RSVD-Δt
A Time-Stepping Algorithm for Resolvent and Harmonic Resolvent Analyses of Large-Scale Flows

by

Ali Farghadan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2024

Ali Farghadan

aliii@umich.edu

ORCID iD: 0000-0002-0327-8222

DEDICATION

To the ones I love most — my mother, father, and brother

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my advisor, Aaron Towne, for his invaluable guidance, expertise, and unwavering support throughout my PhD journey. His dedication and belief in me have made a significant impact on my success. Aaron has been more than an advisor; he has been a true friend, and our friendship has kept me motivated and made my journey enjoyable.

I am deeply grateful to my family, including my parents, Katy and Bahram, and my brother, Mohammad, for their unconditional love, encouragement, and sacrifices. Their presence in my life has been a constant source of strength and motivation.

I would also like to thank my friends, including Peter, Junoh, and Rutvij, for their camaraderie and support. Their presence has made the journey more enjoyable and memorable. I have attended several conferences with Peter, and I could not have asked for a better companion for those experiences. He is truly a smart and well-spoken person. I am grateful for all the friendships that have made living in Ann Arbor a more enriching experience for me.

Additionally, I am thankful for the insightful feedback and contributions from our collaborators, including Eduardo at EMSMA, Andre at ITA, and Daniel and Petronio at Monash University. Their input has enriched my research and made this thesis possible.

I would also like to acknowledge the people at the NCRB gym, where I spent almost every day working out. The great people, staff, and environment have been a vital part of my routine. GO BLUE!

Lastly, I acknowledge MICDE and AFOSR for the funding throughout my research, and the Great Lakes cluster for providing the necessary resources and facilities, which have enabled me to make an impact in my field.

PREFACE

This thesis is designed to motivate the study of resolvent and harmonic resolvent analyses for large-scale flows and to provide comprehensive details on computing the modes efficiently. The final chapter is dedicated to explaining how to acquire, install, and utilize our open-source software for your own projects. I hope this thesis is both informative and enjoyable for those interested in this field.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

BPOD Balanced Proper Orthogonal Decomposition

DNS Direct Numerical Simulation

DMD Dynamic Mode Decomposition

ERD Empirical Resolvent Decomposition

HPC High Performance Computing

KH Kelvin-Helmholtz

LES Large Eddy Simulation

LNS Linearized Navier Stokes

LST Linear Stability Theory

OWNS One-Way Navier-Stokes

PIV Particle Image Velocimetry

POD Proper Orthogonal Decomposition

PSD Power Spectral Density

PSE Parabolized Stability Equations

RK4 4^{th} order Runge–Kutta

ROMs Reduced-Order Models

RSVD Randomized Singular Value Decomposition

SPOD Spectral Proper Orthogonal Decomposition

SVD Singular Value Decomposition

VRA Variational Resolvent Analysis

ABSTRACT

This thesis introduces a novel approach to resolvent analysis, a powerful framework for understanding coherent structures in turbulent flows. Resolvent analysis characterizes input-output relationships within fluid systems, making it possible to identify the most energetic structures and devise effective control strategies. Despite its potential, the application of resolvent analysis to complex flows has been limited by the high computational cost associated with computing resolvent modes, particularly as the problem dimension increases. To overcome this limitation, we have developed an innovative algorithm that combines randomized singular value decomposition with an optimized time-stepping method to efficiently compute the action of the resolvent operator.

The RSVD-Δt algorithm dramatically reduces the computational demands of resolvent analysis, achieving linear scaling in both CPU and memory requirements relative to the problem size. This advancement opens the door for applying resolvent analysis to large-scale systems such as inherently three-dimensional jets, airfoils, and other complex flows. We further refine strategies to minimize computational expenses and control errors, thereby ensuring that our results remain both accurate and reliable. Beyond the standard resolvent analysis, we extend the capabilities of the RSVD-Δt algorithm to handle harmonic resolvent analysis for time-periodic flows. This extension enables a comprehensive characterization of linearized dynamics in the frequency domain, addressing challenges like the coupling of retained frequencies and the singular nature of the harmonic resolvent operator. We demonstrate the validity of this extension through the analysis of a periodically varying Ginzburg-Landau equation, illustrating its potential for studying time-periodic flows.

The RSVD-Δt algorithm's robustness and efficiency are validated through several case studies. Initially, we apply the algorithm to resolvent analysis of the Ginzburg-Landau system, which validates RSVD-Δt against RSVD-LU. We demonstrate its linear scaling and performance by using a three-dimensional round turbulent jet. For harmonic resolvent analysis, RSVD-Δt is validated against RSVD-LU on a periodic Ginzburg-Landau system. We then evaluate its memory and CPU performance on flow over an airfoil. With the tool fully validated, we apply it to a three-dimensional jet with streaks, observing significant gain increases at lower frequencies due to the interaction of KH modes with the streaks. The second application involves analyzing a three-dimensional twin jet, focusing on the sensitivity of gains to nozzle-to-nozzle spacing. Lastly, we conduct harmonic resolvent analysis using RSVD-Δt to investigate the modes of an axisymmetric screeching jet, which reveals the mechanisms underlying the screeching phenomenon.

Our RSVD-Δt algorithm has been meticulously implemented using the powerful PETSc and SLEPc libraries, ensuring seamless scalability and exceptional efficiency. The code is fully open-source (GitHub), providing the scientific community with a versatile and freely accessible tool for both resolvent and harmonic resolvent analysis. By making our code available, we aim to foster innovation in fluid dynamics research, inviting researchers to modify, extend, and adapt our algorithm to meet their specific needs, and thereby creating a collaborative environment that accelerates scientific discovery.

While this thesis focuses on applications to turbulent jets, airfoil flows, and the Ginzburg-Landau problem, the algorithm's scalability and accuracy make it an ideal tool for exploring other turbulent phenomena, such as twin jets, elliptical jets, swept wings, and even full vehicle geometries. This work not only advances the field of resolvent analysis by introducing a more efficient and accurate method for computing resolvent and harmonic resolvent modes but also lays the groundwork for future research in fluid dynamics and related disciplines.

CHAPTER 1

Introduction

1.1 Background and motivation

Fluid dynamics plays a pivotal role in numerous scientific and engineering disciplines, from aerospace design to the study of natural systems such as oceans and atmospheric phenomena. Central to this field are the Navier-Stokes equations, which govern the conservation of momentum in a fluid. Understanding fluid flow behavior, especially in turbulent regimes, remains one of the most challenging problems in classical physics due to its inherent complexity and chaotic nature. Turbulence was first observed by Reynolds [117], where the nondimensionalized number named after him—known as the Reynolds number—serves as an indicator of the transition from laminar to turbulent flow. This number, defined as the ratio of inertial forces to viscous forces, indicates that low Reynolds numbers correspond to laminar flow, while higher values can lead to turbulence. Although an analytical solution to the Navier-Stokes equations without simplification has not been found, many numerical schemes have been developed to advance the understanding of fluid flows, specifically turbulent flows, which remain a top challenge.

Turbulent flows are characterized by chaotic and disorganized motions, but recurring dominant patterns can play a significant role in laminar to turbulent transition [134] and sustaining turbulence [94]. Despite substantial advancements in computational power, turbulence continues to be one of the most challenging problems in fluid dynamics due to its inherently chaotic and multi-scale nature. Direct Numerical Simulation (DNS) have made significant progress but are limited to low Reynolds numbers and canonical flows due to their high computational costs [97, 64]. Large Eddy Simulation (LES) offer a compromise by resolving the large scales of motion while modeling the smaller scales, but they are still computationally intensive and often require empirical tuning and modeling closures, which can limit their accuracy and general applicability [127]. As a result, the field shows a substantial interest to Reduced-Order Models (ROMs) that aim to capture the dominant features of turbulent flows without the need for exhaustive simulations. These models can be categorized into data-driven and operator-based approaches. Data-driven models leverage existing simulation and experimental data to identify patterns [110, 149, 77, 15], while operator-based models focus on deriving approximate solutions to the governing equations.

Model reduction plays a critical role in the study of fluid mechanics due to the complex and high-dimensional nature of fluid flows, especially turbulent ones. In particular, modal decompositions, both data-driven and equation-based, have proven to be effective at identifying low-dimensional sets of modes associated with interpretable coherent structures that significantly influence quantities of engineering interest such as kinetic energy, heat and mass transfer, and noise emissions [149, 160]. This thesis employs resolvent and harmonic resolvent analyses to identify and focus on key coherent structures within turbulent flows. The primary objective is to develop a practical framework for analyzing large-scale turbulent flows using resolvent and harmonic resolvent analysis tools, addressing the scalability challenges of existing algorithms.

1.2 Modal analysis

In the realm of fluid dynamics, as well as other scientific and engineering fields, the complexities introduced by nonlinearity are often addressed by analyzing a linearized version of the governing equations. This linearization is typically achieved by expanding the equations around a known reference state and focusing on the leading-order terms [134, 31]. Adopting linear analysis techniques facilitates insight into the fluid's behavior without grappling with the full complexity of the original nonlinear equations.

Consider the incompressible Navier-Stokes equations in their dimensional form,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{u}, \quad \nabla \cdot \boldsymbol{u} = 0, \tag{1.1}$$

where $\boldsymbol{u}$ is the velocity field, p is the pressure, and Re is the Reynolds number.

By linearizing around a base flow $\boldsymbol{U}$, we obtain

$$\frac{\partial \boldsymbol{u}'}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{u}' + \boldsymbol{u}' \cdot \nabla \boldsymbol{U} = -\nabla p' + \frac{1}{Re}\nabla^2 \boldsymbol{u}' + \boldsymbol{f}. \tag{1.2}$$

Here, $\boldsymbol{u}'$ and p' are perturbations around the base flow, and $\boldsymbol{f}$ contains the remaining nonlinear terms that are interpreted as a forcing term. The linearized operator $\mathcal{L}$ can be written as

$$\mathcal{L}(\boldsymbol{u}') = -\boldsymbol{U} \cdot \nabla \boldsymbol{u}' - \boldsymbol{u}' \cdot \nabla \boldsymbol{U} + \frac{1}{Re}\nabla^2 \boldsymbol{u}'. \tag{1.3}$$

A key component of this approach is the use of matrix decompositions of the linearized differential operator $\mathcal{L}$. These decompositions yield "modes" that can provide valuable insights into underlying physical mechanisms or act as a foundation for representing solutions in conjunction with other modeling techniques. The simplest form of matrix decomposition is eigendecomposition, which has been widely used for many linear systems. In operator-based models, the governing equations are often represented by the Linearized Navier Stokes (LNS) operator. Data-driven models, on the other hand, compute modes directly from data, which can be generated from experimental setups or numerical simulations, including DNS, LES, and others.

These modes, also known as coherent structures, can be seen as the foundational building blocks of turbulence, and modal analysis is an important tool for identifying and understanding these structures [149]. Eigenanalysis, or eigendecomposition, involves decomposing a system into its characteristic modes, which helps in understanding its fundamental behavior. Popular data-driven methods for extracting and analyzing these modes include Proper Orthogonal Decomposition (POD) [140], Balanced Proper Orthogonal Decomposition (BPOD) [168, 80], Dynamic Mode Decomposition (DMD) [132], and Spectral Proper Orthogonal Decomposition (SPOD) [87, 160]. Each of these methods utilizes Singular Value Decomposition (SVD) as a core computational tool.

- **POD** decomposes a dataset into orthogonal modes based on energy content, which can be computed from the SVD of a data matrix containing snapshots of the flow variables. By selecting the modes with the highest singular values, POD captures the most energetic spatial structures in the flow.

- **BPOD** is an extension of POD that takes into account the system's controllability/observability. It uses SVD to identify modes that are most significant not only in terms of energy but also in terms of the system's input-output behavior. BPOD is particularly useful in control applications where understanding how the system responds to inputs is crucial.

- **DMD** seeks to identify modes that describe the dynamic behavior of a system by approximating the eigenvalues and eigenvectors of an assumed linear operator governing the dynamics. SVD is also used to compute the pseudo inverse used to approximate the linear operator.

- **SPOD** identifies modes that are coherent in space and evolve coherently in time. These modes can be computed by taking the SVD of a data matrix containing Fourier modes extracted from flow data, and each mode represents a spatio-temporally coherent structure within the flow.

In the context of low-rank systems, the application of SVD becomes particularly advantageous. Many fluid dynamics systems exhibit low-rank characteristics, meaning that a relatively small number of modes contribute significantly to the system's dynamics [61, 149]. The leading modes can be interpreted with greater clarity and provide a more accurate representation of the underlying physical phenomena, effectively capturing the dominant patterns and structures within the fluid flow [87, 140, 149, 16]. This facilitates the development of simplified yet accurate models [124].

Modal analysis techniques offer significant advantages within a wide range of applications. For instance, in flow control, the reduced set of modes can be used to design more efficient controllers by focusing on the most influential dynamics, thereby achieving effective control with reduced computational effort [17, 91]. In aerodynamic design, these modes help streamline the optimization process by reducing the dimensionality of the design space, making it computationally feasible to explore more design variations [82]. SVD also aids in data compression by retaining the most critical information while discarding less important details, facilitating efficient storage and transmission of large datasets [38].

1.2.1 Linear stability theory

Linear Stability Theory (LST) is a fundamental tool for analyzing the stability of fluid flows under the assumption of small perturbations. It assumes that perturbations evolve according to the linearized governing equations, transforming the problem into an eigenvalue analysis of a linear differential operator [70, 105, 143]. The stability of the flow is determined by the real parts of these eigenvalues: if all eigenvalues have negative real parts, the perturbations decay exponentially, and the flow is considered stable. On the other hand, any eigenvalue with a positive real part indicates instability, suggesting that perturbations grow exponentially and the flow may transition to turbulence [55]. LST is particularly valuable for predicting the onset of turbulence [30, 130].

Nevertheless, LST has limitations. Practical observations often reveal turbulence at Reynolds numbers significantly lower than those predicted by LST. This discrepancy arises because LST primarily accounts for long-term exponential behavior and does not capture transient growth caused by non-normality of the eigenvectors [162, 131, 10]. Such transient growth can amplify perturbations significantly, potentially triggering nonlinear dynamics that lead to turbulence. Understanding these limitations is crucial for developing more comprehensive models of fluid flow stability and transition.

