this flow problem. Other high-order POD modes contribute to small-scale flow structures in the 3D domain. As

the three-dimensional flow shows chaotic features, the difference between these high-order POD modes is reduced compared to the 2D case and thus the convergence of cumulative energy becomes slow, with 29 POD modes capturing 94.8% of the total energy. Increasing the number of POD modes would enable us to recover more turbulent flow dynamics. However, the computational cost increases significantly. Therefore, the truncation number for the 3D turbulent flow is fixed at 30.

2D CYLINDER FLOW AT $Re = 100$

Figure 5.32 shows the l_2-norm of the magnitude of adjoint velocity using different RORs for the 2D cylinder flow at $Re = 100$. Using only the time-averaged primal velocity gives us a stationary adjoint solution. In this case, only the starting phase follows the same route as the baseline case. It is worth mentioning that this steady state differs from the time-averaged value of the unsteady adjoint solution. As the POD analysis is applied to the primal velocity rather than the velocity fluctuation, we refer to the most significant



(a) ROR0 (b) ROR2

(c) ROR4 (d) ROR6

(e) ROR8 (f) ROR29

Figure 5.32: Comparison of the adjoint l_2-norm when the adjoint problem is solved using different reduced-order representations (RORs), which is computed using mean-flow mode and 2, 4, 6, 8, and 29 POD modes for the flow past a cylinder at $Re = 100$.

mode as the mean-flow mode. The remaining POD modes are referred to as the first mode, second mode, and so on. The temporal coefficient scales the reduced-order solution to the amplitude of the mean-flow mode during the reconstruction of the primal flow solutions. Actually, using the mean-flow mode to reconstruct the primal solution

produces similar evolution of adjoint l_2-norm to the one obtained from the mean primal solution.

When we add the first and second POD modes, the phase II and phase III are well captured although the oscillatory amplitude of the adjoint field is overestimated in both peak and trough values. Adding high-order POD modes can reduce the discrepancy. When 6 or 8 POD modes are used to reconstruct the primal solution, the difference in the l_2-norm of the adjoint solution becomes invisible. When we consider a large number of POD modes, i.e. 29 here, the l_2-norm of adjoint velocity has an excellent agreement with that obtained using full-order solutions.

2D CYLINDER FLOW AT $Re = 500$

Figure 5.33 shows the impact of using different RORs on the l_2-norm of the adjoint velocity. As before, using only the mean-flow mode will produce a steady adjoint solution. When we include the first and second POD modes (the dominant ones), the magnitude of the adjoint solution shows shedding oscillations but grows faster and higher than the baseline. Adding the 3rd and 4th POD modes regularizes the adjoint problem for a long time whilst the 5th and 6th POD modes seem to stimulate the instability of the adjoint problem. However, the adjoint field becomes more stable for cases with more than 8 POD modes. The case with 10 POD modes is less accurate than the case with 8 POD modes, but it does limit the overestimation during $t = 500 - 520$. Using 12 POD modes provides a reasonable agreement with the baseline. Figure 5.33(h) compares the l_2-norm of the adjoint velocity using 29 POD modes (ROR29) with the baseline case. The dynamics of adjoint magnitude are well captured using the ROR with 29 POD modes. It is observed that the POD modes play different roles in solving the adjoint problem. The low-order POD modes are important for capturing the kinetic energy but do not necessarily stabilize the adjoint problem.

3D CYLINDER FLOW AT $Re = 500$

Different RORs are now employed to solve the adjoint problem for the 3D turbulent flow. As shown in Figure 5.34(a), using the solution from the mean-flow mode induces a steady adjoint solution while adding more POD modes produces dynamic characteristics. The unbounded growth of adjoint solutions is suppressed using RORs. In this case, increasing the number of POD modes improves the prediction accuracy in early time states ($530 - 516$) as shown in Figure 5.34(h). This can be observed at $t = 524$, 522 and 518. Meanwhile, using a higher-order ROR leads to irregular convergence of the adjoint field during $t = 516 - 505$. In particular, the adjoint magnitude using ROR10 is closer to that from full-order solutions than that from ROR20, as shown in Figure 5.34(g). This could be because the POD modes are responsible for improving the accuracy of adjoint solutions for a certain time, either at the early or later time states, rather than the whole computational time interval when the adjoint problem is solved. Adding more POD modes will make the adjoint field more similar to the one based on the full-order flow solution for later time states, but this is left for future study. Nonetheless, using a truncation number of 30 still recover the large-scale coherent structures, as shown in Figure 5.35(e) and Figure 5.35(f).

Figure 5.35 shows the iso-surface of $Q = 0.1$ for reduced- and full-order *primal* solutions, colored by the spanwise vorticity (ω_z). Using just the mean-flow mode (ROR0)

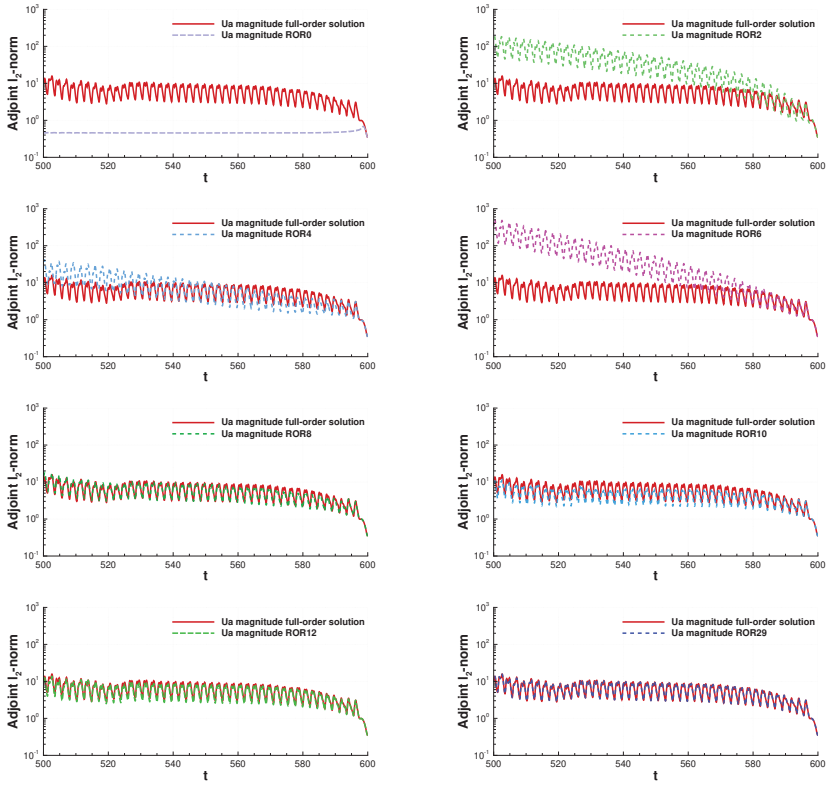


Figure 5.33: Comparison of adjoint l_2-norm when the adjoint problem is solved using different reduced-order representations (RORs), which is computed using mean-flow mode and 2, 4, 6, 8, 10, 12 and 29 POD modes for the flow past a 2D cylinder at $Re = 500$.

results in a simple counter-rotating vortex pair. The vortex-shedding dynamic feature appears when adding the first and second POD modes. Adding more POD modes leads to more small-scale *primal* structures. Using 29 POD modes induces iso-surfaces resembling those obtained from the full-order solutions.

The iso-surfaces of $Q = 1$ for the *adjoint* velocity using various RORs are compared to that from the full-order solution in Figure 5.36, which is colored with the value of the spanwise adjoint vorticity ($\omega_z^{\text{adjoint}}$). The adjoint field obtained using the mean-flow mode (ROR0) exhibits steady structures around the cylinder. Large-scale structures in the adjoint field are captured as we add POD modes, even just with the first and second POD modes. Increasing the number of POD modes also enables the production of adjoint structures in upstream and near-wake regions. Large-scale *adjoint* coherent structures appear in upstream regions for ROR2 and ROR4. Smaller structures appear when using more high-order POD modes. Using 29 POD modes produces coherent structures with a remarkable resemblance to those obtained using the full-order solutions.

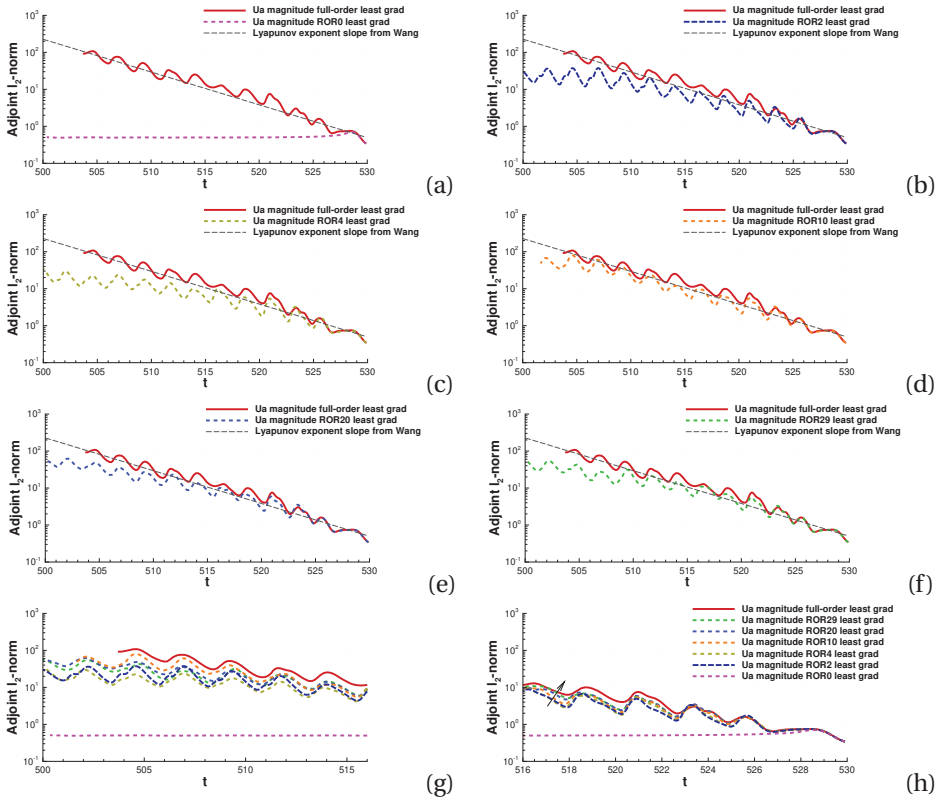


Figure 5.34: Comparison of the l_2-norm of the adjoint velocity solved by different reduced-order representations (RORs) with that from full-order primal solutions. RORs are computed using the mean-flow mode and 2, 4, 10, 20 and 29 POD modes for the turbulent flow past a 3D cylinder at $Re = 500$. The grey long-dashed line denotes the reference slope of the growth from Ref. [39]. These results are compared in (g) early time states (530 – 516) and (h) later states (516 – 505).

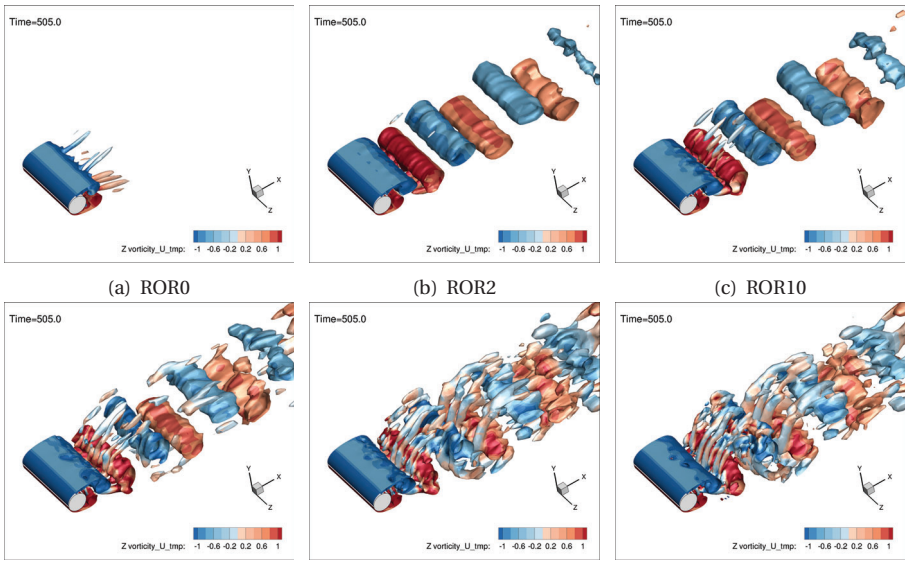
5.5. ACCURACY AND EFFICIENCY

In order to quantitatively evaluate the impacts of RORs, we employ the l_2-norm of the error when comparing RORs with the full-order primal solution, viz.

$$\varepsilon_{\text{Primal}} = \sqrt{\frac{1}{N_{\text{cells}}} \|\boldsymbol{u}_{\text{ROR}} - \boldsymbol{u}\|^2}, \tag{5.148}$$

where N_{cells} denotes the number of mesh cells. $\boldsymbol{u}_{\text{ROR}}$ and $\boldsymbol{u}$ denote the reduced- and full-order solutions respectively. This is referred to as the primal error hereafter and represents the spatially-averaged error. Likewise, the adjoint error based on RORs is calculated with respect to the adjoint solution ($\boldsymbol{u}_a$) obtained using the full-order primal solution,

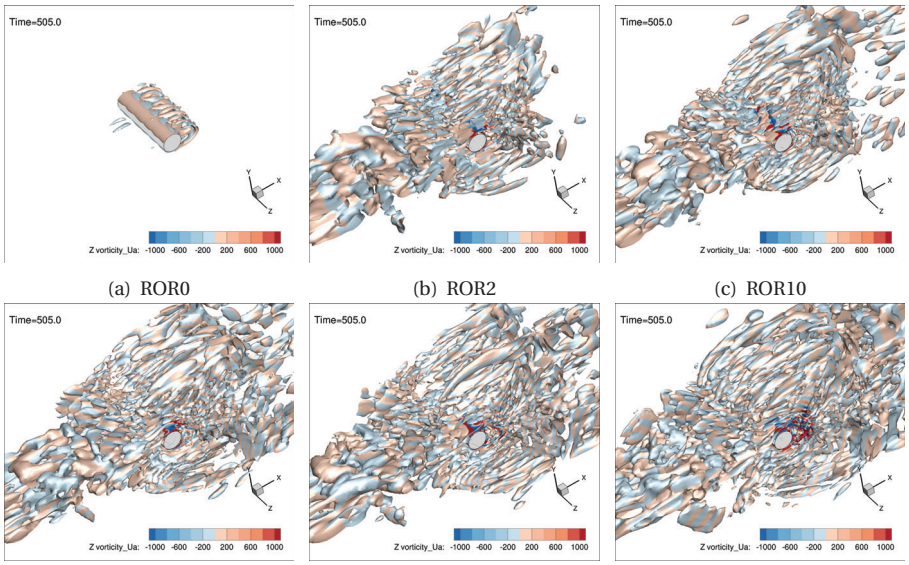
$$\varepsilon_{\text{Adjoint}} = \sqrt{\frac{1}{N_{\text{cells}}} \|\boldsymbol{u}_{a,\text{ROR}} - \boldsymbol{u}_a\|^2}. \tag{5.149}$$



(a) ROR0 (b) ROR2 (c) ROR10

(d) ROR20 (e) ROR29 (f) Full-order solutions

Figure 5.35: Iso-surfaces of $Q = 0.1$ for the reduced-order and full-order *primal* solutions in the 3D cylinder flow at $Re = 500$, colored by vorticity (ω_z) using blue representing $\omega_z < 0$ and red representing $\omega_z > 0$. The darker of color, the larger of vorticity values.