1.2.2 Resolvent analysis

Resolvent (or input-output) analysis originates from classical control theory [33, 76] and has become arguably the most important operator-theoretic modal decomposition techniques in fluid mechanics [95, 149, 72, 123]. Resolvent analysis has been applied to a wide variety of flows, including canonical wall-bounded flows [29, 99], turbulent jets [66, 137, 83, 111], and airfoils [153, 170]. It has been used for diverse tasks including design optimization [21, 116], receptivity analysis [74, 25], and flow control [171, 157, 90, 91]. SVD of the resolvent operator is at the heart of input-output-based studies. The left singular vectors of the resolvent operator, known as the response modes, are often related to the coherent motions in the flow [95, 160]. Specifically, the resolvent modes associated with the largest singular values provide an approximation of the leading SPOD modes [160] and, in some cases, capture the majority of the Power Spectral Density (PSD) of the flow [148]. The right singular vectors, also known as the forcing modes, describe the optimal inputs that lead to the most amplified responses, characterized by the largest singular values (gains), and offer information about the mechanisms driving these responses.

Resolvent analysis can be computationally demanding. Two steps constitute most of the cost: (i) forming the resolvent operator, which involves computing an inverse, and (ii) performing the SVD. Both steps nominally scale like $O(N^3)$, where N is the state dimension. State-of-the-art methods, described below, improve on this scaling, but the computational cost remains a strong function of the state dimension N. The state dimension, in turn, depends acutely on the number of spatial dimensions that must be numerically discretized. While the linearized Navier–Stokes equations are nominally three-dimensional, they can be simplified by expanding the flow variables into Fourier modes in homogenous dimensions, *i.e.*, those in which the base flow about which the equations are linearized does not vary. This markedly reduces the size of the discretized operators that must be manipulated, decreasing the computational cost. Accordingly, inherently three-dimensional flows that do not contain homogeneous directions or other simplifying symmetries are particularly challenging.

Recent advancements aim to overcome these two computational bottlenecks. The second bottleneck can be alleviated by using efficient algorithms to compute only the SVD modes with the largest singular values, which are typically of primary interest, rather than the complete decomposition. Standard methods like power iteration and various Arnoldi methods have been frequently applied for this purpose. More recently, Randomized Singular Value Decomposition (RSVD) [53] has been shown to further reduce the cost of resolvent analysis of one- [96] and two-dimensional [118] problems.

Regarding the first bottleneck, forming the resolvent operator by computing an inverse is feasible only for small systems, *e.g.*, one-dimensional ones. Fortunately, the aforementioned

SVD algorithms do not require direct access to the resolvent operator, but rather its action on a specified forcing vector, *i.e.*, the result of applying the resolvent operator to that vector. Accordingly, we can recast the first bottleneck in terms of the computational cost of computing the action of the resolvent operator on a vector. The standard approach for doing so is to solve a linear system whose solution yields the action of the resolvent operator on the right-hand-side vector via LU decomposition of the inverse of the resolvent operator (which can be directly formed in terms of the linearized Navier-Stokes operator; see §2.3 for details). The computational cost of this approach typically scales like $O(N^{1.5})$ or $O(N^2)$ for two- and three-dimensional problems, respectively, which is tolerable for most two-dimensional problems but quickly becomes intractable for three-dimensional problems. Numerous authors have used this LU-based approach along with Arnoldi methods [139, 66, 137, 75]. Brynjell-Rahkola *et al.* [18] used LU decomposition along with a power iteration with a Laplace preconditioner to increase the convergence rate of the resolvent modes. Ribeiro *et al.* [118] used LU decomposition along with RSVD, which we call "RSVD-LU" in this thesis, and demonstrated significant CPU savings compared to using an Arnoldi iteration. However, the poor cost scaling of the LU decomposition with problem size N remains a limiting factor, impeding the investigation of three-dimensional flows and other large systems. Recently, Houtman *et al.* [63] used an iterative solver as an alternative to LU decomposition to handle large systems. While this approach is attractive due to its potential to achieve $O(N)$ scaling, the absolute cost of iterative solvers depends heavily on the availability of an effective preconditioner, which is typically problem dependent, especially for the ill-conditioned matrices obtained from the linearized Navier-Stokes equations.

Resolvent modes can be computed at a reduced cost for slowly varying flows, *i.e.*, flows whose mean changes gradually in some spatial direction, by using spatial marching methods to approximate the action of the resolvent operator. Spatial marching methods approximately evolve perturbations in the slowly varying direction. The best-known spatial marching method is the Parabolized Stability Equations (PSE), but the inherent ill-posedness of PSE [84] requires deleterious regularization that makes it ill-suited to compute resolvent modes in most cases [158]. One exception is very low frequencies, where PSE has been used to compute resolvent modes corresponding to boundary-layer streaks [128]. The One-Way Navier-Stokes (OWNS) equations [155] overcome many of the limitations of PSE; they are formally well-posed and capture the complete downstream response of the flow. The original formulation did not include a right-hand-side forcing on the linearized equations, which is fundamental to resolvent analysis. This was addressed by a second OWNS variant formulated in terms of a projection operator that splits both the solution and forcing into upstream- and downstream-traveling components [159]. This method has been combined with a power-iteration approach to accurately and efficiently approximate resolvent modes for a range of slowly varying flows ranging from incompressible boundary layers to supersonic jets to hypersonic boundary layers. Recently, the cost of this approach was further reduced by a new recursive OWNS formulation [173]. The fundamental limitation of OWNS-based approaches is their restriction to (mostly) canonical flows that contain a slowly varying direction.

Several data-driven methods for computing resolvent modes have been proposed, which avoid working directly with the resolvent operator at all. Towne *et al.* [156] and Towne [154] introduced Empirical Resolvent Decomposition (ERD). Starting with data in the form of a set of forcing and response pairs, ERD solves an optimization problem to identify modes within the span of the data that maximizes the gain. Another recent approach uses DMD [132] to estimate the resolvent modes from data [59]. This approach benefits from the advancements in DMD [133] and is robust, but to accurately approximate the resolvent modes, many random initial conditions may need to be simulated.

Barthel *et al.* [11] recently proposed a reformulation of resolvent analysis called Variational Resolvent Analysis (VRA). Using the same mathematics that underly ERD, VRA computes resolvent modes by solving a Rayleigh quotient, avoiding the inverse that appears in the definition of the resolvent operator. To make the method computationally advantageous, the response modes are constrained to lie within the span of some other reduced-order basis. Barthel *et al.* [11] obtain this basis from a series of locally parallel resolvent analyses; if the basis is taken from data, VRA becomes ERD. VRA showed speed-up compared to standard approaches for a canonical boundary layer, but it remains to be investigated for more complex scenarios where an effective basis is not evident.

Time-stepping methods offer an alternative approach to overcome the first bottleneck (these methods are sometimes referred to as "matrix-free" approaches, as forming the LNS operator is not necessary). The central idea is to obtain the action of the resolvent operator on a vector by solving the linearized equations in the time domain. A pioneering study by Monokrousos *et al.* [98] used time stepping along with power iteration to compute resolvent modes for a flat-plate boundary-layer flow. Modes at a particular frequency of interest were computed by forcing the linearized equations exclusively at that frequency and time stepping until a steady-state solution is obtained. Gómez *et al.* [50] proposed an iterative procedure for updating the initial conditions to reduce the time required to reach the steady-state solution. This resulted in an 80% reduction of CPU time for a test problem, but only the leading mode at each frequency was obtained. Martini *et al.* [92] introduced two additional variations of time-stepping approaches for computing resolvent modes with improved efficiency. The first, referred to as the transient response method, evaluates the transitional response of the LNS to temporally compact forcing. The second variation, known as the steady-state response method, computes the steady-state solution of the LNS when it is forced with a set of harmonic frequencies. Both methods allow all frequencies of interest to be simultaneously computed by isolating each frequency in the flow response using a discrete Fourier transform. Additionally, the steady-state method can be easily paired with more advanced SVD algorithms (*e.g.*, Arnoldi, rather than power iteration) to obtain multiple resolvent modes at each frequency.

Time-stepping methods for computing resolvent modes are potentially powerful because they obtain the action of the resolvent operator without the need for inverses or LU decomposition. Indeed, we will show that time time-stepping methods can achieve linear cost scaling with the problem dimension N. However, achieving this potential and overall low CPU and memory costs requires careful consideration of numerous factors.

In this thesis, we present a novel approach, abbreviated as "RSVD-Δt", that combines the benefits of RSVD with the advantages of time stepping. In short, the method eliminates the bottleneck in the RSVD-LU approach created by the LU decomposition by obtaining the action of the resolvent operator via an optimized time-stepping approach. All frequencies of interest as computed simultaneously using a steady-state response approach as in Martini *et al.* [92]. Additionally, we develop a novel technique to remove the undesired transient component of the response, shortening the temporal interval over which the equations are integrated and reducing the CPU cost by an order of magnitude in most cases. To minimize memory usage, we utilize streaming calculations for transferring data between the Fourier and time domains. The RSVD-Δt algorithm is shown to exhibit linear scalability both in terms of computational complexity and memory requirements and can be efficiently parallelized. Overall, these capabilities allow us to compute resolvent modes for three-dimensional flows and other large systems that were previously out of reach.

1.2.3 Harmonic resolvent analysis

Resolvent analysis has been extensively employed to comprehend, model, and control statistically stationary turbulent flows such as wall-bounded turbulence [29], turbulent boundary layer [139], and jet flows [113]. However, real-world fluid systems often exhibit non-stationary behavior, including periodicity, rendering resolvent analysis less effective. Examples of flows exhibiting periodic behavior include internal waves in a stratified fluid [163], vortex shedding in the wake of a bluff body [48], and pulsatile blood flow [42]. Recently, an extension of resolvent analysis, known as harmonic resolvent analysis, has been developed by Padovan *et al.* [106]. This extension characterizes the perturbation forcing and responses around a periodic base flow, providing a more effective tool for analyzing, modeling, and controlling periodically time-varying flows. The harmonic resolvent operator can be considered a special case of the harmonic transfer function introduced by Wereley [167], wherein the operator exhibits a dominant frequency.

Investigating the harmonic resolvent operator is one way to identify the optimal spatio-temporal perturbations and their corresponding responses in fluid systems. Unlike the resolvent operator, the harmonic resolvent operator takes into consideration the periodicity of the base flow by representing it in the form of a Fourier series. This allows for characterization of the dynamic behavior of the system, particularly with respect to its spectral components and it provides a means to understand and explore the interactions between input and output modes through the linearized dynamics. Harmonic resolvent analysis determines the triadic interactions between the input frequency ω_1, the base flow frequency $\omega_1 - \omega_2$, and the response frequency ω_2. The number of Fourier modes of the base flow is determined by its power spectral density (PSD), which in turn determines the number of triads interacting with the forcing. Typically, periodic flows with a dominant periodicity exhibit a multimodal nature with their most energetic modes being harmonics of the base frequency. When the base flow is statistically stationary, linearization around the temporal average is sufficient, and traditional resolvent analysis arises as a special case.

Recent studies have expanded on this research area to model and control the dynamics of turbulent flows with periodic motions. Lin *et al.* [85] applied the harmonic resolvent framework for linear time-periodic systems with more than one dominant frequency, offering a tool to study the flow control of a plunging cylinder. In a distinct approach, Franceschini *et al.* [46] proposed a novel method aimed at capturing the phase dependency of small-scale turbulent phenomena in flows exhibiting a periodic limit cycle. This method builds upon classical SPOD and resolvent analyses but introduces a quasi-steady approximation that effectively separates the high-frequency turbulent component from the slow periodic oscillation. Moreover, Heidt & Colonius [54] have pioneered the theory and algorithm for cyclostationary SPOD (CS-SPOD). This innovative approach extends the conventional SPOD algorithm to effectively capture the characteristics of flows characterized by periodic motions. CS-SPOD is closely tied to harmonic resolvent analysis, akin to the relationship between SPOD and resolvent analysis [160].

Harmonic resolvent analysis offers a promising avenue for new discoveries in physics. For example, recent research conducted by Wu *et al.* [169] explored the response of a turbulent separation bubble to zero-net-mass-flux actuation, with a base flow characterized by periodicity. The noteworthy finding revealed an agreement between the reduction in separation bubble size identified through both harmonic resolvent optimal response and direct numerical simulation. Another contribution in this realm comes from the work of Padovan & Rowley

[107], who investigated the driving mechanisms underlying vortex pairing in an incompressible forced axisymmetric jet. They showed that the optimal forcing of the subharmonic resolvent operator featured spatiotemporal structures leading to the nonlinear vortex pairing phenomenon. In general, harmonic resolvent analysis can facilitate the understanding of energy transfer mechanisms [69] and is a potentially powerful tool for secondary stability analyses involving periodic additions to the base flow, with applications such as boundary layer stability and transition [57], screeching jets [36], and the effect of mass injection on secondary instability [78].

From a computational standpoint, computing harmonic resolvent modes presents similar challenges to resolvent analysis, including the need to invert sparse matrices in Fourier space. This challenge is further exacerbated in harmonic resolvent analysis due to expanded system size, where all relevant frequencies are merged into a unified operator, in contrast to resolvent analysis, where each frequency has its own independent system. The second shared challenge involves performing the SVD. Turbulent and other complex flows characterized by periodic motions require high-resolution meshes and potentially large computational domains. This results in a sizable linearized operator, introducing formidable computational challenges when it comes to matrix inversion and performing SVD. RSVD-LU can be used to simultaneously overcome both computational challenges [96, 118, 106]. Nevertheless, it becomes impractical for tackling large-scale turbulent flows since it requires the lower-upper (LU) decomposition of the harmonic resolvent operator. Our algorithm, RSVD-Δt, an improvement to RSVD-LU, addresses bottlenecks in resolvent analysis [44]. The approach of time-stepping, initially developed by Monokrousos *et al.* [98] and further optimized by Martini *et al.* [92], allows for computing the action of an operator on a vector without solving a linear system in Fourier space, making it a key component of RSVD-Δt.

One key objective of this thesis is to expand the applicability of the RSVD-Δt algorithm, initially crafted for resolvent analysis, to facilitate the computation of harmonic resolvent modes. A streaming approach is adopted for time integration, reducing memory usage while maintaining computational efficiency. Moreover, an efficient transient removal strategy is introduced, cutting the temporal length of the integration and thereby minimizing the CPU cost of RSVD-Δt for achieving a desired accuracy. The RSVD-Δt algorithm for periodic flows maintains linear scalability in both CPU time and memory consumption. This scalability enables the efficient computation of harmonic resolvent modes for large-scale turbulent flows that were previously out of reach.

1.3 Thesis overview

This thesis is organized into six chapters, each addressing distinct aspects of resolvent analysis, harmonic resolvent analysis, their applications to large-scale flows, and the development of an accompanying open-source package.