(a) ROR0 (b) ROR2 (c) ROR10

(d) ROR20 (e) ROR29 (f) Full-order solutions

Figure 5.36: Iso-surfaces of $Q = 1$ for the adjoint velocity based on the reduced-order and full-order primal solutions in the 3D cylinder flow at $Re = 500$, colored by the adjoint vorticity ($\omega_z^{\text{adjoint}}$) using blue representing $\omega_z^{\text{adjoint}} < 0$ and red representing $\omega_z^{\text{adjoint}} > 0$. The darker of color, the larger of vorticity values.

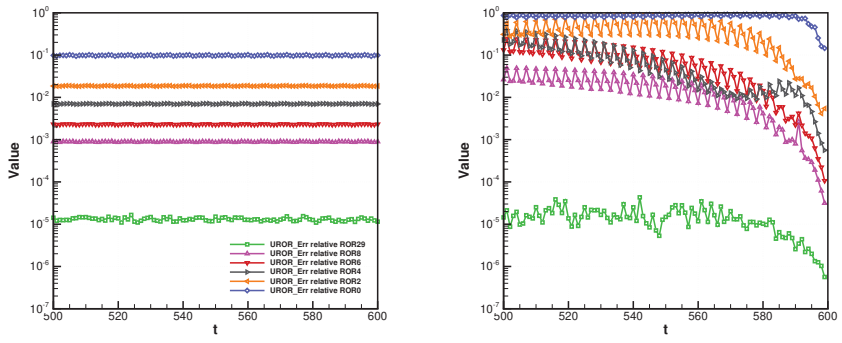
$\boldsymbol{u}_{a,\mathrm{ROR}}$ denotes the adjoint solution obtained using a ROR. $\varepsilon_{\mathrm{Primal}}$ and $\varepsilon_{\mathrm{Adjoint}}$ describe the instantaneous but spatial-averaged error for each snapshot. To compare with the solution fields, the primal or adjoint error is normalized by the l_2-norm of the instantaneous primal or adjoint solution magnitude at the corresponding time, leading to the relative error. To evaluate performance over a computational time interval, a mean error is defined by averaging the instantaneous errors in time as

$$\varepsilon_{\mathrm{mean}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \varepsilon_{\mathrm{Sol}}^{i}, \tag{5.150}$$

where $\varepsilon_{\mathrm{Sol}}^{i}$ can be $\varepsilon_{\mathrm{Primal}}$ for the primal solutions or $\varepsilon_{\mathrm{Adjoint}}$ for the adjoint solutions at the i-th snapshot and N_t denotes the number of snapshots.

5.5.1. 2D CYLINDER FLOW AT $Re = 100$

Figure 5.37(a) shows the evolution of relative primal errors $\varepsilon_{\mathrm{Primal}}$ when we use the mean-flow mode and 2, 4, 6, 8, and 29 POD modes to reconstruct the ROR. Although using just the mean-flow mode is a naive approximation for this case, doing so produces primal errors lower than 0.1 during the computational time interval. The relative primal error is around 10% as shown in the Figure 5.37(a), indicating that the mean value accounts for a significant contribution to the primal solution. Increasing the number of POD modes can significantly reduce the remaining primal error. The ROR with 8 POD modes produces velocities with a difference of less than 0.001. The reconstruction using 29 POD modes can achieve a lower error of 10^{-5}. It is noted that the error remains relatively constant during the computing time interval. The relative adjoint error is higher and less regular than the primal error.



(a) Relative primal error (b) Relative adjoint error

Figure 5.37: Error evolutions of the reduced-order primal solutions, associated adjoint velocity and adjoint pressure for the flow past a 2D cylinder at $Re = 100$ when we use mean-flow mode, and 2, 4, 6, 8 and 29 POD modes. Absolute and relative errors are shown in the top and bottom rows, respectively.

Figure 5.37(b) shows the associated adjoint errors. If we only use the mean value of the primal velocity to solve the adjoint, the resulting adjoint error is an order of magnitude greater than the primal error. This is consistent with the discrepancy shown in

Figure 5.32(a). The adjoint error using 4 POD modes is less regular. Although using 4 POD modes improves the accuracy during the phase II of the adjoint field, the adjoint error grows up to large values in the statistical-steady state. This is because using 4 POD modes can stabilize the development of the adjoint field but the adjoint solution is overestimated during the settle-down phase afterwards, in which the small-scale primal solution is of fundamental importance for accurately approximating the adjoint solution. For all RORs, the magnitude of the adjoint error is roughly one order higher than the associated primal error during phase III. However, a high-accurate ROR primal solution reconstructed from 29 POD modes, for instance, can generate the adjoint field within the same order of magnitude despite large oscillations. Note that the error for the adjoint pressure behaves in a manner similar to that of the adjoint velocity.

To illustrate the impacts of RORs in space, Figure 5.38 shows the magnitude of instantaneous error of the reduced-order primal velocity, and associated adjoint velocity and pressure at $t = 500$ when we use different RORs. It is noted that using the mean values (ROR0) will lead to large errors in wake regions since the vortex-shedding structure is neglected. There is a significant difference in the adjoint solutions in the upstream jet region. This defect is reduced when we consider the first and second POD modes to reconstruct the primal solution. Neglecting high-order POD modes has an impact on the computation of small scales in the vortex street. There is only a tiny region with light error streaks for the adjoint velocity and pressure when we use 8 POD modes for building the ROR. This effect disappears with considering a more accurate ROR with 29 POD modes.

The efficiency metric (see the definition in Equation (4.17)) indicates the memory saved by the ROR compared to the full-order solution. The larger value of the efficiency metric, the less memory cost of the ROR. Figure 5.39 shows this efficiency metric for the 2D cylinder flow at $Re = 100$ in the time interval of $t = 500 - 600$ and the corresponding mean primal and adjoint errors (Equation (5.150)) using various RORs. The ROR with 8 POD modes produces a good accuracy of the adjoint solution (less than 0.01 error) while only requiring 0.24% of the memory. A lower error level (roughly 10^{-5}) can be achieved with a memory cost of less than 1%, as shown for ROR29.

5.5.2. 2D CYLINDER FLOW AT $Re = 500$

Figure 5.40 shows the relative primal error and associated adjoint error when we solve the adjoint problem backward in time with different RORs. The primal error is reduced by increasing the number of POD modes, as expected. However, there is a complex impact on the adjoint field. Using the mean primal solution (i.e. ROR0) induces an adjoint error with an order of magnitude similar to that of the adjoint solution and this error is kept at a constant level. The adjoint error that of using low-order RORs (e.g. ROR2, ROR4, ROR6) grows and becomes larger than that of the steady adjoint, resulting in the divergence of the adjoint field at large times. However, once more than 8 POD modes are added, stability is regained. Adding high-order POD modes can generate the adjoint solution with low errors in the current study. Using ROR29 to solve the adjoint problem gives a computation with a relative error in the order of magnitude at 0.01.

Figure 5.41(a) shows the mean error and the efficiency metric for the cases using various RORs. The reconstruction accuracy of the primal velocity is constantly improved as

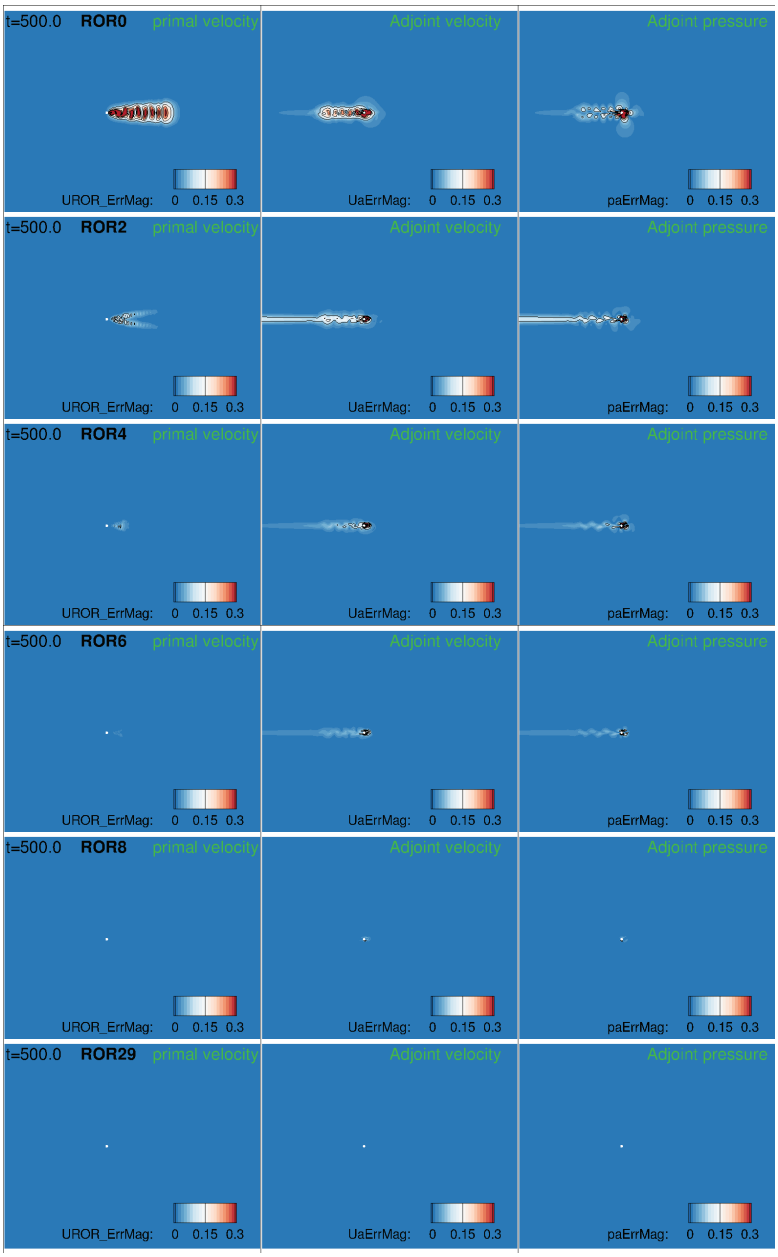


Figure 5.38: Instantaneous error magnitude of reduced-order primal velocity (left), and associated adjoint velocity (middle) and pressure (right) at $t = 500$ for the flow past a 2D cylinder at $Re = 100$. Different rows from top to bottom denote results using the mean-flow mode, 2, 4, 6, 8 and 29 POD modes.

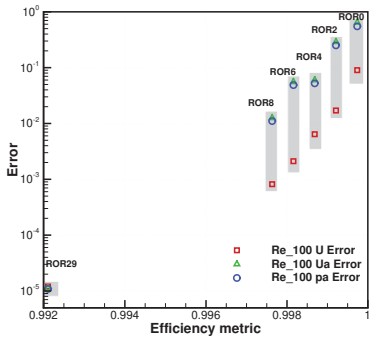
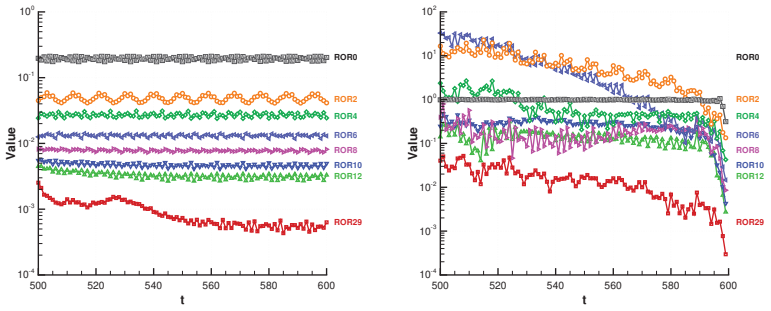


Figure 5.39: Efficiency metric and associated mean errors of primal velocity, adjoint velocity and pressure using different RORs for the 2D cylinder flow at $Re = 100$.



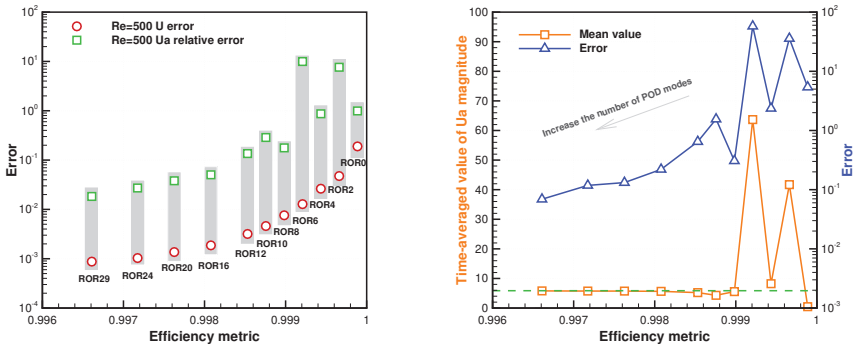
(a) Primal error (b) Adjoint error

Figure 5.40: Relative errors of the reduced-order primal velocity, associated adjoint velocity using different RORs for the flow past a 2D cylinder at $Re = 500$.

more POD modes are used. The mean adjoint error is reduced with a less regular pattern. However, using ROR6 or ROR2 leads to a high error. This is because low-order RORs can cause the divergence of the adjoint field. It is observed that the convergence for this case is slower than the low-Re cylinder flow since the flow field becomes more complex and dynamic. Figure 5.41(b) shows the mean value of the l_2-norm of the adjoint velocity for cases using different RORs. The RORs with low numbers of POD modes (ROR2, ROR6) produce a large discrepancy. However, the mean value of adjoint velocity is well resolved as the number of POD modes is increased, with errors decreasing (though not monotonically).