1.3.1 Key contributions

- **Development of the RSVD-Δt algorithm:** This thesis presents a novel algorithm for the efficient, massively parallel computation of resolvent modes, offering significant improvements in scalability over existing methods while maintaining accuracy. The algorithm possesses theoretical $O(N)$ scaling, which has been numerically validated in its application to three-dimensional flows. This results in a substantial reduction in computational resources, particularly in memory consumption, which also scales linearly with the problem dimensions. This feature makes the algorithm suitable for problems that would otherwise require enormous memory using other methods. Importantly, the RSVD-Δt algorithm does not rely on specific flow conditions or simplifications; it computes the global resolvent modes for any generic problem setup. It does not require pre-training or prior knowledge of the flow field, effectively reproducing what the RSVD-LU algorithm can compute.

- **Optimization of the RSVD-Δt algorithm:** Achieving $O(N)$ scaling in both memory usage and CPU time is a significant accomplishment, but further enhancements are needed to reduce the absolute computational cost for large-scale systems. This can be achieved by introducing techniques that lower simulation costs or minimize memory usage for a given problem.

  To minimize memory consumption, we leverage streaming calculations and, for real-valued problems, the positive/negative symmetry in the frequency domain. To minimize CPU time, our focus was on accelerating the removal of transients (undesirable parts of time-stepping) much faster than natural decay. We developed a novel strategy that removes transients much more quickly—saving up to two orders of magnitude in CPU cost—by exploiting the differing evolution of the transient and steady-state parts of the snapshots.

- **Extension to harmonic resolvent analysis:** We initially developed the algorithm to compute global resolvent modes. Since it efficiently computes the action of the harmonic resolvent operator on a vector, we extended it to handle flows with periodic motions. Harmonic resolvent modes are N_ω times larger in size compared to the same setup without periodicity, where N_ω represents the number of harmonics retained. The increased matrix size and computational costs, driven by triadic interactions and the need to handle near-singularities, are effectively managed by our algorithm, which maintains linear scaling with problem dimensions. Specifically, it scales as $O(NN_\omega) + O(NN_b)$, where N_b is the number of frequencies retained to expand the base flow.

- **Application to previously intractable flows:** The development and validation of our tool required significant time and effort, but it has resulted in a powerful instrument capable of analyzing three-dimensional flows or even large two-dimensional flows in the context of harmonic resolvent analysis. In this thesis, we present a preliminary analysis of streaks in jets, demonstrating how streaks can influence the gains and Kelvin-Helmholtz (KH) wavepackets within a turbulent jet. In addition, we apply RSVD-Δt to a twin jet flow, focusing on the impact of nozzle-to-nozzle spacing on the resolvent gains. We show that the gain is especially sensitive to the nozzle spacing for low-frequency modes with a particular symmetry, consistent with and augmenting previous two-dimensional eigenvalue analyses of twin jets.

  Finally, we conduct a thorough investigation of a screeching jet flow using harmonic resolvent analysis. Screech is a high-pitched, resonant noise generated by the interaction between shock waves and turbulence in supersonic jets. We begin by identifying the screech mode through resolvent analysis of a shock-containing jet, and then add this mode to the mean flow to produce the periodic base flow for harmonic resolvent analysis. This analysis sheds light on how the screech mode can influence the KH wavepacket and guided jet mode. We delve further into the nonlinear interactions between the screech mode and shock cells, which generate higher harmonic responses and acoustic waves—an effect observed in experiments, though its underlying mechanism was previously unclear. Lastly, intriguing observations reveal that the wavenumber spectra show the guided jet wavenumber as the difference between the shock cell and KH wavenumbers. Nonlinear forcing and responses also exhibit wavenumbers corresponding to shock cell spacing, which warrants further investigation.

- **Open-source implementation:** While many modal analysis techniques are straightforward to implement in MATLAB or other software, our algorithm requires careful implementation. The steps of the algorithm are outlined in this thesis. We chose the PETSc/SLEPc libraries to handle the algebra within RSVD-Δt due to their comprehensive capabilities and open-source nature. Therefore, we are excited to share our code as an open-source software package, minimizing the effort required for those interested in using RSVD-Δt. We have dedicated a full chapter to help users get started with our software, providing detailed instructions on installation and input variables.

We are enthusiastic about contributing to the research community and hope that by sharing our code, we can push the boundaries of what's possible in resolvent and harmonic resolvent analysis. We invite you to explore our work, use our tools, and join us in advancing this exciting field.

1.3.2 Roadmap

The organization of the thesis is as follows:

- **Chapter 1: Introduction**
  - Provides a brief overview of turbulent flows, the linearized Navier-Stokes equations, and the challenges associated with modeling fluid dynamics, with a focus on modal analysis techniques.
- **Chapter 2: Resolvent Analysis**
  - Reviews the principles of resolvent analysis, including its formulation and various computational methods.
  - Introduces the RSVD-Δt algorithm, with detailed discussions on computational complexity and optimization.
  - Validates the algorithm using the Ginzburg-Landau system and a three-dimensional round turbulent jet, and includes scaling results illustrating performance in terms of CPU time and memory usage.
- **Chapter 3: Harmonic Resolvent Analysis**
  - Presents harmonic resolvent analysis, covering its formulation, challenges like singularities, truncation methods, and variations such as cross-frequency and subharmonic resolvent analysis.
  - Reviews the RSVD-LU method as the original approach for computing harmonic resolvent modes.
  - Extends the RSVD-Δt algorithm to enhance the efficiency of computing these modes.
  - Validates the extended algorithm on periodic Ginzburg-Landau systems and flow over an airfoil, including empirical scaling plots for performance in terms of CPU time and memory usage.

- **Chapter 4: Applications**
  - Uses the RSVD-Δt algorithm to apply resolvent and harmonic resolvent analyses to previously out-of-reach, three-dimensional problems
  - Studies the impact of streaks on other coherent structures within a turbulent jet.
  - Examines the influence of nozzle-to-nozzle spacing on KH wavepackets within a twin round turbulent jet.
  - Investigates the underlying physics and nonlinear mechanisms critical to understanding screeching jet phenomena.
- **Chapter 5: RSVD-Δt User Manual**
  - Provides a comprehensive guide to using the RSVD-Δt software, including installation instructions, input specifications, and example test cases.
- **Chapter 6: Conclusion**
  - Summarizes the key findings, discusses new applications, and suggests directions for future research.

CHAPTER 2

Resolvent Analysis

Resolvent analysis is a powerful tool for understanding the dynamics of fluid flows, particularly in the context of turbulent flows and flow control. It involves decomposing the nonlinear Navier-Stokes equations into a linear operator, which represents the dynamics of small perturbations around a mean flow, and a nonlinear term, which captures the interactions between these perturbations. By analyzing the resolvent operator, the most amplified perturbations, known as resolvent modes, can be identified, which correspond to the most energetic structures in the flow. These modes can be used to understand the underlying mechanisms driving flow phenomena, such as turbulence, and to design effective control strategies to manipulate the flow. In essence, resolvent analysis provides a framework for characterizing the input-output relationships in fluid flows, allowing us to pinpoint the most influential factors shaping the flow behavior.

In this chapter, we delve into the concept of resolvent analysis, examining its formulation and the latest algorithms for computing resolvent modes. We introduce our algorithm, RSVD-Δt, and validate its performance against ground truth results. We provide a comprehensive analysis of the algorithm's efficiency, focusing on both CPU time and memory usage. To demonstrate scalability, we present empirical scaling results based on a three-dimensional flow, specifically computing the resolvent modes for a three-dimensional round jet. Our algorithm is designed to minimize memory usage and optimize simulation length while preserving accuracy, enhancing its effectiveness and practicality in handling large-scale flows.

2.1 Formulation

Our starting point is the compressible Navier-Stokes equations written as

$$\frac{\partial \boldsymbol{q}}{\partial t} = \boldsymbol{\mathcal{N}}(\boldsymbol{q}), \tag{2.1}$$

where the nonlinear Navier-Stokes operator $\boldsymbol{\mathcal{N}}$ acts on the state vector $\boldsymbol{q} \in \mathbb{C}^N$, which describes the flow discretized in all inhomogeneous directions. A standard Reynolds decomposition

$$\boldsymbol{q}(\boldsymbol{x},t) = \bar{\boldsymbol{q}}(\boldsymbol{x}) + \boldsymbol{q}'(\boldsymbol{x},t) \tag{2.2}$$

partitions the flow state into the time-averaged mean $\bar{\boldsymbol{q}}$ and the fluctuation $\boldsymbol{q'}$. Substituting (2.2) into (2.1) yields

$$\begin{aligned}\frac{\partial \boldsymbol{q}'}{\partial t} &= \boldsymbol{A}(\bar{\boldsymbol{q}})\boldsymbol{q}' + \boldsymbol{B}\boldsymbol{f}'(\bar{\boldsymbol{q}},\boldsymbol{q}'),\\ \boldsymbol{y}' &= \boldsymbol{C}\boldsymbol{q}',\end{aligned} \tag{2.3}$$

where $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ is the LNS operator, $\boldsymbol{B} \in \mathbb{C}^{N\times N_f}$ is an input matrix that can be used to restrict the forcing $\boldsymbol{f'} \in \mathbb{C}^{N_f}$, and $\boldsymbol{C} \in \mathbb{C}^{N_y\times N}$ is an output matrix that extracts the output of interest $\boldsymbol{y'} \in \mathbb{C}^{N_y}$ from the state. The forcing $\boldsymbol{f'}$ can represent an exogenous forcing and/or the nonlinear perturbation terms from the Navier-Stokes equations.

Resolvent analysis is most natural when $\boldsymbol{A}$ is stable, *i.e.*, all of its eigenvalues lie in the left-half plane. If $\boldsymbol{A}$ is unstable, discounting can be used to obtain a stable system [71, 171]. We assume that, if necessary, discounting has already been performed so that $\boldsymbol{A}$ is strictly stable.

Resolvent analysis seeks the forcing that produces the largest steady-state response. Since the steady state is of interest, the solution can be obtained in the frequency domain. Taking the Fourier transform

$$\mathcal{F}(\cdot) = \hat{(\cdot)}(\omega) = \int_{-\infty}^{+\infty} (\cdot)e^{-\mathrm{i}\omega t}\,dt \tag{2.4}$$

of (2.3) and solving for the output yields

$$\hat{\boldsymbol{y}}(\omega) = \boldsymbol{R}(\omega)\hat{\boldsymbol{f}}(\omega), \tag{2.5}$$

where ω is the frequency and $\hat{(\cdot)}$ denotes the frequency counterpart of the time domain vector. The resolvent operator

$$\boldsymbol{R} = \boldsymbol{C}(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \tag{2.6}$$

maps the input forcing to the output response (here, $\mathrm{i} = \sqrt{-1}$ and $\boldsymbol{I}$ is the identity matrix.)

The optimization problem for the most amplified forcing is formally defined as maximizing

$$\sigma = \frac{\|\hat{\boldsymbol{y}}\|_q}{\|\hat{\boldsymbol{f}}\|_f} = \frac{\|\boldsymbol{R}\hat{\boldsymbol{f}}\|_q}{\|\hat{\boldsymbol{f}}\|_f}, \tag{2.7}$$

where $\|\boldsymbol{x}\|_f^2 = \langle \boldsymbol{x},\boldsymbol{x}\rangle_f = \boldsymbol{x}^*\boldsymbol{W}_f\boldsymbol{x}$ computes the f-norm of any vector $\boldsymbol{x}$ and $(\cdot)^*$ denotes the conjugate transpose. $\boldsymbol{W}_f$ is a weight matrix that accounts for numerical quadrature and allows us to define arbitrary norms. Note that input and output norms can be different, *i.e.*, $\|\cdot\|_q = \|\cdot\|_f$ is not required. For notational brevity, we assume identity matrices for the weight, input, and output matrices in what follows. The minor adjustments to our algorithm to accommodate non-identity weight, input, and output matrices are outlined in §2.5.

Solving the Rayleigh quotient (2.7) is equivalent to computing the SVD of the resolvent operator [145]

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*, \tag{2.8}$$

where $\boldsymbol{\Sigma}$ contains the singular values (a.k.a. *gains*), and $\boldsymbol{V}$ and $\boldsymbol{U}$ are right and left singular vectors corresponding to input and output vectors (a.k.a. forcing and response *modes*), respectively.

2.2 Computation

Computing resolvent modes by following the definitions from the previous § involves two computationally intensive steps: (i) forming the resolvent operator by computing the inverse in (2.6) and (ii) computing the full singular value decomposition in (2.8). Both of these steps nominally require $O(N^3)$ operations. This is workable for one-dimensional problems, *e.g.*, a channel flow [96], but quickly becomes intractable for two- and three-dimensional problems.

Instead, most applications of resolvent analysis to two-dimensional problems have adopted an alternative approach that leverages LU decomposition and iterative eigenvalue solvers [139, 66, 137, 153, 75]. This approach utilizes a mathematical equivalence to compute the resolvent modes faster than the natural approach. It is straightforward to verify that computing the right singular vectors of the resolvent operator is equivalent to computing the eigenfunctions of $\boldsymbol{R}^*\boldsymbol{R}$, *i.e.*, $\boldsymbol{R}^*\boldsymbol{R} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^*$. By computing the leading eigenmodes of $\boldsymbol{R}^*\boldsymbol{R}$, both right singular vectors and square of singular values of the resolvent operator are obtained. Recovering the left singular vectors is done via $\boldsymbol{U} = \boldsymbol{R}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$. The leading eigenvalues and eigenvectors can be efficiently computed via Arnoldi iteration [5]. The cost of the Arnoldi method relies on the desired number of modes and the convergence threshold. Computing the LU decomposition of $(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})$ circumvents computing $\boldsymbol{R}$ directly. This is a common practice to speed up the process of constructing the orthonormal basis of the Krylov subspace [152]. However, the $O(N^2)$ scaling remains poor for three-dimensional systems.

The main objective of this chapter is to enable resolvent analysis for high-dimensional systems. Therefore, we discuss state-of-the-art approaches and introduce an improved algorithm specifically designed to tackle three-dimensional flows.

2.3 Computing resolvent modes using RSVD

RSVD is a recent randomized linear algebra technique that provides a low-cost approximation of the leading singular modes of a matrix [53] by sampling its image and range. In the following two subsections, we introduce the RSVD algorithm and discuss its application to resolvent analysis.

2.3.1 RSVD algorithm

Algorithm 1 RSVD-LU

```
1: Inputs: $\boldsymbol{R}, k, q$
2: $\boldsymbol{\Theta} \leftarrow \text{randn}(N, k)$            ▷ Create random test matrices
3: $\boldsymbol{Y} \leftarrow \boldsymbol{R\Theta}$            ▷ Sample the range of $\boldsymbol{R}$
4: if $q > 0$ then            ▷ Optional power iteration
5:     $\boldsymbol{Y} \leftarrow \texttt{PI}(\boldsymbol{R}, \boldsymbol{Y}, q)$            ▷ Algorithm 2
6: $\boldsymbol{Q} \leftarrow \text{qr}(\boldsymbol{Y})$            ▷ Build the orthonormal subspace $\boldsymbol{Q}$
7: $\boldsymbol{S} \leftarrow \boldsymbol{Q}^*\boldsymbol{R}$            ▷ Sample the image of $\boldsymbol{R}$
8: $(\tilde{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \text{svd}(\boldsymbol{S})$            ▷ Obtain $\boldsymbol{\Sigma}, \boldsymbol{V}$
9: $\boldsymbol{U} \leftarrow \boldsymbol{Q}\tilde{\boldsymbol{U}}$            ▷ Recover $\boldsymbol{U}$
10: Outputs: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$
```

There exist several variations of the RSVD algorithm; here, we outline the algorithm from Halko *et al.* [53]. The first step is to sample the range of $\boldsymbol{R}$ by forming its sketch (line 3)

$$\boldsymbol{Y} = \boldsymbol{R\Theta}, \tag{2.9}$$

where $\boldsymbol{\Theta} \in \mathbb{C}^{N\times k}$ is a dense random test matrix (line 2) with $k \ll N$ columns that determines the number of leading modes to be approximated. Increasing the number of test vectors slightly beyond the desired number of modes enhances the accuracy of the leading modes. A feature of high-dimensional random vectors is that they form an orthonormal set with high probability [164], such that, on average, $\boldsymbol{\Theta}$ projects uniformly onto all of the right singular vectors of $\boldsymbol{R}$. Therefore, the sketch preserves the leading left singular vectors of $\boldsymbol{R}$. An orthonormal basis $\boldsymbol{Q} \in \mathbb{C}^{N\times k}$ for the sketch is obtained via QR decomposition (line 6), which is then used to sample the image of $\boldsymbol{R}$ (line 7) as

$$\boldsymbol{S} = \boldsymbol{Q}^*\boldsymbol{R}. \tag{2.10}$$

Computing the SVD of $\boldsymbol{S} \in \mathbb{C}^{k\times N}$ (line 8), which is inexpensive due to its reduced dimension, provides an approximation of the k leading right singular vectors $\boldsymbol{V} \in \mathbb{C}^{N\times k}$ and singular values $\boldsymbol{\Sigma} \in \mathbb{C}^{k\times k}$ of $\boldsymbol{R}$. Finally, the corresponding approximations of the left singular vectors of $\boldsymbol{R}$ can be recovered as $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}} \in \mathbb{C}^{N\times k}$ (line 9).

RSVD accurately estimates the leading modes for matrices with rapidly decaying singular values. For systems with slowly decaying singular values, performing q optional power iterations (lines 4-5 and Algorithm 2) enhances the accuracy of the estimates. The rationale of power iteration is to increase the effective gap between singular values within the sketch by exponentiating them, since

$$(\boldsymbol{R}\boldsymbol{R}^*)^q\boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}(\boldsymbol{V}^*\boldsymbol{V})\boldsymbol{\Sigma}\boldsymbol{U}^*)^q\boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^*)^q\boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}^{2q}\boldsymbol{U}^*)\boldsymbol{Y}. \tag{2.11}$$

Raising the singular values to a high power artificially accelerates the decay rate of the singular values of $\boldsymbol{R}$, improving the effectiveness of the RSVD algorithm. The QR factorizations improve numerical stability, as discussed by Halko *et al.* [53].

Algorithm 2 Power iteration

```
1: Inputs: $\boldsymbol{R}, \boldsymbol{Y}, q$
2: for $i = 1 : q$ do
3:     $\boldsymbol{Q} \leftarrow \mathrm{qr}(\boldsymbol{Y})$          ▷ For stabilization purposes
4:     $\boldsymbol{Y} \leftarrow \boldsymbol{R}^*\boldsymbol{Q}$          ▷ Sample the image of $\boldsymbol{R}$
5:     $\boldsymbol{Q} \leftarrow \mathrm{qr}(\boldsymbol{Y})$          ▷ For stabilization purposes
6:     $\boldsymbol{Y} \leftarrow \boldsymbol{R}\boldsymbol{Q}$          ▷ Sample the range of $\boldsymbol{R}$
7: Output: $\boldsymbol{Y}$
```

2.3.2 RSVD for resolvent analysis

The algorithm outlined in the previous section assumes direct access to the matrix $\boldsymbol{R}$. In the context of resolvent analysis, $\boldsymbol{R}$ is defined in terms of an inverse, which should be avoided. Ribeiro *et al.* [118] addressed this challenge by adopting the approach developed by Jeun *et al.* [66] for computing resolvent modes using an Arnoldi algorithm.

The idea is to replace multiplication of $\boldsymbol{R}$ or $\boldsymbol{R}^*$ by solving an equivalent linear system. For example, $\boldsymbol{Y} = \boldsymbol{R}\boldsymbol{\Theta}$ (line 3 of Algorithm 1) can be obtained by solving the linear system

$$(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})\boldsymbol{Y} = \boldsymbol{\Theta} \tag{2.12}$$

since $\boldsymbol{R}^{-1} = (\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})$. Similarly, $\boldsymbol{S} = \boldsymbol{Q}^*\boldsymbol{R}$ (line 7 of Algorithm 1) can be replaced with solving

$$(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^*\boldsymbol{S}^* = \boldsymbol{Q}. \tag{2.13}$$

The same concept can be used to replace multiplication by $\boldsymbol{R}$ and $\boldsymbol{R}^*$ in Algorithm 2.

Typically, the linear systems are solved by computing an LU decomposition

$$(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A}) = \boldsymbol{L}\boldsymbol{P}, \tag{2.14}$$

where $\boldsymbol{L}$ and $\boldsymbol{P}$ are the lower and upper triangular matrices (we use $\boldsymbol{P}$ to denote the upper triangular matrix instead of $\boldsymbol{U}$ to avoid confusion with the left singular vectors). The same LU decomposition can be used also for $(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^*$ since

$$(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^* = (\boldsymbol{L}\boldsymbol{P})^* = \boldsymbol{P}^*\boldsymbol{L}^*. \tag{2.15}$$

Solving these linear systems is indeed significantly less computationally demanding than computing the inverse of $(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})$ to form $\boldsymbol{R}$ and performing subsequent matrix-matrix multiplication in the RSVD algorithm. The remaining steps of the algorithm incur negligible computational costs and are not altered. In the remainder of our chapter, we will use the term "RSVD-LU" to refer to the modified version of RSVD that is compatible with resolvent analysis [118].

2.4 Computing resolvent modes using time stepping

An alternative class of methods for computing resolvent modes utilizes time stepping. This idea was first proposed by Monokrousos *et al.* [98] and recently was improved upon by Martini *et al.* [92], who introduced two methods: the transient response method and the steady-state response method. The latter was found to be better suited for complex algorithms, and we will employ and extend this method in the present chapter.

2.4.1 The action of the resolvent operator via time stepping

The central idea of the time-stepping approach is to obtain the action of the resolvent operator on a vector (or matrix) by solving the linear system that underlies the resolvent operator in the time domain. In this context, the action of a matrix $\boldsymbol{R}$ on a vector (or matrix) $\boldsymbol{b}$ is defined as follows; Given $\boldsymbol{b}$, our objective is to compute $\boldsymbol{x} = \boldsymbol{R}\boldsymbol{b}$, which is equivalent to solving the linear system $\boldsymbol{R}^{-1}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$.

Starting with a harmonically forced ordinary differential equation (ODE)

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{A}\boldsymbol{q} + \boldsymbol{f}, \tag{2.16}$$

where

$$\boldsymbol{f}(t) = \hat{\boldsymbol{f}} e^{\mathrm{i}\omega t} \tag{2.17}$$

is the harmonic forcing with frequency $\omega \in \mathbb{R}$ and $\hat{\boldsymbol{f}} \in \mathbb{C}^N$ is an arbitrary vector. The steady-state response of (2.16) is

$$\boldsymbol{q}(t) = \hat{\boldsymbol{q}}_s e^{\mathrm{i}\omega t}, \tag{2.18}$$

where

$$\hat{\boldsymbol{q}}_s = (\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^{-1} \hat{\boldsymbol{f}} = \boldsymbol{R} \hat{\boldsymbol{f}} \tag{2.19}$$

is the Fourier-domain solution. Therefore, the action of $\boldsymbol{R}$ can be obtained by computing the steady-state solution $\boldsymbol{q}(t)$ of (2.16) and subsequently taking a Fourier transform to obtain $\hat{\boldsymbol{q}}_s$. Similarly, the action of $\boldsymbol{R}^*$ can be obtained by computing the steady-state response $\boldsymbol{z}(t)$ of the adjoint equation

$$\begin{aligned} -\frac{d\boldsymbol{z}}{dt} &= \boldsymbol{A}^* \boldsymbol{z} + \boldsymbol{f}, \\ \boldsymbol{f} &= \hat{\boldsymbol{f}} e^{\mathrm{i}\omega t}, \end{aligned} \tag{2.20}$$

backward in time and taking a Fourier transform to obtain

$$\hat{\boldsymbol{z}}_s = (-\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A}^*)^{-1} \hat{\boldsymbol{f}} = \boldsymbol{R}^* \hat{\boldsymbol{f}}. \tag{2.21}$$

The arbitrary harmonic forcing term $\hat{\boldsymbol{f}}$ can be a matrix instead of a vector by defining $\hat{\boldsymbol{F}} \in \mathbb{C}^{N\times k}$. In that case, each column of the solutions $\hat{\boldsymbol{Q}}$ and $\hat{\boldsymbol{Z}}$ corresponds to one specific column of the forcing matrix.

2.4.2 Direct and adjoint actions for a range of frequencies

This section describes an important contribution from Martini *et al.* [92] that allows us to compute the action of the resolvent operator for a set of desired frequencies while time-stepping the equations only once. Integrating (2.16) typically generates a transient response T_t before obtaining the desired steady-state solution, as shown in figure 2.1. The length of T_t affects the length of time stepping and the accuracy of the output, as discussed in §2.7.2.2. The discrete nature of time stepping encourages the usage of discrete Fourier transform (DFT) where $\hat{\boldsymbol{q}}_s(\omega)$ can be obtained for a base frequency, ω_{min}, and its harmonics, $n\omega_{min}$, where $n \in \mathbb{Z}$. The DFT necessitates a specific time length of $T_s = 2\pi/\omega_{min}$ in order to accurately resolve the longest wavelength of interest. The number of snapshots within the steady-state period T_s determines the lowest frequency that can be resolved.

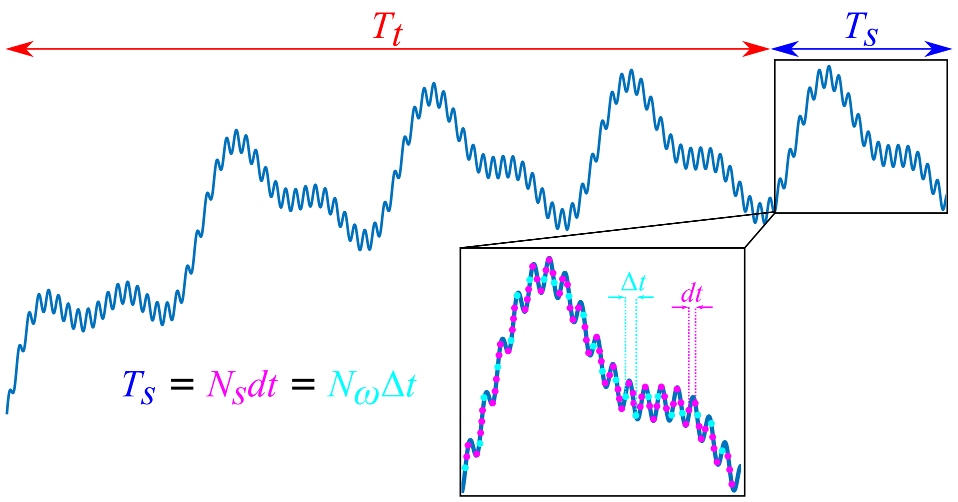


Figure 2.1: Schematic of the response waveform. The solution contains a transient portion of length T_t before the steady-state solution of period T_s is achieved. The numerical solution contains N_s time steps of size dt within one period of the steady-state solution, but only N_ω points with Δt spacing are required to decompose the N_ω frequencies of interest without aliasing.

In order to compute resolvent modes for all frequencies of interest

$$\mathit{\Omega} = \{0, \pm\omega_{min}, \pm 2\omega_{min}, \pm 3\omega_{min}, \ldots, \pm\omega_{max}\}, \tag{2.22}$$

where ω_{max} represents the highest frequency of interest, the forcing term

$$\boldsymbol{f} = \sum_{\omega_j \in \mathit{\Omega}} \hat{\boldsymbol{f}}_j e^{\mathrm{i}\omega_j t} \tag{2.23}$$

must include all frequencies in $\mathit{\Omega}$. The minimum number of snapshots within the T_s-period is $N_\omega = 2\lceil \frac{\omega_{max}}{\omega_{min}} \rceil$ according to Nyquist's theorem [104]. Performing time integration of (2.16) results in computing N_s steady-state snapshots within the T_s-period, where typically $N_s \geq N_\omega$, as the time step (dt) is chosen to ensure sufficient integration accuracy. Ultimately, by choosing N_ω steady-state snapshots, we can determine the Fourier coefficients by taking a DFT.