5.5.3. 3D CYLINDER FLOW AT $Re = 500$

Figure 5.42 shows the impact of using RORs on the accuracy and efficiency of the 3D cylinder turbulent flow. As for the 2D cases, increasing the number of POD modes reduces the mean error of both primal and adjoint solutions, where the primal error is



(a) Overall primal and adjoint error (b) Mean value of adjoint magnitude

Figure 5.41: The time-averaged error of primal and adjoint velocity using different RORs and the mean value of the associated magnitude of adjoint velocity for the flow past a 2D cylinder at $Re = 500$. The green dashed lines denote the reference values from full-order primal solutions.

lower. Accurate computation of the adjoint field requires a relatively high number of POD modes for the 3D turbulent flow. It is also observed that the convergence of the 3D problem is slower than that of the 2D case. This is because using 29 POD modes captures 94.804% of kinetic energy in the 3D simulation, instead of the 99.9976% captured by 29 modes in the 2D case. However, using a ROR still significantly reduces the memory cost for the 3D turbulent flow simulation, with 29 POD modes using only 5% of the memory of the full-order solution. Meanwhile, the reconstruction cost of reduced-order solutions is trivial compared to the cost of solving the adjoint problem. Figure 5.43 shows the error evolution of the primal and adjoint velocity with different RORs when the adjoint problem is solved backward in time. When increasing the number of POD modes, the improvement of the adjoint velocity is observed in early time states (i.e. $t = 530-516$ in region 1), while it becomes less regular in region 2 for later time states ($t = 516-505$).

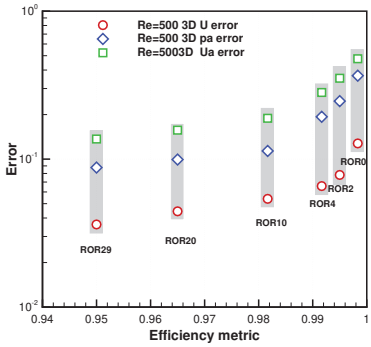
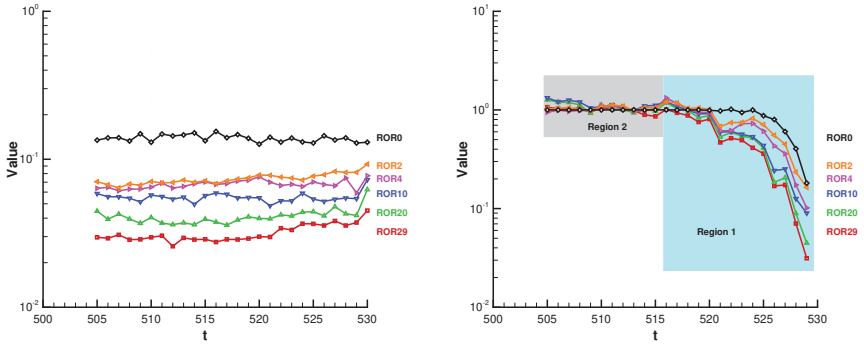


Figure 5.42: Time-averaged error of primal and adjoint velocity over $t \in [525, 530]$ using different RORs for the flow past a 3D cylinder at $Re = 500$.

This is similar to the impact of RORs on the adjoint field presented in Figures 5.34(g) and 5.34(h). In other words, different POD modes have an influence on the solution accuracy in different time intervals. The large errors in region 2 result from a constrained growth of the adjoint field, which produces the adjoint solution with lower magnitudes. However, using RORs with a sufficient number of modes gives a reasonable prediction of the adjoint field, at least for the length of time considered here.



(a) Primal error (b) Adjoint error

Figure 5.43: Error evolution for the primal and adjoint velocity over $t \in [505, 530]$ using different RORs for the turbulent flow past a 3D cylinder at $Re = 500$.

5.6. EFFECT OF RORS ON ERROR ESTIMATION

We now consider the effect of using a ROR on the quality of adjoint-based error estimation. To do so we apply the discrete adjoint method to the 2D and 3D cylinder flow at $Re = 500$ described above. In FVM, the discrete adjoint method has been widely used for steady problems [162]. By virtue of applying this technique on each time step, we can use discrete adjoint solutions and the flow residual to provide the error estimation for unsteady flow problems. At each time step, we have the algebraic system of the discrete solution vector $\hat{\boldsymbol{u}} = [\boldsymbol{u} \quad p]^\intercal$, the solutions on a coarse mesh, as

$$\mathbf{A}\hat{\boldsymbol{u}} = \mathbf{F}, \tag{5.151}$$

where $\boldsymbol{F}$ is the right-hand vector and $\boldsymbol{A}$ denotes the left-hand matrix. The residual is defined in this space as

$$\mathbf{R}(\hat{\boldsymbol{u}}) \equiv \boldsymbol{A}\hat{\boldsymbol{u}} - \mathbf{F} = \mathbf{0}, \tag{5.152}$$

where $\mathbf{R}(\cdot)$ is the residual operator. The error estimation of a QoI [163] can be approximated by

$$\delta J = J_h(\hat{\boldsymbol{u}}_h) - J_H(\hat{\boldsymbol{u}}_H) \approx -\hat{\boldsymbol{u}}_{a,h}^T \mathbf{R}_h(\hat{\boldsymbol{u}}_h^H). \tag{5.153}$$

where $\hat{\boldsymbol{u}}_{a,h}$ and $\hat{\boldsymbol{u}}_h$ denotes the adjoint and flow solutions a fine mesh. The subscript h of $\mathbf{R}$ means the residual is evaluated on the fine space. $\hat{\boldsymbol{u}}_h^H$ denotes the flow solution

interpolated from a coarse mesh ($\hat{\boldsymbol{u}}_H$) to a fine mesh,

$$\hat{\boldsymbol{u}}_h^H = \boldsymbol{I}_h^H \hat{\boldsymbol{u}}_H, \tag{5.154}$$

where $\mathbf{I}_h^H$ is a lossless interpolation operator. For time-dependent problems, the error estimation can be expressed as

$$\Delta J = \frac{1}{T}\int_I \delta J \,\mathrm{d}t \approx \frac{1}{N_t}\sum_{n=1}^{N_t} \delta J = -\frac{1}{N_t}\sum_{i=1}^{N_t} \hat{\boldsymbol{u}}_{a,h}^T \mathbf{R}_h(\hat{\boldsymbol{u}}_h^H) = \frac{1}{N_t}\sum_{n=1}^{N_t} \Delta J^n, \tag{5.155}$$

where ΔJ^n stands for the error estimation on each time step. It is expanded in space as

$$\Delta J^n = -\hat{\boldsymbol{u}}_{a,h}^T \mathbf{R}_h(\hat{\boldsymbol{u}}_h^H) = -\left[\boldsymbol{u}_{a,h}^T \mathbf{R}_h^u(\boldsymbol{u}_h^H, p_h^H) + p_{a,h}^T \mathbf{R}_h^p(\boldsymbol{u}_h^H)\right] = \sum_{i=1}^{N_{\text{cell}}} \Delta J_i^n, \tag{5.156}$$

where ΔJ_i^n denotes the error estimation on the i-th mesh cell with an expression of

$$\Delta J_i^n = -\left[\boldsymbol{u}_{a,h_i}^T \mathbf{R}_h^u(\boldsymbol{u}_{h_i}^H, p_{h_i}^H) + q_{a,h_i}^T \mathbf{R}_h^p(\boldsymbol{u}_{h_i}^H)\right]. \tag{5.157}$$

This can be achieved by Algorithm 6. The error estimation is referred to as the QoI error estimator on each cell. The absolute value of the QoI error estimator is also considered for validating the impacts of RORs on error estimation because this is often used as a mesh adaptation criterion. $\hat{\boldsymbol{u}}_h$ and $\hat{\boldsymbol{u}}_{a,h}$ are interpolated from $\boldsymbol{u}_{\text{ROR}}$ and $\boldsymbol{u}_{a,\text{ROR}}$ on a coarse mesh when a ROR is used for evaluating the error estimation.

Algorithm 6 Procedure for the adjoint-based error estimation in 2D/3D problems

1. Generate a fine mesh ($\mathcal{G}_h$) by uniformly refining the coarse mesh ($\mathcal{G}_H$).
2. Interpolate the primal and solution onto the fine mesh, i.e. $\boldsymbol{u}_h^H, p_h^H, \boldsymbol{u}_{a,h}^H, p_{a,h}^H$.
3. Compute the flow residual on the fine mesh, viz. $\boldsymbol{R}_h^u, \boldsymbol{R}_h^q$.
4. Evaluate the error estimation for all mesh cells.

5.6.1. 2D CYLINDER FLOW AT $Re = 500$

Figure 5.44 shows the distribution of instantaneous (ΔJ_i^n) and time-averaged QoI error estimators (ΔJ_i) using full-order solutions and RORs. The instantaneous and time-averaged error estimators obtained with full-order solutions are shown in Figure 5.44(a) and Figure 5.44(b). The instantaneous estimators show large-error regions in the vortex street. This means that the mesh cells in this region should be refined to improve the computational accuracy of drag. In contrast, the time-averaged estimators show a symmetric distribution with three significant regions: the near-wake region, the intermediate-wake region, and the far-wake region. The near-wake region exists due to the high local flow gradients. The high-error regions in the intermediate and far wake have peak-to-peak wavelengths in [1.2, 1.6] (the black dashed lines). The time-averaged convection

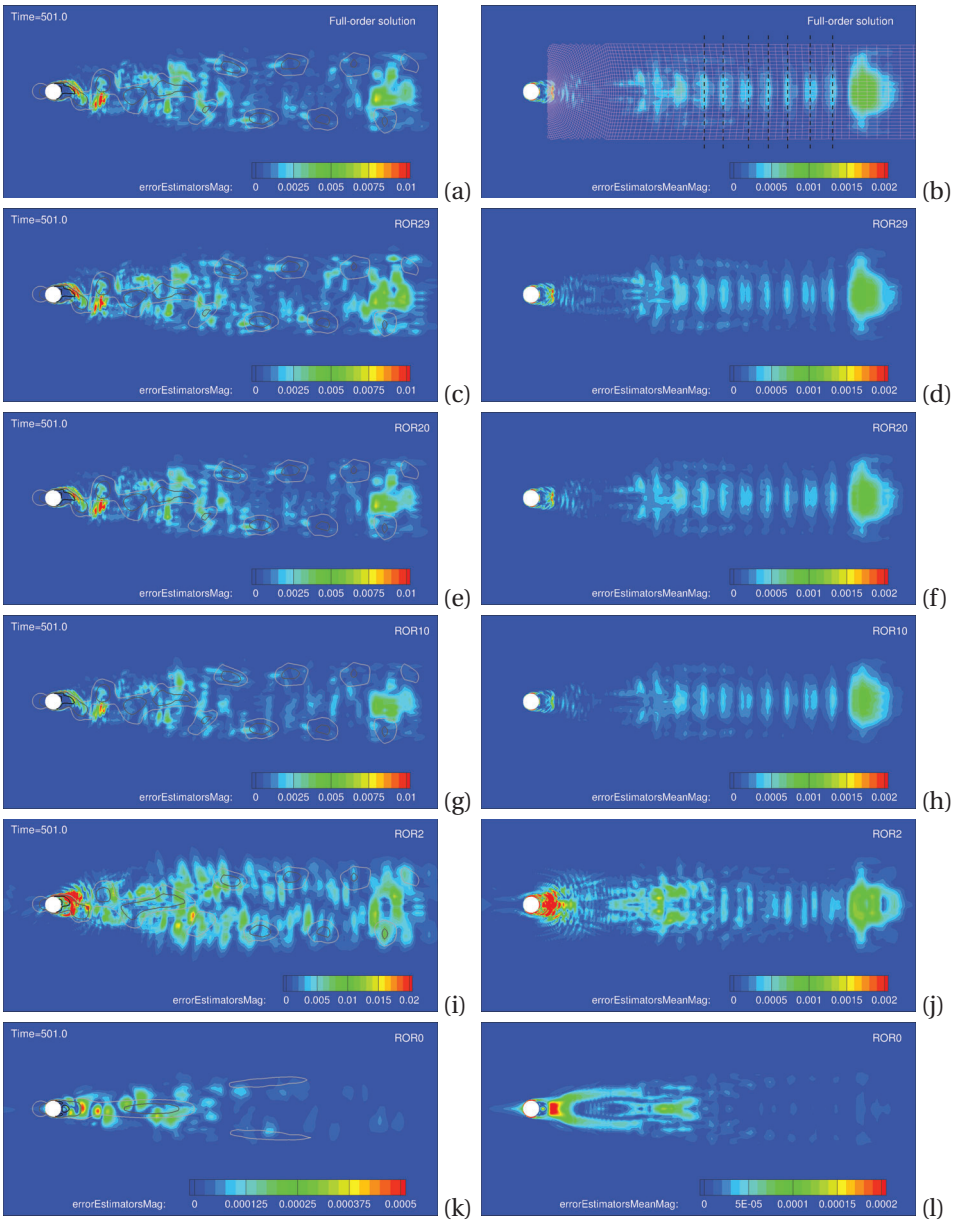


Figure 5.44: Instantaneous (left) and time-averaged (right) QoI error estimators computed using full-order flow solutions and different reduced-order representations (RORs), which is computed using 29, 20, 10, 2 POD modes and mean-flow mode for the flow past a 2D cylinder at $Re = 500$. Solid lines denote the streamwise flow velocity $u = 0$ (black), 0.6 (gray) and 0.8 (white).

speed in the streamwise region is 1.1m/s, and thus the characteristic time for these regions is in [1.1, 1.45]. The results shown are for snapshots sampled at $\Delta t_s = 1$, but increasing the frequency of snapshots by twice or four times has no influence on the distributions in these regions. Figure 5.45 shows the instantaneous residuals of continuity and momentum at $t = 501$. These have a consistent pattern, particularly for the momentum residual which has a large blotch. The coarse mesh could result in these high residuals in the intermediate and far wake regions. The mesh is too coarse to resolve the shedding wavelength, especially in the large blotch, thus initiating the high residual values. Alternatively, the interpolation scheme used to obtain the solutions on a fine mesh introduces high error in these regions and has a significant influence on the residual accuracy.