To elaborate on the previous point, assume a set of snapshots $\boldsymbol{Q}_{N_s} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \ldots, \boldsymbol{q}_{N_s}\}$ (analogous to the pink dots in figure 2.1), where $\boldsymbol{q}_j$ represents the j^{th} steady-state snapshot in the time domain. The fast Fourier transform (FFT) can efficiently compute $\hat{\boldsymbol{Q}}_{N_s} = \{\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, \ldots, \hat{\boldsymbol{q}}_{N_s}\}$. However, the maximum resolved frequency within $\hat{\boldsymbol{Q}}_{N_s}$ surpasses ω_{max} since typically $N_\omega \sim O(10^2)$, and $N_s \sim O(10^3 - 10^5)$. Therefore, an optimal size to resolve all $\omega \in \mathit{\Omega}$ without aliasing is to consider N_ω equally spaced snapshots in

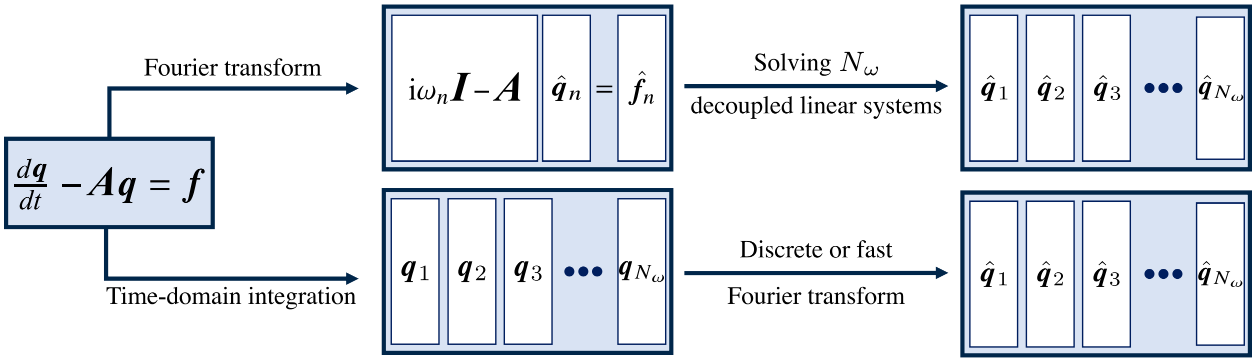

Fourier transform
$\frac{d\boldsymbol{q}}{dt} - \boldsymbol{A}\boldsymbol{q} = \boldsymbol{f}$
$\mathrm{i}\omega_n \boldsymbol{I} - \boldsymbol{A}$
$\hat{\boldsymbol{q}}_n$
$=$
$\hat{\boldsymbol{f}}_n$
Solving N_ω decoupled linear systems
$\hat{\boldsymbol{q}}_1$ $\hat{\boldsymbol{q}}_2$ $\hat{\boldsymbol{q}}_3$ ••• $\hat{\boldsymbol{q}}_{N_\omega}$
Time-domain integration
$\boldsymbol{q}_1$ $\boldsymbol{q}_2$ $\boldsymbol{q}_3$ ••• $\boldsymbol{q}_{N_\omega}$
Discrete or fast Fourier transform
$\hat{\boldsymbol{q}}_1$ $\hat{\boldsymbol{q}}_2$ $\hat{\boldsymbol{q}}_3$ ••• $\hat{\boldsymbol{q}}_{N_\omega}$

Figure 2.2: Flowchart depicting the action of $\boldsymbol{R}$ on N_ω inputs for the RSVD-LU (upper route) and the RSVD-Δt (bottom route) algorithms. Both routes produce the same result, but the bottom route is computationally advantageous for large systems.

$\boldsymbol{Q}_{N_\omega} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, ..., \boldsymbol{q}_{N_\omega}\}$ (analogous to the cyan dots in figure 2.1). Taking the FFT of $\boldsymbol{Q}_{N_\omega}$ yields $\hat{\boldsymbol{Q}}_{N_\omega} = \{\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, ..., \hat{\boldsymbol{q}}_{N_\omega}\}$, where each member $\hat{\boldsymbol{q}}_j$ represents the solution to $(\mathrm{i}\omega_j \boldsymbol{I} - \boldsymbol{A})\hat{\boldsymbol{q}}_j = \hat{\boldsymbol{f}}_j$, with $\omega_j \in \Omega$.

To avoid leakage, the equidistant snapshots within $\boldsymbol{Q}_{N_\omega}$ need to span the entire T_s period, *i.e.*,

$$T_s = dt \times N_s = \Delta t \times N_\omega. \tag{2.24}$$

For a given pair $(\omega_{min}, \omega_{max})$,

$$\Delta t = \frac{T_s}{N_\omega} = \frac{2\pi/\omega_{min}}{2\lceil \frac{\omega_{max}}{\omega_{min}} \rceil} \tag{2.25}$$

is predetermined, so dt must be selected such that $\frac{N_s}{N_\omega} \in \mathbb{N}$.

Figure 2.2 demonstrates the equivalence between computing the action of $\boldsymbol{R}$ for a range of frequencies in both the RSVD-LU and RSVD-Δt algorithms. Starting from the LNS equations, the upper route involves applying a Fourier transform before solving N_ω decoupled linear systems to compute the action of the resolvent operator on N_ω forcing inputs. The bottom route involves integrating the LNS equations in the time domain, followed by a Fourier transform to generate the same output as the upper route. All frequencies of interest, $\omega \in \Omega$, are included in the forcing so that the time stepping is performed only once, and the response at each frequency is obtained using a DFT or FFT.

2.5 RSVD-Δt: RSVD with time stepping

Our algorithm, which we refer to as RSVD-Δt, uses time stepping to eliminate the computational bottleneck within the RSVD algorithm for large systems. Specifically, solving the direct and adjoint LNS equations to apply the action of $\boldsymbol{R}$ and $\boldsymbol{R}^*$ circumvents the need for LU decomposition, improving the scaling of the algorithm (see §2.6), enabling resolvent analysis for the large systems typical of three-dimensional flows. RSVD-Δt is outlined in Algorithm 3 and described in what follows.

Algorithm 3 RSVD-Δt

```
 1: Inputs: $\boldsymbol{A}, k, q, \Omega$, TSS, $dt, T_t$
 2: $\hat{\boldsymbol{\Theta}} \leftarrow \mathrm{randn}(N, k, N_\omega)$                     ▷ Create random test matrices
 3: $\hat{\boldsymbol{Y}} \leftarrow$ DirectAction($\boldsymbol{A}, \hat{\boldsymbol{\Theta}}$,TSS, $dt, T_t$)          ▷ Sample the range of $\boldsymbol{R}$
 4: if $q > 0$ then                                                   ▷ Optional power iteration
 5:     $\hat{\boldsymbol{Y}} \leftarrow$ PI($\boldsymbol{A}, \hat{\boldsymbol{Y}}, q$,TSS, $dt, T_t$)          ▷ Algorithm 2 with time stepping
 6: $\hat{\boldsymbol{Q}} \leftarrow \mathrm{qr}_\Omega(\hat{\boldsymbol{Y}})$                      ▷ Build the orthonormal subspace $\hat{\boldsymbol{Q}}$
 7: $\hat{\boldsymbol{S}} \leftarrow$ AdjointAction($\boldsymbol{A}^*, \hat{\boldsymbol{Q}}$,TSS, $dt, T_t$)         ▷ Sample the image of $\boldsymbol{R}$
 8: $(\tilde{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \mathrm{svd}_\Omega(\hat{\boldsymbol{S}})$                              ▷ Obtain $\boldsymbol{\Sigma}, \boldsymbol{V}$
 9: $\boldsymbol{U} \leftarrow (\hat{\boldsymbol{Q}}\tilde{\boldsymbol{U}})_\Omega$                                            ▷ Recover $\boldsymbol{U}$
10: Outputs: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ for all $\omega \in \Omega$
```

Algorithm 3: Inputs include: linearized operator $\boldsymbol{A}$, number of modes k, number of power iterations q, frequency range Ω, TSS abbreviation for time-stepping scheme (*e.g.*, backward Euler), time step dt, and the transient length T_t. Outputs include: k response modes $\boldsymbol{U}$, k forcing modes $\boldsymbol{V}$ and the corresponding gains $\boldsymbol{\Sigma}$. Here, k, q, Ω are common parameters with RSVD-LU. $(\cdot)_\Omega$ means the function is separately applied to each $\omega \in \Omega$. `DirectAction` and `AdjointAction` are functions that solve the direct and adjoint LNS equations, respectively, with a predefined forcing. `PI` is a function that performs the power iteration.

As in the standard RSVD algorithm, the first step is to create random forcing matrices to sketch $\boldsymbol{R}$. Since our time-stepping approach computes all frequencies of interest at once, a separate test matrix $\hat{\Theta} \in \mathbb{C}^{N \times k}$ is generated for each frequency $\omega \in \Omega$ (line 2). Next (line 3), the `DirectAction` function solves the LNS equations forced by the set of test matrices in the time domain to obtain the sketch $\hat{\boldsymbol{Y}}$ of the resolvent operator $\boldsymbol{R}$ for all $\omega \in \Omega$. Line 4 checks whether or not power iteration is desired, and if so (*i.e.*, $q > 0$), line 5 jumps to algorithm 2 to increase the accuracy of resolvent modes. All instances of applying the action of the resolvent operator or its adjoint in Algorithm 2 are performed via time stepping. In line 6, an orthonormal subspace $\hat{\boldsymbol{Q}}$ is constructed for the sketch at each frequency via QR decomposition. Note that the Ω subscript indicates that the operation is performed separately for each frequency $\omega \in \Omega$. Next, in line 7, the `AdjointAction` function solves the adjoint LNS equations forced by the set of $\hat{\boldsymbol{Q}}$ matrices in the time domain to sample the image of the resolvent operator $\boldsymbol{R}$ for all $\omega \in \Omega$. Finally, the estimates of the k leading right singular vector $\boldsymbol{V}$ and gains $\boldsymbol{\Sigma}$ are obtained via an economy SVD of the $N \times k$ matrix $\hat{\boldsymbol{S}}$ (line 8), and left singular vectors $\boldsymbol{U}$ are recovered in line 9.

For the sake of notational brevity, we have described resolvent analysis and the RSVD-Δt algorithm in the absence of non-identity input, output, and weight matrices in the main text

$$\begin{tikzcd}
\hat{\boldsymbol{F}} \arrow[r, "\text{Input}"] & \boxed{\text{Time stepping}} \arrow[r, "\text{Output}"] & \hat{\boldsymbol{Y}}
\end{tikzcd}$$

$$\begin{tikzcd}
\hat{\boldsymbol{F}} \arrow[r, "\text{Weighted}"] & \hat{\boldsymbol{F}}_W \arrow[r, "\text{Filtered}"] & \hat{\boldsymbol{F}}_{W,B} \arrow[r, "\text{Input}"] & \boxed{\text{Time stepping}} \arrow[r, "\text{Output}"] & \hat{\boldsymbol{Y}} \arrow[r, "\text{Filtered}"] & \hat{\boldsymbol{Y}}_C \arrow[r, "\text{Weighted}"] & \hat{\boldsymbol{Y}}_{C,W}
\end{tikzcd}$$

Figure 2.3: The schematic of computing the action of $\boldsymbol{R}$ on top and the action of $\tilde{\boldsymbol{R}}$ on the bottom row.

(see §2.1). We briefly explain the modifications required to include these additional matrices. In this case, solving the generalized Rayleigh quotient (2.7) is equivalent to computing the SVD of the weighted resolvent operator [160]

$$\tilde{\boldsymbol{R}} = \boldsymbol{W}_q^{1/2}\boldsymbol{C}(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{W}_f^{-1/2}, \tag{2.26a}$$

$$\tilde{\boldsymbol{R}} = \tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}^*, \tag{2.26b}$$

and further

$$\begin{aligned} \boldsymbol{U} &= \boldsymbol{W}_q^{-1/2}\tilde{\boldsymbol{U}}, \\ \boldsymbol{V} &= \boldsymbol{W}_f^{-1/2}\tilde{\boldsymbol{V}}, \end{aligned} \tag{2.27}$$

where $\boldsymbol{\Sigma}$ contains the gains, and $\boldsymbol{V}$ and $\boldsymbol{U}$ are forcing and response modes, respectively. The resolvent operator is recovered as

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\boldsymbol{W}_f. \tag{2.28}$$

Time-stepping can effectively act as a surrogate for the action of the weighted resolvent operator $\tilde{\boldsymbol{R}}$ (or equivalently $\tilde{\boldsymbol{R}}^*$). In other words, our objective is to compute

$$\hat{\boldsymbol{y}} = \tilde{\boldsymbol{R}}\hat{\boldsymbol{f}} = \boldsymbol{W}_q^{1/2}\boldsymbol{C}(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{W}_f^{-1/2}\hat{\boldsymbol{f}} \tag{2.29}$$

for all $\omega \in \Omega$ using time stepping. The process begins by computing the product between $\hat{\boldsymbol{f}}_W = \boldsymbol{W}_f^{-1/2}\hat{\boldsymbol{f}}$ in Fourier space, followed by $\hat{\boldsymbol{f}}_{W,B} = \boldsymbol{B}\hat{\boldsymbol{f}}_W$. The products involving weight and input/output matrices are efficiently executed due to their sparsity. These operations are conducted for all $\omega \in \Omega$ to obtain $\hat{\boldsymbol{F}}_{W,B}$. Subsequently, the action of $(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}$ is computed on $\hat{\boldsymbol{F}}_{W,B}$ using time stepping to yield $\hat{\boldsymbol{Y}}$. The resulting output undergoes $\hat{\boldsymbol{y}}_C = \boldsymbol{C}\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{y}}_{C,W} = \boldsymbol{W}_q^{1/2}\hat{\boldsymbol{y}}_C$, which are repeated for all frequencies to obtain $\hat{\boldsymbol{Y}}_{C,W}$. Figure 2.3 visually illustrates the order of calculations for $\boldsymbol{R}$ in the top row and $\tilde{\boldsymbol{R}}$ in the bottom row. An analogous process is utilized to compute the action of $\tilde{\boldsymbol{R}}^*$.

2.6 Computational complexity

The primary advantage of the RSVD-Δt algorithm is its reduced computational cost. In this section, we discuss the CPU and memory cost scaling of applying the action of the resolvent operator via time stepping and compare it to LU-based approaches, as summarized in table 2.1. We assume that the LNS equations are discretized using a sparse scheme such as finite differences, finite volume, or finite elements. Once the linearized operator $\boldsymbol{A}$ is constructed, the goal is to solve the linear system given by

$$(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b} \tag{2.30}$$

to compute the action of $\boldsymbol{R}$ on $\boldsymbol{b}$.

2.6.1 CPU cost

Direct solvers find the solution of (2.30) to machine precision. A common approach is to find the LU decomposition of $(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})$ and solve the decomposed system via back substitution. The process of computing lower and upper triangular matrices with full or partial pivoting can be extremely expensive for large systems [32] and is often the dominant cost of solving a linear system [89]. Once the LU decomposition is obtained, solving the LU-decomposed system is typically comparatively inexpensive. The theoretical cost scaling of LU decomposition of the sparse matrices that arise from collocation-based discretization methods (like finite differences) is $O(N^{1.5})$ and $O(N^2)$ for two-dimensional and three-dimensional systems, respectively [2]. The larger scaling exponent and number of grid points present in a three-dimensional problem make the LU decomposition of the corresponding linear operator costly. Optimized algorithms for computing LU decomposition are available in open-source software packages such as LAPACK [4], MUMPS [3], PARDISO [129], and Hypre [41], which are designed to leverage massive parallelization. The LU decomposition becomes increasingly dominant (compared to solving the LU-decomposed system or other algorithmic steps) as the size of the system increases for both the standard Arnoldi-based method and the RSVD-LU algorithm, reducing the computational advantage of the latter.