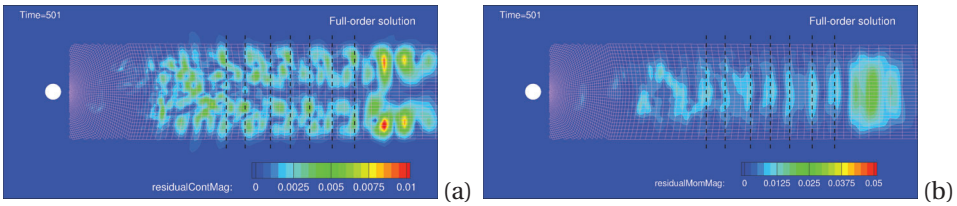


Figure 5.45: Instantaneous residuals for the (a) continuity and (b) momentum for the interpolated solutions on a fine mesh at $t = 501$ for the flow past a 2D cylinder at $Re = 500$.

When we only use the mean value of the primal solution, the instantaneous estimators (Figure 5.44(k)) follow the vortex shedding pattern whilst their values are lower than that from the full-order solutions. This is because a steady adjoint field is obtained under this scenario. The time-averaged estimators (see Figure 5.44(l)) show a symmetric pattern in near-wake regions, but it differs from the one based on the full-order solutions. In that sense, the time-averaged error estimators using the ROR0 present the characteristics of the mean-flow field while ignoring the transient dynamics. The time-averaged estimators could lead to a mesh suitable for steady simulations but not sufficient for unsteady simulations. Adding first and second POD modes (Figure 5.44(i) and Figure 5.44(j)) produces estimator distributions similar to the ones based on the full-order solutions for the instantaneous and time-averaged values, albeit with larger magnitudes. The three significant regions in the wake are observed clearly in this case. When the number of POD modes is further increased, the distribution of error estimators increasingly resembles the full-order case, with the difference becoming trivial when using a ROR with 20 POD modes.

To evaluate the influence on QoI error estimators, the difference of time-averaged error estimators is defined by

$$\epsilon_{\Delta J} = \frac{1}{N_{\text{cells}}} \sum_{i=1}^{N_{\text{cells}}} \left\| \Delta J_i^{\text{ROR}} - \Delta J_i^{\text{full-order}} \right\|, \tag{5.158}$$

where ΔJ_i^{ROR} and $\Delta J_i^{\text{full-order}}$ denote the time-averaged error estimator in the i-th mesh cell using RORs and full-order solutions. The absolute difference is given by

$$\epsilon_{\Delta J}^{\text{abs}} = \frac{1}{N_{\text{cells}}} \sum_{i=1}^{N_{\text{cells}}} \left\| \, |\Delta J_i^{\text{ROR}}| - |\Delta J_i^{\text{full-order}}| \, \right\|. \tag{5.159}$$

These two differences are shown in Figure 5.46. The error of QoI error estimators, for both the actual values and their absolutes, is reduced with an increasing number of POD modes although the convergence is not regular. The ROR with 20 POD modes shows a low difference in the current case while further increasing the number of POD modes increases the difference. This could be because the higher POD modes are related to the small-scale features, which have a complex influence on the computation of error estimators. The difference of the absolute values is lower than 10^{-3} for all cases while increasing the number of POD modes further reduces this difference with a good efficiency metric.

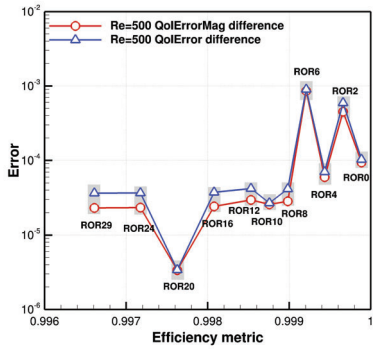


Figure 5.46: Difference between the time-averaged QoI error estimators using different RORs and that from full-order flow solutions for the flow past a 2D cylinder at $Re = 500$. The triangle-blue line denotes the difference computed based on actual values of error estimators ($\epsilon_{\Delta J}$) while the circle-red line represents that from the absolute of error estimators ($\epsilon^{\text{abs}}_{\Delta J}$).

5.6.2. 3D CYLINDER FLOW AT $Re = 500$

The QoI error estimation for the 3D turbulent flow is studied using different RORs in this section. The analyses are conducted for both short-time and long-time periods sampled by $\Delta t_s = 1$. This is because the solution characteristics in a long-time period differ from the baseline due to the reduction of the adjoint growth. This property can make the adjoint problem more tractable for turbulent flows. However, it is important to determine if the ability to estimate the QoI error is maintained for such long times, which will be addressed below.

SHORT-TIME PERIOD

A short-time period ($t \in [525, 530]$) is first considered. Figure 5.47 shows the slice of instantaneous error in x-y and x-z planes at $t = 525$. For convenience, the QoI error estimator computed from the full-order primal solution is referred to as the baseline value. The instantaneous QoI error estimators in the baseline case exhibit vortex-shedding characteristics. The spanwise distribution is nearly homogeneous in near-wake regions and regions before the cylinder. The regions with large errors localize around the shear layer (i.e. $u = 0$). Other significant regions occur along the vortex-shedding street.

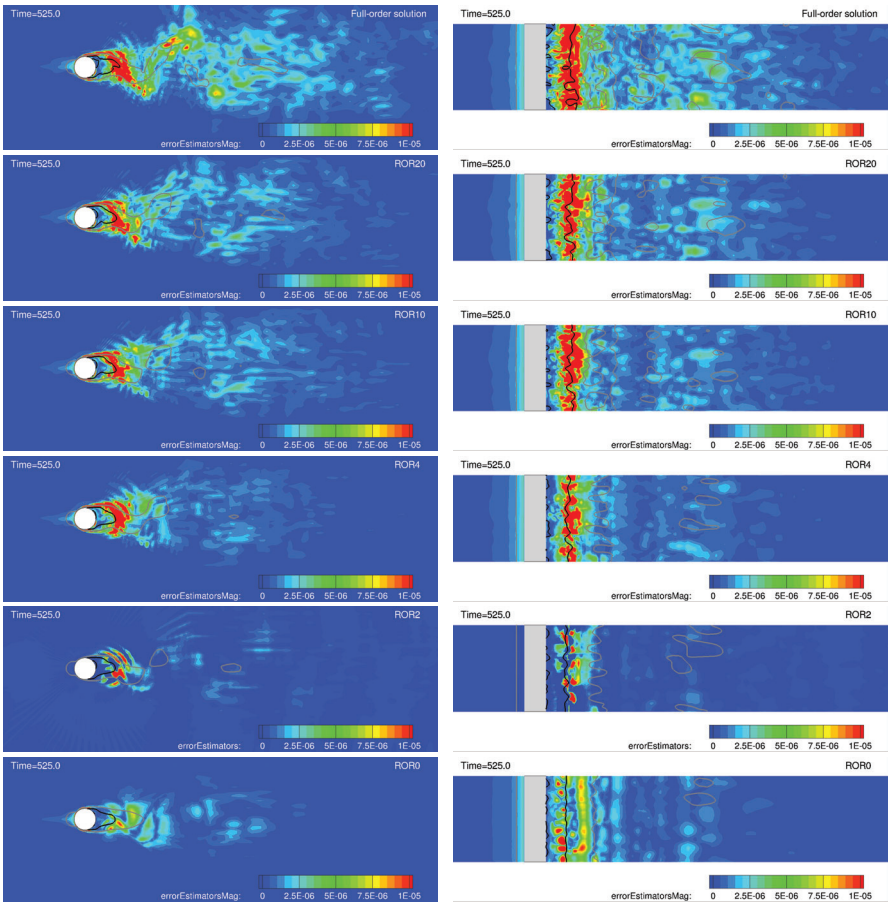


Figure 5.47: Instantaneous QoI error estimators in x-y (left, at $z = 0$) and x-z (right, at $y = 0$) planes evaluated at $t = 525$ for the flow past a 3D cylinder at $Re = 500$, based on full-order flow solutions and different reduced-order representations (RORs) that are computed using 20, 10, 4, 2 POD modes and mean-flow mode. Solid lines denote the streamwise primal velocity, $u = 0$ (black) and 0.6 (grey).

When we use the mean primal solution (ROR0) for solving the adjoint problem, the distribution of the instantaneous QoI error estimators differs from that using full-order unsteady solutions. The former has a lower magnitude with less chaotic features in the spanwise direction. Adding the first and second POD modes results in an error distribution much more similar to the baseline distribution and magnitude. The resemblance to the baseline is increased as higher-order POD modes are added to the ROR, as shown in Figure 5.47. A ROR with 20 POD modes is capable of capturing the regions with large errors both in streamwise and spanwise directions.

Time-averaged QoI error estimators for the baseline and using RORs are shown in Figure 5.48. The distribution of QoI error estimators with ROR0 misses the early shear layer and central wake regions ($x > 3L_d$) while captures the large error regions in near

wake. This reasonable prediction in the plane of $y = 0$ is because of the short time period, in which the adjoint field is less dynamic and the momentum residuals are well captured in these regions as shown in Figure 5.49. The time-averaged error estimators with a small sampling step, as shown in Figure 5.50, depicts that using ROR0 can overestimate the error in the regions upstream of the cylinder ($x < -L_d$). When a ROR with the first two

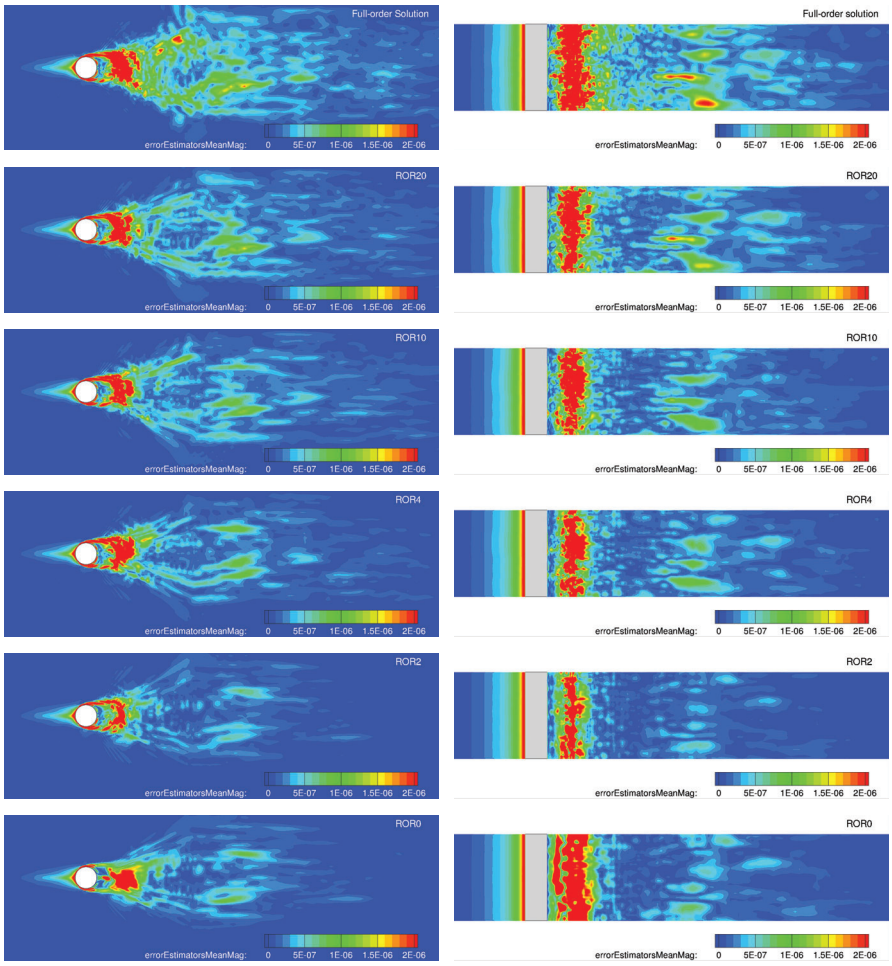


Figure 5.48: Time-averaged QoI error estimators in x-y (left, at $z = 0$) and x-z (right, at $y = 0$) planes for the flow past a 3D cylinder at $Re = 500$ over a short-time period $t \in [525, 530]$, based on full-order flow solutions and different reduced-order representations (RORs) that are computed using 20, 10, 4, 2 POD modes and mean-flow mode.

dominant POD modes, those regions are recognized well although they are narrow and sparse compared to the baseline. The structure of QoI error estimators becomes similar to the baseline as more POD modes are considered, with small-localized regions in the cylinder wake. Using ROR20 provides a much closer distribution in the x-y plane while

it is still a bit thin in the spanwise plane. Note that the QoI error estimators in regions before the cylinder shows a two-dimensional feature in the spanwise direction.

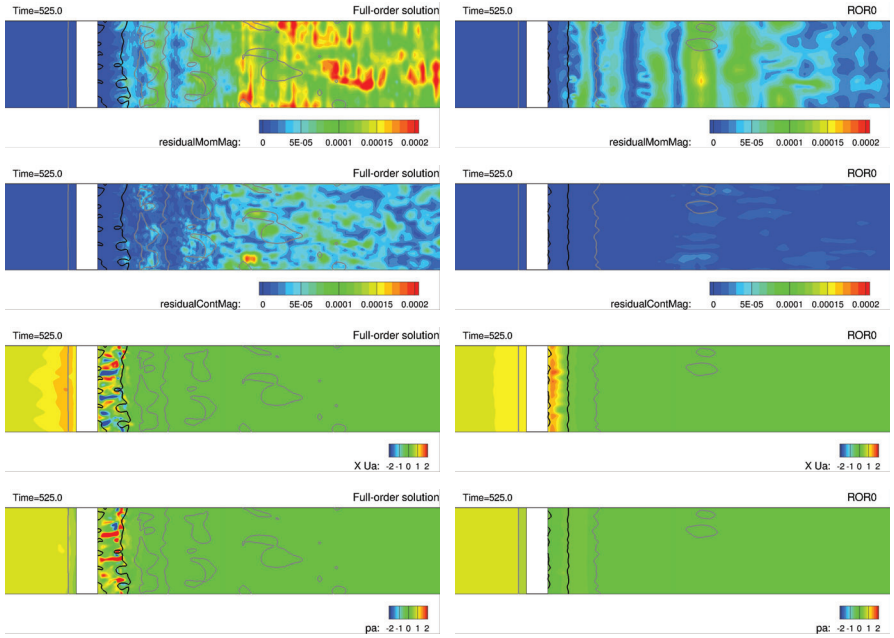


Figure 5.49: Instantaneous distribution of the magnitude of momentum residuals (first row), the magnitude of continuity residual (second row), x-component of adjoint velocity (third row), and adjoint pressure (fourth row) in x-z ($y = 0$) planes for the flow past a 3D cylinder at $Re = 500$, based on full-order flow solutions (left) and ROR0 (right) using only the mean-flow mode. Solid lines denote the streamwise primal velocity, $u = 0$ (black) and 0.6 (grey).

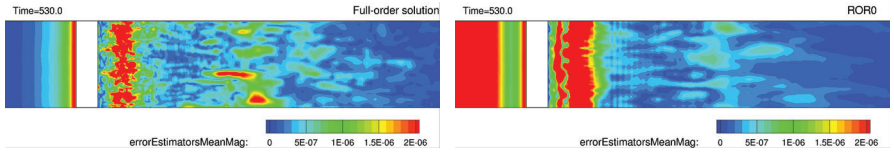


Figure 5.50: Time-averaged QoI error estimators in x-z ($y = 0$) planes for the flow past a 3D cylinder at $Re = 500$ over a short-time period $t \in [525, 530]$ with a small sampling step of $\Delta t = 0.1$, computed based on full-order flow solutions and ROR0 using only the mean-flow mode.

LONG-TIME PERIOD

Figure 5.51 shows the instantaneous QoI error estimators at $t = 505$ where we consider a long-time period ($t \in [505, 530]$) for the adjoint problem and associated error estimation. The QoI error estimator using mean-flow solutions (ROR0) is much lower than the baseline and the dominant large-error regions are not well recognized. However, adding POD

modes for the ROR improves the estimate of the instantaneous QoI error. When we use the first two dominant POD modes (ROR2), the most significant regions with high QoI error estimators are found in the near cylinder wake. Other far-wake regions, which are neglected using a ROR2, are gradually identified with increasing numbers of POD modes. The distribution of QoI error estimators with 20 POD modes shows a resemblance to the baseline for the regions with high adjoint-based errors.

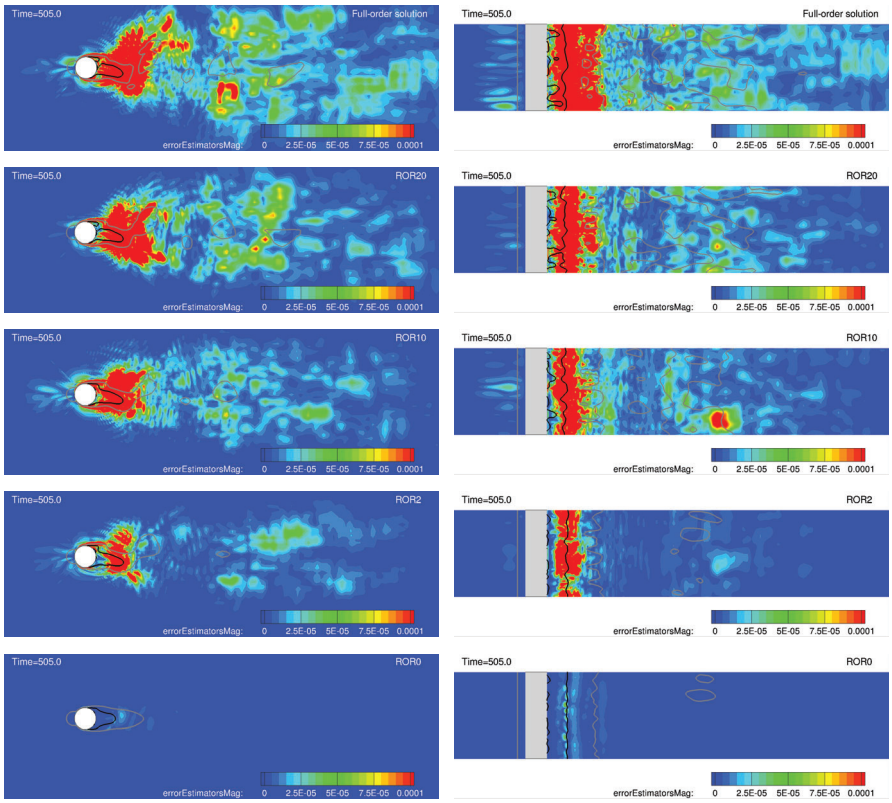


Figure 5.51: QoI error estimators in x-y (left, at $z = 0$) and x-z (right, at $y = 0$) planes evaluated at $t = 505$ for the flow past a 3D cylinder at $Re = 500$, based on full-order flow solutions and different reduced-order representations (RORs) that are computed using 20, 10, 2 POD modes and mean-flow mode. Solid lines denote the streamwise primal velocity, $u = 0$ (black) and 0.6 (grey).

The time-averaged QoI error estimators over this long-time period are presented in Figure 5.52. As shown in Figure 5.52(b), the time-averaged error estimates exhibit the inhomogeneous distribution of high-error regions in the spanwise direction. This is similar to the adjoint Q-structure as shown in Figure 5.30, in which those structures become more and more significant in the upstream regions of the cylinder as the adjoint problem is solved backward in time. This may also be because of the relatively limited time length used here, which is 25 non-dimensional time units, compared to the vortex-shedding period of 4.76. It is expected that the error estimates are homogeneous in the spanwise

direction if we use a time long enough. However, the stability of the adjoint problem needs to be treated carefully. We focus on the impacts of using RORs on the distribution of error estimates here.

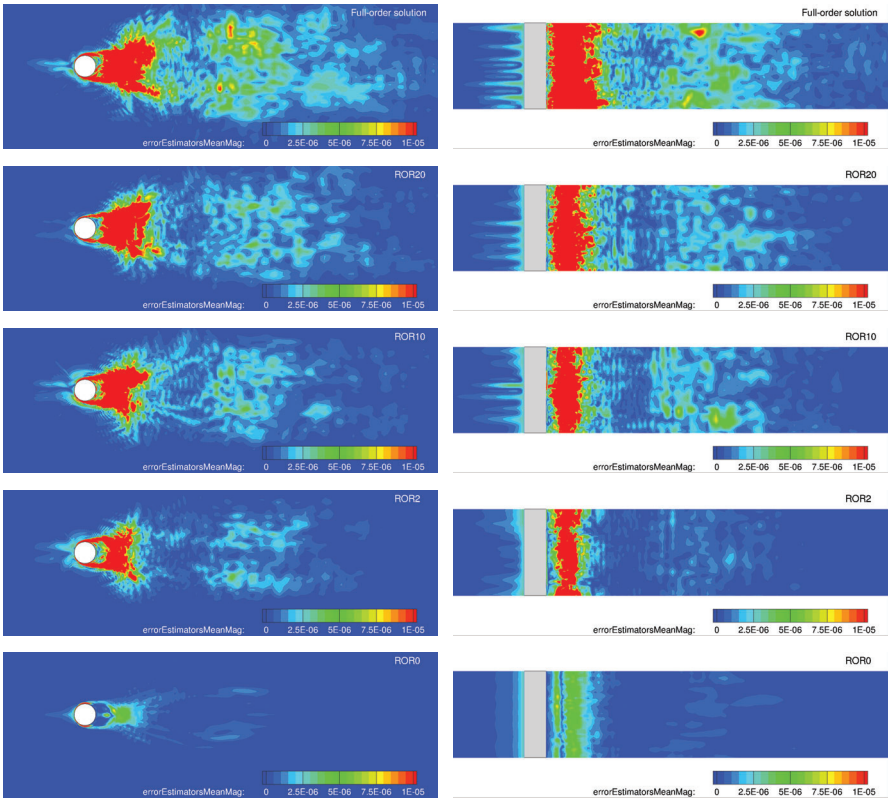
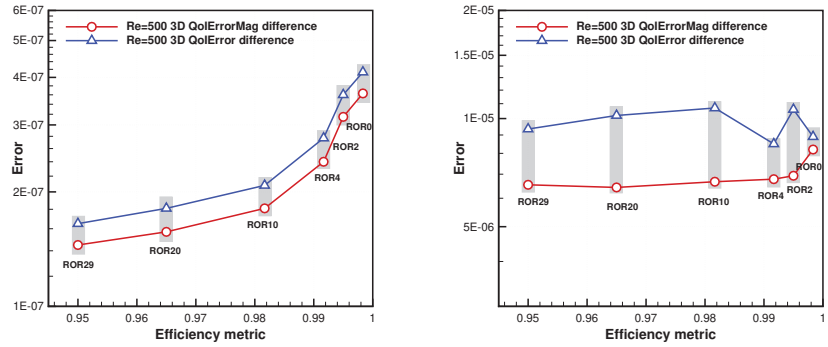


Figure 5.52: Time-averaged QoI error estimators in x-y (left, at $z = 0$) and x-z (right, at $y = 0$) planes for the flow past a 3D cylinder at $Re = 500$ over a long-time period $t \in [505, 530]$, based on full-order flow solutions and different reduced-order representations (RORs) that are computed using 20, 10, 2 POD modes and mean-flow mode.

The impact of RORs on the mean QoI error estimators is similar to the instantaneous case. Again, the distribution with only mean values of the primal flow solutions merely captures a small region after the cylinder while other RORs with POD modes identify much more regions with high QoI error over the flow domain. Increasing the number of POD modes to 20 indeed gives us a good x-y prediction for the error distribution. It is noted that using high-order POD modes is necessary to replicate the distribution structure of QoI error estimators in the regions before the cylinder, where it exhibits a 2D distribution in the short-time analysis. Using a ROR with 20 POD modes can identify the high-error regions before the cylinder and in the cylinder wake. Overall, using the ROR with a proper number of POD modes (20 or more) preserves the ability to seek the region that should be refined for the current turbulent flow.

Figure 5.53 shows the difference of the QoI error estimators between using different RORs and full-order solutions for the 3D turbulent case. Decreasing trends are observed for both short- and long-time period errors as the number of POD modes is increased. The results from the short-time period show a good agreement between the actual and absolute values of QoI error estimators. This attributes to the good convergence for the adjoint field in region 1 as shown in Figure 5.43(b). The deviation between the actual and absolute values in both cases points out that the sign of QoI error estimators can be different when RORs are used. This is probably because the truncation number of POD modes, in this case, leads to neglecting the multiscale of flow solutions and the flow development is delayed compared to that of full-order solutions.



(a) Short-time period (b) Long-time period

Figure 5.53: Difference between the time-averaged QoI error estimators using different RORs and that from full-order flow solutions for the flow past a 3D cylinder at $Re = 500$. The triangle-blue line denotes the difference computed based on actual values of error estimators ($\epsilon_{\Delta J}$) while the circle-red line represents that from the magnitude of error estimators ($\epsilon^{\text{abs}}_{\Delta J}$).

5.7. SUMMARY

The unsteady adjoint problem is formulated for multi-dimensional fluid flow problems in this chapter. The coupling of the PIMPLE solver with the enhanced online algorithm for POD analysis is first described. Then the impacts of RORs on adjoint solutions and adjoint-based error estimation for 2D and 3D flow past a circular cylinder are discussed. The main achievements are as follows.

- A general procedure was presented for the derivation of adjoint equations and their associated boundary conditions. The initial condition was prescribed at the end time for the adjoint problem using a homogeneous boundary condition.
- Interfaces between OpenFOAM and libROM were developed to allow access to memory for both scalar and vector fields created and solved in OpenFOAM. These interfaces reduce the duplication of simulation data and make the online procedure feasible for high-dimensional simulations. The online POD analysis was verified using the 2D cylinder flow at $Re = 300$.

- The accuracy of the flow solver for 2D and 3D cylinder flows was demonstrated with comparison to data in literature before it was applied for the analysis of the transient adjoint problem. The adjoint solver was validated by considering steady flow past a 2D cylinder at $Re = 40$. The steady adjoint was found to be consistent with the results reported in the literature.
- The unsteady adjoint solver was validated by considering a vortex-shedding flow past a 2D cylinder at $Re = 100$. The evolution of the adjoint field is categorized into three phases, including initial growth (phase I), transit (phase II), and a settling-down phase (phase III). The adjoint field becomes a statistically-steady state during the settling-down phase.
- Using the ROR can significantly reduce the memory requirement (only 1% here) for primal solutions of the 2D cylinder flow at $Re = 100$. A ROR with 8 POD modes produces the adjoint field with a lower error (around 1%) while RORs with a lower number of POD modes show a difference in regions around the cylinder and upstream.
- The adjoint field exhibits more dynamic characteristics in a high-Re 2D cylinder flow at $Re = 500$ than at $Re = 100$. The adjoint field develops faster and adjoint vortexes merge after the phase II. The adjoint field can grow without bound over long times when we only use 2, 4, or 6 POD modes. As more POD modes are added, the adjoint field gradually becomes more and more similar to the one based on full-order solutions. A reasonable accuracy (errors less than 2%) is obtained using 29 POD modes while requiring only 0.34% of the memory for the full-order solution. Using the time-averaged flow solution results in adjoint-based error estimation in which the distribution of QoI error estimators is distinct from the baseline. Adding first and second POD modes incorporates the dynamics of the flow field although the resulting magnitudes are of the estimators larger than those of the baseline. The full pattern of QoI error estimators is recovered when we use 10 or more POD modes, which then accurately capture the large-error regions for computing the drag.
- For the 3D cylinder turbulent flow, the adjoint field is shown to grow exponentially using the full-order solutions, which is in good agreement with the literature. However, using a ROR reduces the growth of the transient adjoint field. The coherent structures of the adjoint fields are reasonably captured with relatively few POD modes. When the ROR is built with 29 POD modes, the dynamics of primal and adjoint fields are recovered well. Meanwhile, increasing the number of POD modes leads to a more accurate representation of the QoI error estimation for both the short-time and long-time periods. In contrast, using only the mean-flow mode produces results distinct from the baseline. Using a ROR with 20 POD modes can give us a good prediction for the distribution of large QoI error estimators with less than 4% of the storage requirement of the full-order solutions.