Iterative solvers contain convergence criteria that can be adjusted to reduce computational cost at the expense of a less accurate solution. The performance of iterative solvers strongly depends on the condition number κ, the ratio between the largest and smallest eigenvalues of a matrix. Matrices with condition numbers of great than $\sim 10^4$ are considered to be ill-conditioned [126], which can cause slow convergence and numerical stability issues for iterative solvers [142]. The LNS operator $\boldsymbol{A}$ is typically a sparse but ill-conditioned matrix.

Problem size	Action of $\boldsymbol{R}$	CPU time	Memory
Two-dimensional	LU decomposition	$O(N^{1.5})$	$O(N^{1.2})$
	time stepping	$O(N)$	$O(N)$
Three-dimensional	LU decomposition	$O(N^{2})$	$O(N^{1.6})$
	time stepping	$O(N)$	$O(N)$

Table 2.1: The scaling of CPU time and memory requirements with respect to N for computing the action of $\boldsymbol{R}$ (or $\boldsymbol{R}^*$) using time stepping and LU decomposition.

When ω is small, $(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})$ inherits the ill-conditioning of $\boldsymbol{A}$, making the use of an iterative solver challenging. The conditioning improves as ω increases, so the lowest frequencies control the overall cost of using an iterative method to compute resolvent modes. In addition to the condition number, other properties such as the size, sparsity pattern, and density (or sparsity ratio) of a matrix can also ease or aggravate the situation [161].

In principle, iterative solvers are attractive when solving (2.30) up to machine precision is unnecessary, as is the case when using the RSVD algorithm, which is already an approximation. The main challenge remains the typically high condition number of $(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})$, as explained above. One potential solution is the common practice of using a preconditioner [125]. Preconditioners are matrices that are multiplied on the left, right, or both sides of the target matrix to decrease its condition number and thus increase the convergence of iterative solvers. The methods of computing preconditioners and numerous related theories and practices are neatly summarized in a few surveys [6, 12, 109]. Despite numerous advancements in this field, not all matrices have effective preconditioners. Some recent studies [63] are exploring the use of iterative methods to solve (2.30) within the context of resolvent analysis, but direct methods, especially LU decomposition, have long been the dominant choice [96, 66, 137, 118].

The cost of time-stepping methods rely on integrating the LNS equations in the time domain. Time-stepping of ODEs (such as the one in (2.16)) has a long history and is a mature field [52, 166, 161]. Herein, two classes – implicit and explicit integration schemes – are available and widely used in the scientific computing community.

Implicit integrators possess better stability properties but require a system of the form

$$\boldsymbol{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b} \tag{2.31}$$

to be solved at every iteration. Here, $\boldsymbol{b} \in \mathbb{C}^{N\times k}$ is a function of the solution at previous time and the exogenous forcing (if present), and $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{N\times N}$ is the temporal discretized operator, which is a function of the linear operator $\boldsymbol{A}$. For example, $\boldsymbol{\mathcal{A}}$ can be written as a first-order polynomial of the form $\boldsymbol{\mathcal{A}} = c_1\boldsymbol{I} + c_2\boldsymbol{A}$ for multi-step methods, where constants are determined based on integration scheme and time step, *e.g.*, $\boldsymbol{\mathcal{A}} = \boldsymbol{I} - dt\boldsymbol{A}$ for backward Euler. A superficial comparison between (2.31) and (2.30) indicates that implicit time steppers suffer from the same issues elaborated above. However, the key difference is that $\boldsymbol{A}$ is multiplied by the (small) time step dt, so the ill-conditioning of $\boldsymbol{A}$ is largely overwhelmed by the ideal conditioning of the identity matrix $\boldsymbol{I}$. This improved conditioning makes possible the application of iterative solvers. Implicit integrators require at least one LU decomposition of $\boldsymbol{\mathcal{A}}$ for direct solvers or a preconditioner for indirect solvers, which are not $O(N)$ operations. However, this one-time cost is often small enough that it is overwhelmed by other operations such that the observed computational complexity remains $O(N)$.

For explicit integrators, the solution at each time step is an explicit function of the solution (and exogenous forcing) at previous time steps. Accordingly, a solution of a linear system is not required, and each step contains only inexpensive sparse matrix-vector products for a linear ODE such as (2.30), making each step rapidly computable. The downside of explicit methods is that they are less numerically stable and often require many small steps to ensure stability for stiff systems [147]. Nevertheless, the drastically smaller cost of each step for explicit integrators often outweighs the disadvantage of requiring many small steps, and many computational fluid dynamics codes are equipped with explicit integrators such as Runge–Kutta schemes.

Explicit integrators involve repeatedly multiplying the sparse matrix $\boldsymbol{A}$ with vectors during the time-stepping process, which scales like $O(N)$. The number of times these operations must be performed over a fixed time interval is determined by the time step dt. If the time step is set by a CFL-like stability condition, then it scales like $O(N^{-1/2})$ and $O(N^{-1/3})$ for two- and three-dimensional problems, respectively, and the total number of time steps N_t in a fixed time interval scales as the inverse of these values. The overall CPU cost scales like $O(NN_t)$, leading to $O(N^{3/2})$ and $O(N^{4/3})$ scaling for two- and three-dimensional problems, respectively. In practice, however, the time step is chosen to control the error associated with the highest frequency of interest rather than being determined by a CFL condition, as discussed in §2.7.2.1, and is thus independent of N. Accordingly, we observe linear scaling when using explicit integrators in practice.

2.6.2 Memory requirements

Supercomputers and parallel solvers can keep the hope of computing the LU decomposition of massive and poorly conditioned systems alive; however, massive calculations require massive storage, and memory becomes the top issue [28]. Generally, direct solvers are more robust than iterative solvers but can consume significant memory due to the fill-in process of factorization [89]. The memory requirement associated with LU decomposition for resolvent analysis has been empirically observed to scale like $O(N^{1.2})$ and $O(N^{1.6})$ for two-dimensional and three-dimensional systems, respectively [159]. The exponents are not guaranteed and can become better or worse depending on the system of interest.

Explicit integration schemes have certain advantages over implicit integration schemes. Explicit schemes typically do not require much space for sparse matrix-vector products. The required memory is mainly used to store the forcing and response modes in Fourier space which scales like $O(N)$, as will be discussed in §2.8.1.1. On the other hand, implicit integration schemes, in addition to the Fourier space matrices, require memory for solving (2.31), which depends heavily on the sparsity of the LU-decomposed matrices or the iterative methods employed. For some systems, these methods may scale worse than $O(N)$, resulting in increased memory requirements.

2.6.3 Matrix-free implementation

So far, we have assumed that the LNS matrix $\boldsymbol{A}$ is explicitly formed. In contrast to the standard frequency-domain approaches including the RSVD-LU algorithm, our time-stepping approach can be applied in a matrix-free manner using any code with linear direct and adjoint capabilities without explicitly forming $\boldsymbol{A}$ [108, 92]. In this case, the cost scaling of our algorithm will follow that of the underlying Navier-Stokes code, which is again typically linear with the problem dimension.

2.7 Sources of error in the RSVD-Δt algorithm

Next, we identify sources of error within the RSVD-Δt algorithm, which stem from the RSVD approximation and the time-stepping approach used to compute the action of $\boldsymbol{R}$. By effectively addressing these sources of error, the RSVD-Δt method can be optimized for improved efficiency.

2.7.1 RSVD approximation

RSVD offers estimates of the resolvent modes rather than exact ground truth. The accuracy of these estimates is extensively discussed in Halko *et al.* [53], and it naturally depends on the gain separation. As mentioned earlier, incorporating power iteration and employing a few extra test vectors beyond the desired number of modes can improve the accuracy of the resolvent modes. In many cases, the approximation error of RSVD is the primary source of error in RSVD-Δt, such that it accurately reproduces the results of the RSVD-LU algorithm.

2.7.2 Time stepping sources of error

When computing the action of $\boldsymbol{R}$ and $\boldsymbol{R}^*$ using time stepping, two types of errors are introduced in addition to the RSVD approximation.

2.7.2.1 Truncation error

The first source of time-stepping error is the truncation error of the numerical integration schemes used to solve the time-domain equations. Common approaches include classical numerical integration schemes such as Runge–Kutta, implicit/explicit Euler, Adams-Moulton family, and others [52, 166]. These methods introduce truncation errors resulting from the approximation of Taylor series expansions. Hence, a chosen time step introduces an expected truncation error, with higher-order schemes providing greater precision.

Local truncation error (LTE) is derived for ODEs as

$$LTE = C\frac{d^p f(t)}{dt^p}O(dt^p), \tag{2.32}$$

where C is a constant, and p is the order of the time-stepping scheme. In this study, our focus is on ODEs with harmonic forcing $f(t) = \hat{f}e^{\mathrm{i}\omega t}$. Substituting the forcing term into (2.32), we observe that

$$LTE \propto O((\omega dt)^p). \tag{2.33}$$

This equation indicates that for a fixed time step dt, the error in the computed resolvent modes will be frequency dependent and vary as ω^p. Therefore, in addition to satisfying any stability constraints, the time step dt must be selected such that $\omega_{max}dt$ is sufficiently small to obtain accurate resolvent modes up to the maximum desired frequency ω_{max}.

2.7.2.2 Transient error

The second source of time-stepping error arises from the unwanted transient response. The solution of (2.16) can be written as a sum of its transient and steady-state components,

$$\boldsymbol{q}(t) = \boldsymbol{q}_t(t) + \boldsymbol{q}_s(t), \tag{2.34}$$

where the transient part $\boldsymbol{q}_t$ decays to zero as $t \to \infty$ and the steady-state part $\boldsymbol{q}_s$ is T-periodic, *i.e.*, $\boldsymbol{q}_s(t+T) = \boldsymbol{q}_s(t)$. Taking the Fourier transform of each part leads to

$$\hat{\boldsymbol{q}}(\omega) = \hat{\boldsymbol{q}}_t(\omega) + \hat{\boldsymbol{q}}_s(\omega). \tag{2.35}$$

Only the steady-state solution is desired, so any non-zero transient part constitutes an error in our representation of the action of the resolvent operator (or its adjoint) on the prescribed forcing. The transient response can be understood as the response of the system to an initial condition that is not synced with the forcing applied to the system. It may initially grow for non-normal systems like the LNS equations [131] but eventually decays at the rate of the least-damped eigenvalue of $\boldsymbol{A}$.

We define the transient error as the ratio between the norms of the transient and steady-state responses,

$$\epsilon = \frac{\|\boldsymbol{q}_t\|}{\|\boldsymbol{q}_s\|}, \tag{2.36}$$

where the l^2-norms can be replaced with $\|\cdot\|_q$ for non-identity weight matrices. In cases where we solve (2.16) with a zero initial condition (which is often the case), *i.e.*, $\boldsymbol{q}(0) = \boldsymbol{q}_t(0) + \boldsymbol{q}_s(0) = 0$, the transient error is initially one,

$$\epsilon(0) = \frac{\|\boldsymbol{q}_t(0)\|}{\|\boldsymbol{q}_s(0)\|} = 1. \tag{2.37}$$

In the long term, the transient error approaches zero,

$$\lim_{t\to\infty} \epsilon(t) = \lim_{t\to\infty} \frac{\|\boldsymbol{q}_t(t)\|}{\|\boldsymbol{q}_s(t)\|} = 0, \tag{2.38}$$

since $\|\boldsymbol{q}_s\|$ remains bounded.

The eigenspectrum of the linearized system $\boldsymbol{A}$ provides insights into the long-term response of the homogeneous system. Any initial perturbation will eventually follow the least-damped mode. However, in practice, computing the eigenspectrum of $\boldsymbol{A}$ is challenging, especially for large systems. Even obtaining a small number of eigenvalues using the Krylov-Schur method can be cumbersome. Therefore, a practical approach to understanding the long-term behavior of a system is to simulate the homogeneous ODE

$$\frac{d\boldsymbol{q}_h}{dt} - \boldsymbol{A}\boldsymbol{q}_h = 0, \tag{2.39}$$

initialized with a random state [39, 37]. A random perturbation represents a worst-case scenario, as it excites all the slow modes of $\boldsymbol{A}$. By monitoring the norm of $\boldsymbol{q}_h$ over time, we can estimate the slowest decay rate, which corresponds to the real part of the least-damped eigenvalue of $\boldsymbol{A}$. This also gives us an indication of the expected magnitude of the transient error. Performing a DFT on one cycle of the transient response allows us to determine the anticipated level of transient error within the desired frequency range.

While it is possible to simply wait for the transient error to naturally decay over time, this approach comes with increased CPU cost, as it requires longer simulation durations. In §2.8.2, we will present an efficient method to achieve a smaller transient error within a shorter time frame.

2.8 Optimizing the RSVD-Δt algorithm

In this section, we present several approaches aimed at reducing the CPU cost and memory requirements of the RSVD-Δt algorithm. These approaches, combined with the improved cost scaling of RSVD-Δt compared to the RSVD-LU algorithm as discussed in §2.6, are crucial in facilitating affordable resolvent analysis of complex three-dimensional flows.

2.8.1 Minimizing memory requirements

First, we describe several strategies to minimize the memory required to compute resolvent modes for a given problem.

2.8.1.1 Streaming Fourier sums

A straightforward implementation of computing the action of $\boldsymbol{R}$ (or $\boldsymbol{R}^*$) via time stepping entails (i) transferring the forcing from Fourier space to the time domain, $\hat{\boldsymbol{F}} \xrightarrow{\text{iFFT}} \boldsymbol{F}$, ($ii$) performing integration to obtain the steady-state solutions saved with a specific time interval, as explained in §2.4.2, and (iii) transferring the response back to frequency space, $\boldsymbol{Q} \xrightarrow{\text{FFT}} \hat{\boldsymbol{Q}}$. A schematic of these steps is displayed in figure 2.4(a).