6

CONCLUSIONS AND OUTLOOK

6.1. CONCLUSIONS

This research work addressed the challenges encountered when applying adjoint-based error estimation to unsteady high-fidelity simulations. To this end, we studied adjoint-based error estimation and mesh adaptation for 1D model problems, developed an enhanced online algorithm for POD analysis of high-dimensional data, and investigated the impacts of using RORs on 2D and 3D cylinder flows. Numerical experiments show the potential memory savings using RORs for solving the adjoint problem for unsteady flows. The main contributions of this work are in four different areas as described below.

In Chapter 2, the effectiveness of adjoint-based error estimation in VMM was studied for linear advection-diffusion problems. Firstly, the adjoint-based correction was shown to provide an efficient way to evaluate a QoI compared to using uniform mesh refinement. Secondly, VMM-driven error estimation was shown to provide super convergence and more accurate solutions than traditional FEM. The results indicate that using adjoint solutions and unresolved-scales solutions is crucial for accurate error estimation in VMM. It is anticipated that the benefit of using the adjoint method is more obvious in 2D/3D problems because adjoint-based error estimation allows us to solve the flow problem on a relatively coarse mesh while the computational cost increases dramatically for solving the NS equations of 2D/3D problems on a fine mesh.

In Chapter 3, an enhanced online algorithm for POD analysis was developed to deal with the high dimensionality of LES primal solutions. This method is based on the incremental SVD, which can efficiently perform the POD analysis as the flow problem is solved. Two lower-bound estimators were established to evaluate the energy captured by the selected POD modes, which allow for a posteriori estimation of the accuracy of a ROR. The choice for mode truncation number and the resulting cost reduction are case dependent in practice. Our error estimator can guide users by detecting if the value of this user-defined parameter is appropriate. It is reasonable to assume that future users have some experience with their application field and can therefore make a reasonably conservative estimate. Even if this estimate is very conservative, the cost reduction can still be significant. Our experience with large 3D flow simulations indicates that less than 1% of the full modal basis is typically required for a physical analysis [164]. Numerical results demonstrate that the enhanced online algorithm can significantly reduce the computational cost compared with the standard incremental SVD while maintaining good accuracy. The enhanced online algorithm was shown to scale well in parallel and its speedup remained constant when using multiple cores. Overall, the results of this chapter imply that the enhanced online algorithm will be useful for the engineering analysis of high-dimensional problems.

In Chapter 4, we developed an adjoint-based mesh adaptation strategy for unsteady 1D Burgers problems augmented by an enhanced online algorithm for building POD-based RORs. These were used to replace the full-order primal solution when solving the adjoint problem, and the error estimators for an AMR algorithm were evaluated using these RORs and adjoint solutions. Numerical results showed that the QoI estimate converged reliably on very coarse meshes during adjoint-based AMR and reached a high level of accuracy at moderate levels of refinement. Using a ROR with four POD modes (capturing 99.9% of total energy) reproduced the convergence history of a full-order solution-driven AMR. Although using a single-mode ROR led to suboptimal meshes, the

QoI obtained was still better than the one obtained using uniform refinement. The enhanced online algorithm allows for efficient ROR-driven adjoint-based AMR while not altering the fast convergence from ROR-driven AMR. As it is a purely data-based technique, the enhanced online algorithm can be expected to deliver similar or greater reductions in the memory required for the adjoint-based mesh adaptation in 2D/3D problems.

In Chapter 5, we derived an unsteady adjoint formulation for 2D/3D flow problems, which can be extended to various QoIs. The enhanced online algorithm developed in Chapter 3 was amalgamated into OpenFOAM using two interfaces. The unsteady adjoint solver was analyzed on 2D and 3D cylinder flows. Steady adjoint solutions were found when the adjoint problem is solved with only the time-averaged primal solution. When using the full-order solution, the 2D cylinder adjoint solution at $Re = 100$ was found to undergo three phases: initial growth (phase I), transit (phase II), and settling-down phase (phase III). It arrived at a statistically-steady state during phase III. Using RORs was found to significantly reduce the memory requirement with a limited effect on the adjoint field. For example, a difference around 1% was observed for the adjoint field using a ROR with 8 POD modes while requiring less than 0.3% of the storage of full-order primal solutions.

The evolution of a 2D cylinder adjoint field was observed to behave similarly for $Re = 100$ and $Re = 500$, with the latter having more dynamic features. The $Re = 500$ adjoint field also developed faster in phase II and was found to grow significantly when only 2, 4, or 6 POD modes were used. As more POD modes were added, the adjoint field gradually became similar to the one based on full-order solutions. Using 29 POD modes for the $Re = 500$ produced adjoint solutions with a difference of less than 2% at only 0.34% of storage requirement. The QoI error estimation using only the mean flow solution showed a pattern distinct from the one using full-order unsteady solutions. This indicates that the regions that are important for determining drag can be different for steady and unsteady simulations.

For a 3D turbulent flow past a circular cylinder at $Re = 500$, the growth of the adjoint field is exponential as the problem is solved backward in time. Nevertheless, this unbounded growth was reduced by using RORs. Using a ROR with 29 POD modes produced adjoint solutions with dynamic features similar to those obtained using full-order solutions. This ROR required less than 5% of the storage requirement of the full-order solutions, with an error lower than 5.2%. It was able to predict the regions with high QoI error estimators for short-time and long-time periods, in both streamwise and spanwise directions. This indicates that using an appropriate ROR can make the adjoint-based sensitivity analysis of LES predictions more tractable.

6.2. OUTLOOK

We have paved ways to reduce the computational cost of solving the adjoint problem for LES using RORs constructed with an enhanced online algorithm. However, there is still some work to do before applications to engineering problems.

There are different subgrid models used in LES. We chose the VMM to investigate the adjoint-based error estimations and AMR in this work. Considering other subgrid models, such as implicit subgrid models [165, 166], would be a further step to explore

the utility of adjoint-based mesh adaptation for practical applications.

The quality of a ROR can be efficiently evaluated by the error estimates proposed in this work. However, the choice of a truncation number still requires user input. The algorithm can be further improved by developing a method for choosing the truncation number automatically. In addition, the thresholds used in the incremental SVD need to be adjusted for various problems. To minimize the influence of round-off errors, the development of user-independent criteria for choosing such thresholds would be useful.

Typically, POD analysis is usually carried out on slices or sub-domains of flow problems in high-fidelity simulations. This is because of the demanding memory requirement for storing the data for the whole domain. The proposed algorithm avoids the storage requirement and thus can be used for such analysis over the whole computational domain, which will provide us with a global view of the dynamics in complex flow fields. It is believed that this enhanced algorithm can also benefit DMD [167] for analyzing high-dimensional and complex flow problems.

We found that using RORs in adjoint-based error estimation can effectively identify the regions that have largest effects on the accuracy of predictions for 2D and 3D problems. Thus, using RORs in AMR is the next step in developing feasible adjoint-based AMR for engineering problems. The current study showed that some RORs can produce results similar to those of full-order primal solutions but some induce unlimited growth of the adjoint solutions. It is conjectured that different POD modes play different roles in solving the adjoint problem. We typically select the first to fifth POD modes to build a ROR if the truncation number is selected to be 5, for instance. This ROR may lead to an unlimited growth of adjoint solutions while a different ROR using first to fourth POD modes and a seventh POD mode could give more stable adjoint solutions. Consequently, the study of how to select POD modes for building RORs would be useful to understand the mechanism on which a reliable ROR relies and thus to find a proper ROR for solving the adjoint problem. Recently, researchers have solved the steady adjoint problem using time-averaged primal solutions from LES. This has been shown to be effective for shape optimization for flows with separation [168] and mesh adaption for LES [169, 170]. We observed that the steady adjoint obtained when using mean primal solutions gives a distribution that significantly differs from the unsteady case. It would be interesting to study the difference between those approaches and establish guidelines for the usage of adjoint-based methods in steady and unsteady problems.

BIBLIOGRAPHY

[1] J. Slotnick, et al., CFD vision 2030 study: a path to revolutionary computational aerosciences, Contractor Report NASA/CR–2014-218178 (NASA Langley Research Center, 2014).

[2] P. R. Spalart, and V. Venkatakrishnan, On the role and challenges of CFD in the aerospace industry, The Aeronautical Journal **120**, 209–232 (2016).

[3] F. D. Witherden, and A. Jameson, Future directions in computational fluid dynamics, AIAA paper 2017-3791 (2017).

[4] G. Castiglioni, et al., Numerical simulations of separated flows at moderate reynolds numbers appropriate for turbine blades and unmanned aero vehicles, International Journal of Heat and Fluid Flow **49**, 91–99 (2014).

[5] J. Quaatz, et al., Large-eddy simulation of a pseudo-shock system in a laval nozzle, International Journal of Heat and Fluid Flow **49**, 108–115 (2014).

[6] P. Sagaut, Large eddy simulation for incompressible flows: an introduction, 3rd ed (Springer, Berlin, 2006).

[7] J. Larsson, Grid-adaptation for chaotic multi-scale simulations as a verification-driven inverse problem, AIAA paper 2018-0371 (2018).

[8] J. Larsson, and Q. Wang, The prospect of using large eddy and detached eddy simulations in engineering design, and the research required to get there, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **372**, 20130329 (2014).

[9] K. J. Fidkowski, and D. L. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA Journal **49**, 673–694 (2011).

[10] T. Dubois, F. Jauberteau, and R. Temam, Dynamic multilevel methods and the numerical simulation of turbulence (Cambridge University Press, 1999).

[11] S. Léonard, M. Terracol, and P. Sagaut, A wavelet-based adaptive mesh refinement criterion for large-eddy simulation, Journal of Turbulence **7**, N64 (2006).

[12] A. Jameson, Aerodynamic shape optimization using the adjoint method, Technical Report (von Karmann Institute, 2003).

[13] M. B. Giles, and N. A. Pierce, An introduction to the adjoint approach to design, Flow, Turbulence and Combustion **65**, 393–415 (2000).

[14] J. Hoffman, et al., Towards a parameter-free method for high reynolds number turbulent flow simulation based on adaptive finite element approximation, Computer Methods in Applied Mechanics and Engineering **288**, 60–74 (2015).

[15] R. Becker, and R. Rannacher, An optimal control approach to a posteriori error estimation in finite element methods, Acta Numerica **10**, 1–102 (2001).

[16] R. Becker, V. Heuveline, and R. Rannacher, An optimal control approach to adaptivity in computational fluid mechanics, International Journal for Numerical Methods in Fluids **40**, 105–120 (2002).

[17] M. J. Berger, and A. Jameson, Automatic adaptive grid refinement for the euler equations, AIAA Journal **23**, 561–568 (1985).

[18] M. Aftosmis, M. Berger, and G. Adomavicius, A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries, AIAA paper 2000-808 (2000).

[19] C. Roy, Strategies for driving mesh adaptation in CFD, AIAA paper 2009-1302 (2009).

[20] S. B. Pope, Ten questions concerning the large-eddy simulation of turbulent flows, New Journal of Physics **6**, 35 (2004).

[21] A. Bazile, et al., Variational Multiscale error estimator for anisotropic adaptive fluid mechanic simulations: Application to convection-diffusion problems, Computer Methods in Applied Mechanics and Engineering, 22 (2018).

[22] D. A. Venditti, and D. L. Darmofal, Grid adaptation for functional outputs: application to two-dimensional inviscid flows, Journal of Computational Physics **176**, 40–69 (2002).

[23] M. A. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, AIAA Journal **42**, 1854–1862 (2004).

[24] T. J. Baker, Mesh adaptation strategies for problems in fluid dynamics, Finite Elements in Analysis and Design **25**, 243–273 (1997).

[25] P. K. Sweby, and H. C. Yee, On the dynamics of some grid adaption schemes, NASA contractor report NASA-CR 195092 (NASA Ames Research Center, 1994).

[26] T. A. Oliver, A high-order, adaptive, discontinuous galerkin finite element method for the reynolds-averaged navier-stokes equations, PhD thesis (Massachusetts Institute of Technology, U.S., 2008).

[27] R. Balasubramanian, and J. Newman, Adjoint-based error estimation and grid adaptation for functional outputs: application to two-dimensional, inviscid, incompressible flows, Computers & Fluids **38**, 320–332 (2009).

[28] J. Jansson, et al., Time-Resolved Adaptive Direct FEM Simulation of High-Lift Aircraft Configurations, *Numerical Simulation of the Aerodynamics of High-Lift Configurations* (Springer, Cham, 2018), pp. 67–92.

[29] M. Nazarov, and J. Hoffman, Residual-based artificial viscosity for simulation of turbulent compressible flow using adaptive finite element methods, International Journal for Numerical Methods in Fluids **71**, 339–357 (2013).

[30] L. Shi, and Z. Wang, Adjoint-based error estimation and mesh adaptation for the correction procedure via reconstruction method, Journal of Computational Physics **295**, 261–284 (2015).

[31] M. Nazarov, and J. Hoffman, An adaptive finite element method for inviscid compressible flow, International Journal for Numerical Methods in Fluids **64**, 1102–1128 (2010).

[32] M. Kouhi, E. Oñate, and D. Mavriplis, Adjoint-based adaptive finite element method for the compressible euler equations using finite calculus, Aerospace Science and Technology **46**, 422–435 (2015).

[33] M. Nemec, M. Aftosmis, and M. Wintzer, Adjoint-based adaptive mesh refinement for complex geometries, AIAA paper 2008-725 (2008).

[34] Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification (National Academies Press, Washington, U.S., 2012).

[35] R. P. Dwight, Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation, Journal of Computational Physics **227**, 2845–2863 (2008).

[36] M. Giles, et al., Adjoint equations in CFD - duality, boundary conditions and solution behaviour, AIAA paper 1997-1850 (1997).