The first step requires zero-padding $\hat{\boldsymbol{F}} \in \mathbb{C}^{N\times k\times N_\omega}$ since $\boldsymbol{F} \in \mathbb{C}^{N\times k\times N_s}$ is required at all $N_s \gg N_\omega$ points in the period associated with the time step $dt \ll \Delta t$ required for accurate time stepping. The iFFT is computationally efficient but storing its output requires a minimum memory allocation of $O(NkN_s)$, excluding space for the iFFT calculations themselves. $\hat{\boldsymbol{F}}$ is automatically discarded before proceeding to the second step. In step (ii), $\boldsymbol{f}_j \in \boldsymbol{F}$ is used to force the linear system at each time step until the transient ends, and the steady-state responses are stored in $\boldsymbol{Q}$. After integration, $\boldsymbol{F}$ is no longer needed and is removed. Lastly, obtaining $\hat{\boldsymbol{Q}}$ from $\boldsymbol{Q}$ using an FFT requires an $O(NkN_\omega)$ space to store the output. Overall,

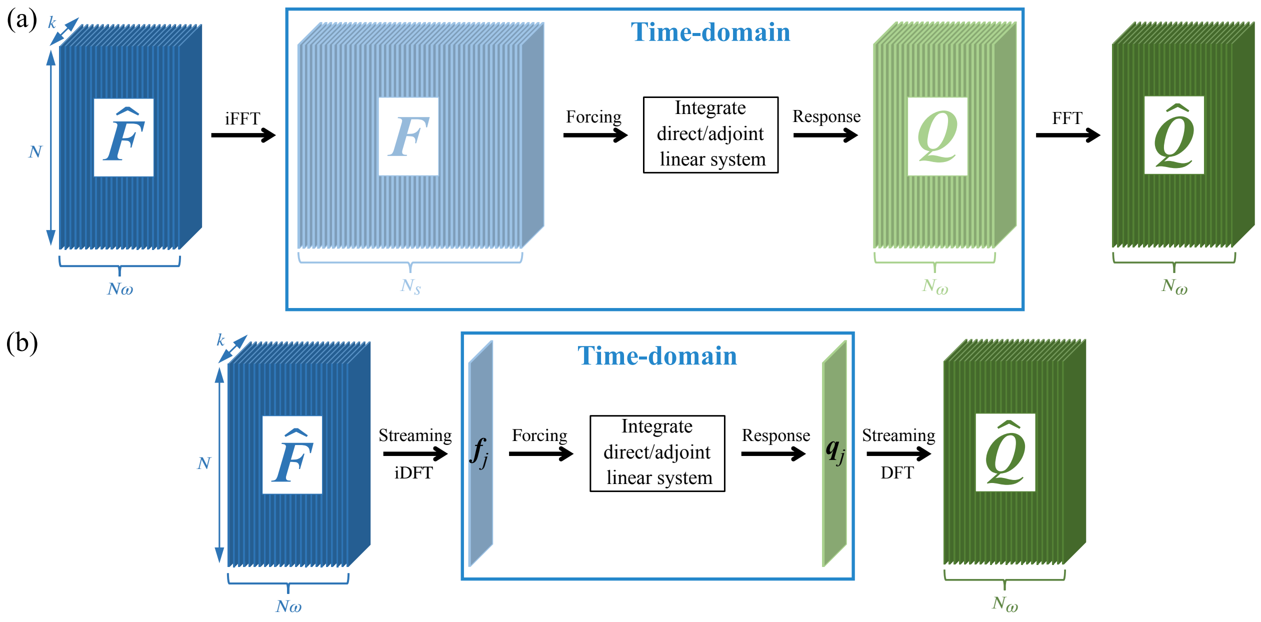


Figure 2.4: Schematic of the action of $\boldsymbol{R}$ with (a) FFT/iFFT and (b) streaming DFT/iDFT methods to transform between the Fourier and time domains.

a minimum memory allocation of $O(NkN_s) + O(NkN_\omega)$ is necessary to store both $\boldsymbol{F}$ and $\boldsymbol{Q}$ simultaneously.

The memory requirements of this process can be significantly reduced by leveraging streaming Fourier sums, as in the streaming SPOD algorithm proposed by Schmidt & Towne [135]. This procedure is shown schematically in figure 2.4(b). In the streaming approach, a new forcing snapshot is created before each time step and promptly removed afterward. Also, the contribution to the Fourier modes of the response is computed only at specific time steps, after which the snapshot of the solution can be discarded. This eliminates the need to permanently store any data in the time domain, reducing the memory requirement to $2 \times O(NkN_\omega)$ for storing $\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{Q}}$. The streaming implementation utilizes the DFT formulation to create forcing inputs and compute the effect of steady-state response data on the ensemble of Fourier coefficients, as demonstrated in the following.

At each time step, the instantaneous forcing is created from its Fourier mode using the definition of the inverse Fourier transform,

$$\boldsymbol{f}_p = \sum_{s=1}^{N_\omega} \boldsymbol{Z}'_{ps} \hat{\boldsymbol{f}}_s, \tag{2.40}$$

where $\boldsymbol{Z}'_{ps} = exp(-2\pi \mathrm{i}/N_s)^{(p-1)(s-1)}$. The integer p $(1 \leq p \leq N_s)$ specifies the phase of the periodic forcing at the current time step. Here, $\hat{\boldsymbol{f}}_s \in \mathbb{C}^{N \times k \times N_\omega}$ denotes Fourier modes that are accessible from memory. The sum is taken over every $\omega \in \Omega$, and it outputs the p^{th} time

$\mathcal{F}/\mathcal{F}^{-1}$	CPU time	Memory
iFFT	$Nk \times O(N_s log(N_s))$	$O(NkN_s)$
FFT	$Nk \times O(N_\omega log(N_\omega))$	$O(NkN_\omega)$
Streaming iDFT	$Nk \times O(N_{\text{total}} N_\omega)$	$O(NkN_\omega)$
Streaming DFT	$Nk \times O(N_\omega^2)$	$O(NkN_\omega)$

Table 2.2: Comparison of CPU time and memory requirements using FFT/iFFT and streaming DFT/iDFT methods transfer back and forth between Fourier space and time domain. $N_{\text{total}} = N_t + N_s$ is the total number of time steps including transient and steady-state parts.

domain snapshot $\boldsymbol{f}_p \in \mathbb{C}^{N\times k}$. This process continues in a loop of size N_s until the transient is passed and steady-state data is computed.

The response Fourier modes can be computed from the time-domain steady-state solutions in a similar streaming fashion. Following the definition of the DFT, each temporal snapshot $\boldsymbol{q}_l$ within the steady-state response contributes to each each Fourier mode according to the partial sum

$$[\hat{\boldsymbol{q}}_s]_r = [\hat{\boldsymbol{q}}_s]_{r-1} + \boldsymbol{Z}_{ls}\boldsymbol{q}_r = \sum_{l=1}^{r} \boldsymbol{Z}_{ls}\boldsymbol{q}_l, \tag{2.41}$$

where $\boldsymbol{Z}_{ls} = exp(-2\pi \mathrm{i}/N_\omega)^{(l-1)(s-1)}, 1 \le (l,s) \le N_\omega$. Here, $[\hat{\boldsymbol{q}}_s]_r$ represents the sum of contributions up to $\boldsymbol{q}_r$, which is the r^{th} steady-state response and should be removed after adding its contribution to the sum. The partial sum is complete once $r = N_\omega$, *i.e.*, the effect of all N_ω steady-state data is included.

A subtle but important difference between the iDFT matrix $\boldsymbol{Z'} \in \mathbb{C}^{N_\omega \times N_s}$ and the DFT matrix $\boldsymbol{Z} \in \mathbb{C}^{N_\omega \times N_\omega}$ is their sizes: $\boldsymbol{Z}$ is used to generate N_s temporal snapshots of the forcing from N_ω Fourier modes, while $\boldsymbol{Z'}$ is used to convert N_ω temporal snapshots of the steady-state solution into N_ω Fourier modes. The steaming process of the adjoint equations is identical, except the equations are integrated backward in time and indices within the Fourier sums are adjusted accordingly.

The CPU time and memory requirement of the FFT/iFFT and streaming DFT/iDFT approaches are summarized in table 2.2. Although the streaming method incurs slightly higher CPU cost due to the efficiency of the FFT algorithm, this CPU overhead is negligible compared to the cost of taking a time step. Moreover, the memory savings of the streaming method can be substantial; the ratio of the memory required by the iFFT and streaming iDFT methods used to create the forcing snapshots scales like $O(N_s/N_\omega)$, where $N_\omega \sim O(10^2)$, and $N_s \sim O(10^3 - 10^5)$ are typical values. Overall, the substantial memory benefit of the streaming method outweighs the small CPU penalty, especially for large systems.

2.8.1.2 Optimal cost for real-valued matrices

The linear operator $\boldsymbol{A}$ is often real-valued, in which case the memory requirements can be further reduced. Assuming $\boldsymbol{R} = (\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^{-1} = \boldsymbol{U\Sigma V}^*$, the resolvent operator corresponding to $-\omega$ can be written as

$$\boldsymbol{R}_{-\omega} = (-\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^{-1} = (\overline{\mathrm{i}\omega \boldsymbol{I}} - \overline{\boldsymbol{A}})^{-1} = \overline{(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^{-1}} = \overline{\boldsymbol{R}}_{\omega} = \overline{\boldsymbol{U}}\boldsymbol{\Sigma}\overline{\boldsymbol{V}}^{*}, \tag{2.42}$$

where $\bar{(\cdot)}$ denotes the complex conjugate and $\boldsymbol{A} = \overline{\boldsymbol{A}}$ when $\boldsymbol{A}$ is real-valued. Equation (2.42) proves that the gains of positive and negative frequencies are symmetric and the resolvent modes are complex conjugates of one another. Therefore, computing the resolvent modes for positive $\omega \in \Omega$ naturally provides results for negative frequencies. This symmetry halves the CPU cost for the RSVD-LU algorithm but does not reduce the memory requirement. On the other hand, in the case of RSVD-Δt, the memory requirements are halved, but there is no significant reduction in the CPU, as further elaborated.

Since the frequencies of interest become $\Omega_+ = \{0, +\omega_{min}, +2\omega_{min}, ..., +\omega_{max}\}$, the total number of frequencies becomes $\lfloor \frac{N_\omega}{2} \rfloor + 1$. In this scenario, only Fourier coefficients corresponding to $\omega \in \Omega_+$ are saved and the memory storage required for both input and output matrices ($\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{Q}}$ discussed in §2.8.1.1) is halved. In terms of CPU, generating the forcing and computing the response is twice as fast but the speed-up is not significant as the time stepping remains identical to the complex-valued case.

2.8.1.3 An additional option for reducing memory

If additional memory savings are required, the memory requirements of RSVD-Δt can be sharply reduced by dividing the frequencies of interest into multiple sets at the expense of additional CPU cost. For instance, when the frequencies are divided into d equal groups, the memory requirement is reduced by a factor of d. The penalty of doing so is that the CPU time scales proportionally with d, since the entire algorithm needs to be repeated for each group of frequencies. The RSVD-LU algorithm offers no such opportunity to reduce memory requirements, *e.g.*, to make a particular calculation possible on a given computer, at the expense of higher CPU cost.

2.8.2 Minimizing the CPU cost: efficient transient removal

Within the time-stepping process, the removal of the transient responses is crucial and is naturally accomplished through the long-time integration of (2.16), as discussed in §2.7.2.2. Nonetheless, certain LNS operators exhibit a painfully slow decay rate, resulting in lengthy transient durations and costly time stepping. Therefore, we present an efficient transient removal strategy to minimize the CPU cost.

Our strategy uses the differing evolution of the steady state and transient parts of the solution to directly compute and remove the transient from the solution. Considering two solutions of (2.16), $\boldsymbol{q}_1 = \boldsymbol{q}(t_1)$ and $\boldsymbol{q}_2 = \boldsymbol{q}(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts, as in (2.34), as

$$\begin{aligned} \boldsymbol{q}_1 &= \boldsymbol{q}_{s,1} + \boldsymbol{q}_{t,1}, \\ \boldsymbol{q}_2 &= \boldsymbol{q}_{s,2} + \boldsymbol{q}_{t,2}, \end{aligned} \tag{2.43}$$

where $\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}, \boldsymbol{q}_{t,1}$, and $\boldsymbol{q}_{t,2}$ are four unknowns. Applying a prescribed forcing in (2.16) at a single frequency ω yields

$$\boldsymbol{q}_{s,2} = \boldsymbol{q}_{s,1} e^{\mathrm{i}\omega\Delta t}. \tag{2.44}$$

Also, the transient response follows the form of a homogenous response, resulting in

$$\boldsymbol{q}_{t,2} = e^{\boldsymbol{A}\Delta t} \boldsymbol{q}_{t,1}. \tag{2.45}$$

Simplifying (2.43), (2.44), and (2.45) for $\boldsymbol{q}_{t,1}$, we obtain

$$(\boldsymbol{I} - e^{-\mathrm{i}\omega\Delta t} e^{\boldsymbol{A}\Delta t}) \boldsymbol{q}_{t,1} = \boldsymbol{b}, \tag{2.46}$$

where $\boldsymbol{b} = \boldsymbol{q}_1 - \boldsymbol{q}_2 e^{-\mathrm{i}\omega\Delta t}$ is known from the time-stepping solution. Equation (2.46) holds for any two points in time with arbitrary separation Δt. The exact steady-state solution with no transient error is obtained by solving (2.46) for $\boldsymbol{q}_{t,1}$ and using (2.43) to obtain $\boldsymbol{q}_{s,1} = \boldsymbol{q}_1 - \boldsymbol{q}_{t,1}$.

The prescribed forcing in RSVD-Δt consists of a range of frequencies, hence, it requires a pre-processing step to enable the transient removal strategy. We utilize $\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, ..., \boldsymbol{q}_{N_\omega}\}$ to construct $\hat{\boldsymbol{Q}} \in \mathbb{C}^{N \times N_\omega}$, where the snapshots are equidistant with a time interval of Δt. Additionally, we define $\boldsymbol{Q}^{\Delta t} = \{\boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{q}_4, ..., \boldsymbol{q}_{N_\omega+1}\}$ as a shifted matrix, resulting in $\hat{\boldsymbol{Q}}^{\Delta t} \in \mathbb{C}^{N \times N_\omega}$. Here, $\hat{\boldsymbol{q}}_j \in \hat{\boldsymbol{Q}}$ represents $\boldsymbol{q}_1$ in the above equations, while $\hat{\boldsymbol{q}}_j^{\Delta t} \in \hat{\boldsymbol{Q}}^{\Delta t}$ represents $\boldsymbol{q}_2$, both oscillating at the same frequency. Therefore, a single time stepping is sufficient to obtain (2.46) for all $\omega \in \Omega$.

We emphasize two crucial aspects of our strategy. Firstly, it functions as a post-processing step that comes into play after acquiring simulation snapshots, which encompass both transient and steady-state components. Its primary aim is to selectively remove the transient portion. Secondly, our strategy does not introduce any modifications to the LNS operator. Instead, it is tailored to solving equations that leave the steady-state response unaffected.