[37] K. Duraisamy, et al., Error estimation for high speed flows using continuous and discrete adjoints, AIAA paper 2010-128 (2010).

[38] M. B. Giles, and N. A. Pierce, Analytic adjoint solutions for the quasi-one-dimensional euler equations, Journal of Fluid Mechanics **426**, 327–345 (2001).

[39] Q. Wang, and J.-H. Gao, The drag-adjoint field of a circular cylinder wake at reynolds numbers 20, 100 and 500, Journal of Fluid Mechanics **730**, 145–161 (2013).

[40] D. A. Venditti, and D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow, Journal of Computational Physics **164**, 204–227 (2000).

[41] M. A. Ceze, and K. J. Fidkowski, High-order output-based adaptive simulations of turbulent flow in two dimensions, AIAA Journal **54**, 2611–2625 (2016).

[42] R. Hartmann, et al., Error estimation and adaptive mesh refinement for aerodynamic flows, ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications (2010), pp. 339–353.

[43] M. B. Giles, and N. A. Pierce, Adjoint error correction for integral outputs, *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics* (Springer, Berlin, Heidelberg, 2003), pp. 47–95.

[44] T. J. Barth, Space-time error representation and estimation in navier-stokes calculations, Complex effects in large eddy simulations, Lecture Notes in Computational Science and Engineering (2007), pp. 29–48.

[45] T. J. Barth, and M. G. Larson, A posteriori error estimates for higher order godunov finite volume methods on unstructured meshes, Finite volumes for complex applications III (2002).

[46] R. Hartmann, and P. Houston, Adaptive discontinuous galerkin finite element methods for the compressible euler equations, Journal of Computational Physics **183**, 508–532 (2002).

[47] K. J. Fidkowski, and D. L. Darmofal, Output-based adaptive meshing using triangular cut cells, TR-06-2 (MIT Aerospace Computational Design Laboratory Report, 2006).

[48] F. Alauzet, and A. Loseille, A decade of progress on anisotropic mesh adaptation for computational fluid dynamics, Computer-Aided Design **72**, 13–39 (2016).

[49] J. Hoffman, On duality-based a posteriori error estimation in various norms and linear functionals for large eddy simulation, SIAM Journal on Scientific Computing **26**, 178–195 (2004).

[50] R. Rannacher, Adaptive galerkin finite element methods for partial differential equations, Journal of Computational and Applied Mathematics **128**, 205–233 (2001).

[51] N. Pierce, and M. Giles, Adjoint Recovery of Superconvergent Functionals from PDE Approximations, SIAM Review **42**, 247–264 (2000).

[52] K. J. Fidkowski, Output error estimation strategies for discontinuous galerkin discretizations of unsteady convection-dominated flows, International Journal for Numerical Methods in Engineering **88**, 1297–1322 (2011).

[53] J. Hoffman, and C. Johnson, Adaptive finite element methods for incompressible fluid flow, *Error estimation and adaptive discretization methods in computational fluid dynamics* (Springer, Berlin, Heidelberg, 2003), pp. 97–157.

[54] J. Hoffman, and C. Johnson, A new approach to computational turbulence modeling, Computer Methods in Applied Mechanics and Engineering **195**, 2865–2880 (2006).

[55] J.-D. Müller, and M. Giles, Solution adaptive mesh refinement using adjoint error analysis, AIAA paper 2001-2550 (2001).

[56] D. A. Venditti, and D. L. Darmofal, Grid adaptation for functional outputs of 2-D compressible flow simulations, AIAA paper 2000-2244 (2000).

[57] D. Venditti, and D. Darmofal, A multilevel error estimation and grid adaptive strategy for improving the accuracy of integral outputs, AIAA paper 1999-3292 (1999).

[58] R. P. Dwight, Goal-oriented mesh adaptation for finite volume methods using a dissipation-based error indicator, International Journal for Numerical Methods in Fluids **56**, 1193–1200 (2008).

[59] A. Carrier, J. Nicolle, and C. Deschênes, Use of the adjoint method to assess the error in simulations, ASME fluids engineering division summer meeting (2016), p. 9.

[60] M. Ceze, and K. J. Fidkowski, Drag prediction using adaptive discontinuous finite elements, Journal of Aircraft **51**, 1284–1294 (2014).

[61] D. A. Venditti, and D. L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics **187**, 22–46 (2003).

[62] A. Loseille, A. Dervieux, and F. Alauzet, Fully anisotropic goal-oriented mesh adaptation for 3d steady euler equations, Journal of Computational Physics **229**, 2866–2897 (2010).

[63] K. J. Fidkowski, Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints, Journal of Computational Physics **341**, 258–277 (2017).

[64] M. Besier, and R. Rannacher, Goal-oriented space-time adaptivity in the finite element galerkin method for the computation of nonstationary incompressible flow, International Journal for Numerical Methods in Fluids **70**, 1139–1166 (2012).

[65] J. A. Krakos, and D. L. Darmofal, Anisotropic output-based mesh optimization for unsteady flows, AIAA paper 2013-3083 (2013).

[66] J. Hoffman, Adaptive simulation of the subcritical flow past a sphere, Journal of Fluid Mechanics **568**, 77 (2006).

[67] K. J. Fidkowski, and Y. Luo, Output-based space–time mesh adaptation for the compressible navier–stokes equations, Journal of Computational Physics **230**, 5753–5773 (2011).

[68] D. J. Lea, M. R. Allen, and T. W. N. Haine, Sensitivity analysis of the climate of a chaotic system, Tellus A **52**, 523–532 (2000).

[69] M. Schmich, and B. Vexler, Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations, SIAM Journal on Scientific Computing **30**, 369–393 (2008).

[70] B. Flynt, and D. Mavriplis, Discrete adjoint based adaptive error control in unsteady flow problems, AIAA paper 2012-78 (2012).

[71] Q. Wang, P. Moin, and G. Iaccarino, Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation, SIAM Journal on Scientific Computing **31**, 2549–2567 (2009).

[72] Q. Wang, Forward and adjoint sensitivity computation of chaotic dynamical systems, Journal of Computational Physics **235**, 1–13 (2013).

[73] K. C. Hall, J. P. Thomas, and E. H. Dowell, Proper orthogonal decomposition technique for transonic unsteady aerodynamic flows, AIAA Journal **38**, 1853–1862 (2000).

[74] T. P. Miyanawala, and R. K. Jaiman, Decomposition of wake dynamics in fluid–structure interaction via low-dimensional models, Journal of Fluid Mechanics **867**, 723–764 (2019).

[75] M. J. Zahr, and C. Farhat, Progressive construction of a parametric reduced-order model for PDE-constrained optimization, International Journal for Numerical Methods in Engineering **102**, 1111–1135 (2015).

[76] T. Bui-Thanh, K. Willcox, and O. Ghattas, Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications, AIAA Journal **46**, 2520–2529 (2008).

[77] S. S. Ravindran, A reduced-order approach for optimal control of fluids using proper orthogonal decomposition, International Journal for Numerical Methods in Fluids **34**, 425–448 (2000).

[78] K. Taira, et al., Modal analysis of fluid flows: an overview, AIAA Journal **55**, 4013–4041 (2017).

[79] P. Holmes, et al., Turbulence, coherent structures, dynamical systems and symmetry, 2nd ed (Cambridge University Press, New York, 2012).

[80] J. N. Kutz, Data-driven modeling & scientific computation: methods for complex systems & big data, 1st edition (Oxford University Press, Oxford, 2013).

[81] T. Lassila, et al., Model order reduction in fluid dynamics: challenges and perspectives, *Reduced order methods for modeling and computational reduction* (Springer International Publishing, Cham, 2014), pp. 235–273.

[82] B. R. Noack, et al., A hierarchy of low-dimensional models for the transient and post-transient cylinder wake, Journal of Fluid Mechanics **497**, 335–363 (2003).

[83] R. Rubini, D. Lasagna, and A. Da Ronch, The l_1 -based sparsification of energy interactions in unsteady lid-driven cavity flow, Journal of Fluid Mechanics **905**, A15 (2020).

[84] S. Walton, O. Hassan, and K. Morgan, Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions, Applied Mathematical Modelling **37**, 8930–8945 (2013).

[85] Z. Sun, et al., Non-intrusive reduced-order model for predicting transonic flow with varying geometries, Chinese Journal of Aeronautics **33**, 508–519 (2020).

[86] M. Brand, Incremental singular value decomposition of uncertain data with missing values, European conference on computer vision (2002), pp. 707–720.

[87] C. Baker, K. Gallivan, and P. Van Dooren, Low-rank incremental methods for computing dominant singular subspaces, Linear Algebra and Its Applications **436**, 2866–2888 (2012).

[88] H. Fareed, et al., Incremental proper orthogonal decomposition for PDE simulation data, Computers & Mathematics with Applications **75**, 1942–1960 (2018).

[89] G. M. Oxberry, et al., Limited-memory adaptive snapshot selection for proper orthogonal decomposition, International Journal for Numerical Methods in Engineering **109**, 198–217 (2017).

[90] K. Li, et al., POD-based model order reduction with an adaptive snapshot selection for a discontinuous galerkin approximation of the time-domain maxwell's equations, Journal of Computational Physics **396**, 106–128 (2019).

[91] P. Phalippou, et al., 'on-the-fly' snapshots selection for proper orthogonal decomposition with application to nonlinear dynamics, Computer Methods in Applied Mechanics and Engineering **367**, 113120 (2020).

[92] C. Vezyris, E. Papoutsis-Kiachagias, and K. Giannakoglou, On the incremental singular value decomposition method to support unsteady adjoint-based optimization, International Journal for Numerical Methods in Fluids **91**, 315–331 (2019).

[93] S. Toosi, and J. Larsson, Grid-adaptation and convergence-verification in large eddy simulation: a robust and systematic approach, AIAA paper 2018-3406 (2018).

[94] J. Gullbrand, and F. K. Chow, The effect of numerical errors and turbulence models in large-eddy simulations of channel flow, with and without explicit filtering, Journal of Fluid Mechanics **495**, 323–341 (2003).

[95] T. J. Hughes, et al., The variational multiscale method—a paradigm for computational mechanics, en, Computer Methods in Applied Mechanics and Engineering **166**, 3–24 (1998).

[96] Y. Bazilevs, K. Takizawa, and T. E. Tezduyar, New directions and challenging computations in fluid dynamics modeling with stabilized and multiscale methods, Mathematical Models and Methods in Applied Sciences **25**, 2217–2226 (2015).

[97] N. Ahmed, et al., A review of variational multiscale methods for the simulation of turbulent incompressible flows, Archives of Computational Methods in Engineering **24**, 115–164 (2017).

[98] Y. Bazilevs, et al., Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering **197**, 173–201 (2007).

[99] I. Akkerman, et al., The role of continuity in residual-based variational multiscale modeling of turbulence, Computational Mechanics **41**, 371–378 (2008).

[100] S. J. Hulshoff, E. A. Munts, and J. Labrujere, Wall-stress boundary conditions for variational-multiscale LES, International Journal for Numerical Methods in Fluids **66**, 1341–1353 (2011).

[101] L. Chen, S. J. Hulshoff, and Y. Wang, 2d residual-based LES of flow around a pipeline close to a flat seabed, Computer Methods in Applied Mechanics and Engineering **363**, 112788 (2020).

[102] B. N. Granzow, M. S. Shephard, and A. A. Oberai, Output-based error estimation and mesh adaptation for variational multiscale methods, en, Computer Methods in Applied Mechanics and Engineering **322**, 441–459 (2017).

[103] N. Chandramoorthy, et al., Feasibility analysis of ensemble sensitivity computation in turbulent flows, AIAA Journal **57**, 4514–4526 (2019).

[104] Q. Wang, R. Hu, and P. Blonigan, Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations, Journal of Computational Physics **267**, 210–224 (2014).

[105] N. Chandramoorthy, and Q. Wang, Sensitivity computation of statistically stationary quantities in turbulent flows, AIAA paper 2019-3426 (2019).

[106] C. Talnikar, Q. Wang, and G. M. Laskowski, Unsteady adjoint of pressure loss for a fundamental transonic turbine vane, Journal of Turbomachinery **139**, 031001 (2017).

[107] P. J. Blonigan, and Q. Wang, Multiple shooting shadowing for sensitivity analysis of chaotic dynamical systems, Journal of Computational Physics **354**, 447–475 (2018).

[108] K. Shawki, and G. Papadakis, A preconditioned multiple shooting shadowing algorithm for the sensitivity analysis of chaotic systems, Journal of Computational Physics **398**, 108861 (2019).

[109] A. Ni, and Q. Wang, Sensitivity analysis on chaotic dynamical systems by non-intrusive least squares shadowing (NILSS), Journal of Computational Physics **347**, 56–77 (2017).

[110] D. Lasagna, A. Sharma, and J. Meyers, Periodic shadowing sensitivity analysis of chaotic systems, Journal of Computational Physics **391**, 119–141 (2019).

[111] J. Hoffman, and C. Johnson, Stability of the dual navier–stokes equations and efficient computation of mean output in turbulent flow using adaptive DNS/LES, Computer Methods in Applied Mechanics and Engineering **195**, 1709–1721 (2006).

[112] L. Lapworth, The challenges for Aero-Engine CFD, Technical Report (ICFD 25th Anniversary Meeting, Oxford, 2008), p. 27.

[113] T. J. R. Hughes, and G. Sangalli, Variational Multiscale Analysis: the Fine-scale Green's Function, Projection, Optimization, Localization, and Stabilized Methods, en, SIAM Journal on Numerical Analysis **45**, 539–557 (2007).

[114] X. Li, S. Hulshoff, and S. Hickel, An enhanced algorithm for online proper orthogonal decomposition and its parallelization for unsteady simulations, Computers & Mathematics with Applications **126**, 43–59 (2022).

[115] C. W. Rowley, and S. T. Dawson, Model reduction for flow analysis and control, Annual Review of Fluid Mechanics **49**, 387–417 (2017).

[116] N. Ferro, S. Micheletti, and S. Perotto, POD-assisted strategies for structural topology optimization, Computers & Mathematics with Applications **77**, 2804–2820 (2019).

[117] X. Sun, and J.-I. Choi, Non-intrusive reduced-order modeling for uncertainty quantification of space–time-dependent parameterized problems, Computers & Mathematics with Applications **87**, 50–64 (2021).

[118] M. Bergmann, L. Cordier, and J.-P. Brancher, Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model, Physics of Fluids **17**, 097101 (2005).

[119] T. Akman, Local improvements to reduced-order approximations of optimal control problems governed by diffusion-convection-reaction equation, Computers & Mathematics with Applications **70**, 104–131 (2015).

[120] G. Berkooz, P. Holmes, and J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annual Review of Fluid Mechanics **25**, 539–575 (1993).

[121] J. L. Lumey, Stochastic tools in turbulence, Applied mathematics and mechanics (Academic Press, New York, 1970).

[122] L. Sirovich, Turbulence and the dynamics of coherent structures part i: coherent structures, Quarterly of Applied Mathematics **45**, 561–571 (1987).

[123] R. A. Horn, and C. R. Johnson, Matrix analysis (Cambridge University Press, New York, 1990).

[124] B. Champagne, Adaptive eigendecomposition of data covariance matrices based on first-order perturbations, IEEE Transactions on Signal Processing **42**, 2758–2770 (1994).

[125] Chao Xu, Lixiang Luo, and E. Schuster, On recursive proper orthogonal decomposition via perturbation theory with applications to distributed sensing in cyber-physical systems, Proceedings of the 2010 american control conference (2010), pp. 4905–4910.

[126] N. Halko, P. G. Martinsson, and J. A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Review **53**, 217–288 (2011).

[127] C. Bach, et al., Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems, International Journal for Numerical Methods in Engineering **118**, 209–241 (2019).

[128] H. Fareed, and J. R. Singler, Error analysis of an incremental proper orthogonal decomposition algorithm for PDE simulation data, Journal of Computational and Applied Mathematics **368**, 112525 (2020).

[129] X. Li, S. Hulshoff, and S. Hickel, Towards adjoint-based mesh refinement for large eddy simulation using reduced-order primal solutions: preliminary 1d burgers study, Computer Methods in Applied Mechanics and Engineering **379**, 113733 (2021).

[130] C. A. Beattie, et al., A domain decomposition approach to POD, Proceedings of the 45th IEEE conference on decision and control (2006), pp. 6750–6756.

[131] Z. Wang, B. McBee, and T. Iliescu, Approximate partitioned method of snapshots for POD, Journal of Computational and Applied Mathematics **307**, 374–384 (2016).

[132] M. A. Iwen, and B. W. Ong, A distributed and incremental SVD algorithm for agglomerative data analysis on large networks, SIAM Journal on Matrix Analysis and Applications **37**, 1699–1718 (2016).

[133] W. Arrighi, G. Oxberry, and K. Chand, libROM user guide and design, Technical Report (Lawrence Livermore National Laboratory, 2015).

[134] Y. Choi, et al., Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems, Journal of Computational Physics **424**, 109845 (2021).

[135] D. A. Jackson, Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches, Ecology **74**, 2204–2214 (1993).

[136] J. Caldwell, P. Wanless, and A. E. Cook, A finite element approach to Burgers' equation, Applied Mathematical Modelling **5**, 189–193 (1981).

[137] N. J. Georgiadis, D. P. Rizzetta, and C. Fureby, Large-eddy simulation: current capabilities, recommended practices, and future research, AIAA Journal **48**, 1772–1784 (2010).

[138] Z. Wang, and A. Oberai, Spectral analysis of the dissipation of the residual-based variational multiscale method, Computer Methods in Applied Mechanics and Engineering **199**, 810–818 (2010).

[139] C. Othmer, A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows, International Journal for Numerical Methods in Fluids **58**, 861–877 (2008).

[140] I. Kavvadias, E. Papoutsis-Kiachagias, and K. Giannakoglou, On the proper treatment of grid sensitivities in continuous adjoint methods for shape optimization, Journal of Computational Physics **301**, 1–18 (2015).

[141] A. Zymaris, et al., Continuous adjoint approach to the spalart–allmaras turbulence model for incompressible flows, Computers & Fluids **38**, 1528–1538 (2009).

[142] W. K. Anderson, and V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, Computers & Fluids **28**, 443–480 (1999).

[143] R. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, Journal of Computational Physics **62**, 40–65 (1986).

[144] J. Park, K. Kwon, and H. Choi, Numerical solutions of flow past a circular cylinder at reynolds numbers up to 160, KSME International Journal **12**, 1200–1205 (1998).

[145] J. Kim, D. Kim, and H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, Journal of Computational Physics **171**, 132–150 (2001).

[146] S. C. R. Dennis, and G.-Z. Chang, Numerical solutions for steady flow past a circular cylinder at reynolds numbers up to 100, Journal of Fluid Mechanics **42**, 471–489 (1970).

[147] M. Meyer, S. Hickel, and N. Adams, Assessment of implicit large-eddy simulation with a conservative immersed interface method for turbulent cylinder flow, International Journal of Heat and Fluid Flow **31**, 368–377 (2010).

[148] A. S. Grove, F. H. Shair, and E. E. Petersen, An experimental investigation of the steady separated flow past a circular cylinder, Journal of Fluid Mechanics **19**, 60–80 (1964).

[149] V. Kitsios, Recovery of fluid mechanical modes in unsteady separated flows, PhD thesis (University of Melbourne, Australia, 2010).

[150] H. M. Blackburn, and R. D. Henderson, A study of two-dimensional flow past an oscillating cylinder, Journal of Fluid Mechanics **385**, 255–286 (1999).

[151] R. D. Henderson, Details of the drag curve near the onset of vortex shedding, Physics of Fluids **7**, 2102–2104 (1995).

[152] G. E. Karniadakis, and G. S. Triantafyllou, Three-dimensional dynamics and transition to turbulence in the wake of bluff objects, Journal of Fluid Mechanics **238**, 1–30 (1992).

[153] H. Baek, and G. Karniadakis, Suppressing vortex-induced vibrations via passive means, Journal of Fluids and Structures **25**, 848–866 (2009).

[154] L.-X. Chen, et al., Direct numerical simulation of heat transfer of lead–bismuth eutectic flow over a circular cylinder at re = 500, Frontiers in Energy Research **10**, 823590 (2022).

[155] M. M. Zdravkovich, Flow Around Circular Cylinders–Volume 1: Fundamentals (Oxford University Press, New York, 1997).

[156] J. H. Lienhard, Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders (Washington State University, Pullman, WA, 1966).

[157] A. Roshko, On the development of turbulent wakes from vortex streets, Technical Report 1191 (NACA, 1954).

[158] S. Sirisup, et al., DNS and experiments of flow past a wired cylinder at low reynolds number, European Journal of Mechanics - B/Fluids **23**, 181–188 (2004).

[159] C. Norberg, Fluctuating lift on a circular cylinder: review and new measurements, Journal of Fluids and Structures **17**, 57–96 (2003).

[160] L. Xiyun, and L. Guocan, A large eddy simulation of the near wake of a circular cylinder, ACTA MECHANICA SINICA, 13 (2002).

[161] J. Jeong, and F. Hussain, On the identification of a vortex, Journal of Fluid Mechanics **285**, 69–94 (1995).

[162] G. K. W. Kenway, et al., Effective adjoint approaches for computational fluid dynamics, Progress in Aerospace Sciences **110**, 100542 (2019).

[163] S. M. Kast, An introduction to adjoints and output error estimation in computational fluid dynamics, arxiv Report (2016).

[164] V. Pasquariello, S. Hickel, and N. A. Adams, Unsteady effects of strong shock-wave/boundary-layer interaction at high reynolds number, Journal of Fluid Mechanics **823**, 617–657 (2017).

[165] S. Hickel, T. Kempe, and N. A. Adams, Implicit large-eddy simulation applied to turbulent channel flow with periodic constrictions, Theoretical and Computational Fluid Dynamics **22**, 227–242 (2008).

[166] M. Meyer, et al., A conservative immersed interface method for large-eddy simulation of incompressible flows, Journal of Computational Physics **229**, 6300–6317 (2010).

[167] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of Fluid Mechanics **656**, 5–28 (2010).

[168] G. Alessi, et al., Adjoint shape optimization coupled with LES-adapted RANS of a u-bend duct for pressure loss reduction, Computers & Fluids **228**, 105057 (2021).

[169] Y. Jiang, and S. Nadarajah, Adjoint-based error estimation for grid adaptation for large eddy simulation, Computers & Fluids **236**, 105295 (2022).

[170] K. J. Fidkowski, Output-based error estimation and mesh adaptation for unsteady turbulent flow simulations, Computer Methods in Applied Mechanics and Engineering **399**, 115322 (2022).

ACKNOWLEDGEMENTS

I have envisioned my PhD study before my PhD journal but it was never such realistic until I finalized it. This experience cannot happen without the opportunity to pursue a PhD study at Delft. Thank Prof. Stefan Hickel for offering me this opportunity and making my journey come true. I sincerely appreciate your help and advice in the past, present as well as future.

Thank Stefan for our weekly discussions and talks. It was inspiring for me to have chats with you when I encountered difficulties. We came up with new ideas for the work, which was interesting and exciting. It is not easy to make a decision by myself only, but you have guided me in the right direction and carefully analyzed my situation while giving me the right to make the final decision. You also provided me with the freedom for exploring different research topics and attending conferences and workshops. As discussed at our first meeting, I wanted to develop the applied LES for engineering applications, you had answered “hopefully" for this ambitious goal. Fortunately, we have paved the way to this aim. Participating in the supervision of your master student was really impressive. You are not only the promoter but also a role model.

I need to give sincere thanks to Dr. Steven Hulshoff. You have taught me to plan and manage my research project in a proper manner. I really appreciate your consistent support during my PhD study and the revision of our papers. Your suggestions made a significant contribution to our achievements. The discussions with you opened my mind. It was an impressive experience to join your course as a teaching assistant. I would be lonely and depressed during my research without your participation in our weekly meeting. You can offer me a third-party view of my work. Thanks for your patience, practice and revision for improving my academic writing English skills.

It is my honour to have Dr. Richard as an independent committee member of my PhD work, from the go/no-go meeting to yearly progress meetings. Your advice and comments have made my work more convincing from the view of point for engineering applications. I appreciated a lot for your help with the development of adjoint boundary conditions in OpenFOAM. Thanks for Dr. Sander’s help in setting up a cluster account and discussions on using OpenFOAM. It was also inspiring to receive feedback from Fulvio and Marc during Aero PhD Day as well as Bas’ suggestions on registration for the VKI course. Life and study in TUD would be tough without the assistance of Colette. Thank you very much for arranging the accommodation at the beginning of my PhD study, the administration for attending conferences and courses, and the interesting activities and dinners within the Aero group.

It was a great pleasure to stay with my PhD fellows. Thank Tiago and Zeno for sharing your insights about research topics. It became easy to manipulate OpenFOAM with the help of Martin, Jodi and Thomas. Thank Varun and Yuyang for playing table tennis as well as open discussion about numerical methods. It was enjoyable to have chats and beers with Beppe, Wouter, Rakesh, and Shaafi during PhD drinks. Thank Luis and Mo-

hamad for sharing your experiences as well as playing chess. Thanks for the lunch-time happiness and joys with Gabriel, Alberto, Alexander, Alessandro, Edoardo, Christoph, Constantin, Kushal, Sagar, Jane, Haris, Julio, Ata, Babak, Tyler, Kherlen. It was great to have parties and chats with Jun, Haohua, Weibo, Yi, Ming, Kaisheng, Wencan, Mengjie, Ruiying, and Renzhi.

Special thanks to my friends in the ASM groups, Lubin, Hongwei, Xi, Nan (Tao), Ran, Yuqian, Nan (Yue), Xiaopeng, and Yuzhe. I enjoyed the happy time and laughing with you guys during every talk, BBQ and dinner. Thanks to my friends, Xiaocong, Xiangcou, Na, Xuan, Jiefang, Qingqing, and Qingxi, for the enjoyable time having homemade hotpots and authentic Chinese cuisines at Delft.

Last but not least, thanks for the unconditioned support and love from my family members, my parents and my uncle. You allow me to explore the world. Thanks to my girlfriend, Yuanyuan, for your trust across continents. It is memorable to share my time with you.

September 2022

6

CURRICULUM VITÆ

Xiaodong LI

02-1991 Born in Weifang, China.

EDUCATION

2010–2014 Bachelor in Aircraft Design and Engineering
Northwestern Polytechnical University

2014–2017 Master in Fluid Mechanics
Northwestern Polytechnical University

2017–2022 PhD. Aerodynamics
Delft University of Technology
Thesis: Towards efficient adjoint-based mesh optimization for predictive Large Eddy Simulation
Promotor: Prof. dr. S. Hickel

AWARDS

2022 ECCOMAS scholarship

LIST OF PUBLICATIONS

JOURNAL ARTICLES

2. **X. Li, S. Hulshoff, S. Hickel**, *An enhanced algorithm for online Proper Orthogonal Decomposition and its parallelization for unsteady simulations*, Computers and Mathematics with Applications, **126** 43–59 (2022).

1. **X. Li, S. Hulshoff, S. Hickel**, *Towards adjoint-based mesh refinement for Large Eddy Simulation using reduced-order primal solutions: preliminary 1D Burgers study*, Computer Methods in Applied Mechanics and Engineering, **379** 113733 (2021).

CONFERENCE PROCEEDINGS AND PRESENTATIONS

4. **X. Li, S. Hulshoff, S. Hickel**, *A strategy to optimal block-incremental singular value decomposition for unsteady high-fidelity simulation data*, ECCOMAS Congress 2022, Oslo, Norway, June 5-9, 2022.

3. **X. Li, S. Hulshoff, S. Hickel**, *Mesh Adaptation Using Adjoint Methods and Reduced-Order Models for Large Eddy Simulation*, WCCM–ECCOMAS 2020, **800** 1-11 (2021). (online)

2. **X. Li, S. Hulshoff, S. Hickel**, *Online Reduced-order Model and its Application on Adjoint-based Mesh Adaptation for Large Eddy Simulation*, International Congress of Theoretical and Applied Mechanics, August, 2021. (online)

1. **X. Li, S. Hickel, S. Hulshoff**, *Posteriori error estimation in unsteady Burgers problem based on Adjoint and variational multiscale method*, JMBC Burgers Symposium, May 23, 2019, Lunteren, Netherlands.