Solving (2.46) can be computationally expensive, particularly for large systems, even if we assume that computing $e^{\boldsymbol{A}\Delta t}$ is feasible. To address this issue, one possible approach is to choose a small Δt and expand the exponential term as $e^{\boldsymbol{A}\Delta t} = \sum_j \frac{(\boldsymbol{A}\Delta t)^j}{j!}$. However, this leads to solving a similar linear system to (2.30), which we wish to avoid. Another approach is to leverage iterative methods (*e.g.*, GMRES) when Δt is sufficiently large. Although the solution may converge within a reasonable time frame, solving similar systems needs to be repeated for all test vectors and frequencies. To overcome these challenges, we propose employing Petrov-Galerkin (or Galerkin) projection to obtain an affordable, approximate solution of (2.46).

Consider a low-dimensional representation of the transient response as

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}\boldsymbol{\beta}_1, \tag{2.47}$$

where $\boldsymbol{\phi} \in \mathbb{C}^{N\times r}$, with $r \ll N$, is an orthonormal trial basis spanning the transient response and $\boldsymbol{\beta}_1 \in \mathbb{C}^r$ represents the coefficients describing the transient in this basis. By substituting (2.47) into (2.46), the linear system

$$(\boldsymbol{I} - e^{-\mathrm{i}\omega\Delta t}e^{\boldsymbol{A}\Delta t})\boldsymbol{\phi}\boldsymbol{\beta}_1 = \boldsymbol{b} \tag{2.48}$$

is overdetermined. Petrov-Galerkin projection with test basis $\boldsymbol{\psi} \in \mathbb{C}^{N\times r}$ is employed to close (2.48), giving

$$\boldsymbol{\psi}^*(\boldsymbol{I} - e^{-\mathrm{i}\omega\Delta t}e^{\boldsymbol{A}\Delta t})\boldsymbol{\phi}\boldsymbol{\beta}_1 = \boldsymbol{\psi}^*\boldsymbol{b}. \tag{2.49}$$

Solving (2.49) for $\boldsymbol{\beta}_1$ and inserting the solution into (2.47) yields

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}(\boldsymbol{\psi}^*\boldsymbol{\phi} - e^{-\mathrm{i}\omega\Delta t}\tilde{\boldsymbol{M}})^{-1}\boldsymbol{\psi}^*\boldsymbol{b}, \tag{2.50}$$

where

$$\tilde{\boldsymbol{M}} = \boldsymbol{\psi}^* e^{\boldsymbol{A}\Delta t}\boldsymbol{\phi} \in \mathbb{C}^{r\times r} \tag{2.51}$$

is a reduced matrix that maps the coefficients. The advantage of this strategy is that it allows for the computation of the inverse of $(\boldsymbol{\psi}^*\boldsymbol{\phi} - e^{-\mathrm{i}\omega\Delta t}\tilde{\boldsymbol{M}})$ due to its reduced dimension. Obtaining $\tilde{\boldsymbol{M}}$ is also an efficient process, involving two steps: (i) integrating the columns of $\boldsymbol{\phi}$ over Δt, and (ii) projecting $e^{\boldsymbol{A}\Delta t}\boldsymbol{\phi}$ onto the columns of $\boldsymbol{\psi}$. The construction cost of $\tilde{\boldsymbol{M}}$ for each $\omega \in \Omega$ is primarily determined by the first step. Specifically, when the number of columns in $\boldsymbol{\phi}$ is $r = N_\omega$ and $\Delta t = T_s/N_\omega$, the total cost of constructing $\tilde{\boldsymbol{M}}$ for all $\omega \in \Omega$ is equivalent to integrating the LNS equations for an additional T_s duration.

Galerkin projection is a special case of the above procedure in which the test and trial functions are the same, *i.e.*, $\boldsymbol{\phi}$ is also the test function. Using this strategy with either

Galerkin or Petrov-Galerkin projections, the accuracy of the solution relies on the ability of the column space of $\boldsymbol{\phi}$ to adequately span the transient response. Thus, the challenge lies in constructing an appropriate basis to accurately capture the transient behavior. Before the introduction of appropriate trial bases, we note that one can construct a new $\boldsymbol{\phi}$ for each $\omega \in \Omega$, however, the bases that we define later are universal for all frequencies. Hence, the reduced matrix $\tilde{\boldsymbol{M}}$ is constructed once for all frequencies. Subsequently, (2.50) obtains transient responses at each frequency and updates the steady-state responses.

Given the rapid decay of most terms in the transient response, it is advantageous to utilize the least-damped eigenvectors of $\boldsymbol{A}$ as the chosen trial basis. By excluding the least-damped eigenvectors, we effectively increase the decay rate of the transient response. Let λ_1 denote the least-damped eigenvalue of $\boldsymbol{A}$, with $\boldsymbol{V}_1$ representing the corresponding eigenvector. We define $\boldsymbol{\phi} = \boldsymbol{V}_1$, thereby removing the transient component projected onto $\boldsymbol{V}_1$. As a result, the norm of the updated transient, obtained by subtracting this projection, follows the decay rate associated with the second least-damped eigenvalue of $\boldsymbol{A}$. Similarly, the trial basis $\boldsymbol{\phi}$ can encompass the first $r-1$ least-damped eigenvectors, $\boldsymbol{\phi} = \text{orth}\{\boldsymbol{V}_1, \boldsymbol{V}_2, ..., \boldsymbol{V}_{r-1}\}$, leading to a decay rate governed by the r^{th} least-damped eigenvalue of $\boldsymbol{A}$. For this particular trial basis, Petrov-Galerkin projection can be utilized, where $\boldsymbol{\psi}$ incorporates the adjoint eigenvectors. This approach ensures the complete elimination of transient projection onto the least-damped modes. To be clear, this procedure does not eliminate the impact of these modes on the steady-state response, but only on the transient response.

The main challenge associated with this trial basis is the computational cost of computing the least-damped eigenvectors (and adjoint eigenvectors in the case of Petrov-Galekin projection), especially for large systems, even when using algorithms designed for this purpose, *e.g.*, Krylov-based methods [39, 37]. Overall, the least-damped modes of $\boldsymbol{A}$ are most helpful for systems that suffer from only a few slowly decaying modes.

Another powerful trial basis is formed by stacking the snapshots into a matrix during the integration of the LNS equations, resulting in $\boldsymbol{\phi} = \text{orth}\{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, ..., \boldsymbol{q}_r\}$ (an orthogonalization of the matrix of snapshots). Specifically, $\boldsymbol{\phi}$ can be constructed as the union of $\hat{\boldsymbol{Q}}$ and $\hat{\boldsymbol{Q}}^{\Delta t}$ as a reliable trial basis. Performing QR decomposition on this matrix is essential to ensure orthogonality. As the LNS equations are allowed to run for a longer duration, $\boldsymbol{\phi}$ becomes an increasingly effective trial basis, providing improved estimates of the transient responses across all frequencies $\omega \in \Omega$. We have observed that this basis is particularly accurate for higher frequencies compared to lower ones.

A feature of our transient-removal approach is its flexibility in incorporating multiple trial bases. For instance, by considering the matrix of least-damped eigenvectors of $\boldsymbol{A}$ in $\boldsymbol{\phi}_1$ and the on-the-fly snapshots in $\boldsymbol{\phi}_2$, a combined trial basis $\boldsymbol{\phi} = \boldsymbol{\phi}_1 \cup \boldsymbol{\phi}_2$ can be constructed and orthogonalized. The combination of trial bases, with $\boldsymbol{\phi}_2$ being highly effective at higher frequencies, offers benefits at lower frequencies.

The expected transient error remaining before and after applying our transient removal approach can be estimated using a preprocessing step. We begin by integrating the homogeneous system (2.39) using a random initial condition with unit norm. By employing (2.43), (2.44), and (2.45), and assuming $\boldsymbol{q}_s = 0$, we can apply either Petrov-Galerkin or Galerkin projection to calculate the updated transient norms. This approach is feasible when $\boldsymbol{\phi}$ does not depend on real-time simulation, such as when it represents the matrix of least-damped eigenvectors. However, if $\boldsymbol{\phi}$ consists of snapshots, we must generate synthetic snapshots. To accomplish this, we set $\boldsymbol{q}_{s,0} = -\boldsymbol{q}_{t,0}$ to ensure the initial snapshot $\boldsymbol{q}_0 = \boldsymbol{q}_{s,0} + \boldsymbol{q}_{t,0}$ equals zero. Subsequent snapshots are obtained by superimposing the transient responses (from the homogeneous simulation) onto steady-state responses generated as $\boldsymbol{q}_{s,j} = e^{\mathrm{i}\omega j \Delta t}\boldsymbol{q}_{s,1}$, where Δt is the time-distance between snapshots. Using this technique, we can construct $\boldsymbol{\phi}$ for varying periods and assess the efficacy of the transient removal strategy. The updated transient error, similar to (2.36), is computed as the ratio of norms between the updated transient and steady-state responses, which monotonically decreases after the transient growth phase. This iterative process is performed for all $\omega \in \Omega$, necessitating the generation of fresh snapshots for the steady-state responses while keeping the transient response fixed. The computational expense associated with obtaining this *a priori* error estimate is primarily determined by the integration of the homogeneous system and typically constitutes less than 5% of the overall cost of executing the complete algorithm for computing the resolvent modes. We illustrate the application of this strategy using various trial bases in §2.10.

2.9 Removing the least-damped modes using eigenvalues only

The transient removal strategies described in §2.8.2 require a basis for the transient, either in the form of eigenvectors for the least-damped eigenvalues or data. In this section, we outline an alternative procedure to expedite the decay of transients that that uses knowledge of the least-damped eigenvalues themselves. Considering two solutions of (2.16), $\boldsymbol{q}_1 = \boldsymbol{q}(t_1)$ and $\boldsymbol{q}_2 = \boldsymbol{q}(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts as

$$\begin{aligned} \boldsymbol{q}_1 &= \boldsymbol{q}_{s,1} + \boldsymbol{q}_{t,1}, \\ \boldsymbol{q}_2 &= \boldsymbol{q}_{s,2} + \boldsymbol{q}_{t,2}, \end{aligned} \tag{2.52}$$

where $\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}, \boldsymbol{q}_{t,1}$, and $\boldsymbol{q}_{t,2}$ are four unknowns. The transient parts can be written as

$$\begin{aligned} \boldsymbol{q}_{t,1} &= \boldsymbol{q}_{\lambda_1,1} + \boldsymbol{q}_{rest,1}, \\ \boldsymbol{q}_{t,2} &= \boldsymbol{q}_{\lambda_1,2} + \boldsymbol{q}_{rest,2}, \end{aligned} \tag{2.53}$$

where we assume the unknowns $\boldsymbol{q}_{\lambda_1,j}$ evolve as $\sim e^{\lambda_1 t}$, where λ_1 is the least-damped eigenvalue. Hence,

$$\boldsymbol{q}_{\lambda_1,2} = \boldsymbol{q}_{\lambda_1,1} e^{\lambda_1 \Delta t}, \tag{2.54}$$

where $\boldsymbol{q}_{\lambda_1,j}$ is essentially the projection of the transient response onto the least-damped eigenmode of $\boldsymbol{A}$ at $t = t_j$. The steady-state evolution at a prescribed forcing at a single frequency ω follows (2.44). Therefore, in case of $||\boldsymbol{q}_{rest,j}|| = 0$, the system of equations is deterministic and $\boldsymbol{q}_{t,1}$ can be found as

$$\boldsymbol{q}_{t,1} = \frac{\boldsymbol{b}}{c}, \tag{2.55}$$

where $\boldsymbol{b} = \boldsymbol{q}_1 - \boldsymbol{q}_2 e^{-\mathrm{i}\omega\Delta t}$ is known from the time stepping and $c = 1 - e^{(\lambda_1 - \mathrm{i}\omega)\Delta t}$ is constant. Otherwise, *i.e.*, $||\boldsymbol{q}_{rest,j}|| \neq 0$, by simplifying terms, the transient part can be written as

$$\boldsymbol{q}_{t,1} = \frac{\boldsymbol{b}}{c} - \frac{(1-c)\boldsymbol{q}_{rest,1} - \boldsymbol{q}_{rest,2} e^{-\mathrm{i}\omega\Delta t}}{c}. \tag{2.56}$$

Based on the fundamental assumption, the second term, which is unknown, decays faster than $e^{\lambda_{1,r} t}$. Therefore, by removing the first term $\frac{\boldsymbol{b}}{c}$, which is known, the residual eventually follows the second least-damped eigenvalue. If the forcing term encompasses a range of frequencies, the same relationships remain valid for each frequency after undergoing a DFT, and $\frac{\boldsymbol{b}}{c}$ can be separately eliminated for each $\omega \in \Omega$. Note that the eigenvector associated with λ_1 was never used.

This procedure can be generalized to target the d least-damped eigenmodes of $\boldsymbol{A}$. The solution at each time with arbitrary distances can be expanded as

$$\boldsymbol{q}_l = \boldsymbol{q}_{s,l} + \sum_{j=1}^{d} \boldsymbol{q}_{\lambda_j,l} + \boldsymbol{q}_{rest,l}, \tag{2.57}$$

for $1 \leq l \leq d+1$. Utilizing the same relationships, we can eliminate the slowest components, ensuring that the residual term decays faster than all d modes. This procedure is developed to steepen the decay rate and shorten the transient length to meet the desired accuracy. The outcomes of this procedure closely resemble the output of the efficient transient strategy using Galerkin projection with the least-damped eigenmodes as the basis. The transient error can be estimated in a similar manner as described for the projection-based approach.

2.10 Test cases

In this section, the RSVD-Δt algorithm is tested using two problems. First, the accuracy of the algorithm and the effectiveness of the transient removal strategy are verified using the complex Ginzburg-Landau equation. Second, the computational efficiency and scalability of the algorithm are demonstrated and compared to that of the RSVD-LU algorithm using a three-dimensional discretization of a round jet.

2.10.1 Complex Ginzburg-Landau equation

The complex Ginzburg-Landau equation was initially derived for analytical studies of Poiseuille flow [146] and has subsequently been used more generally as a convenient model of a flow susceptible to non-modal amplification [65, 7, 22, 20]. Here, we use it as an inexpensive test case to validate our algorithm. The complex Ginzburg-Landau system follows the form of (2.3) with

$$\begin{gathered} \boldsymbol{A} = -\nu \frac{\partial}{\partial x} + \gamma \frac{\partial^2}{\partial x^2} + \mu(x), \\ \mu(x) = (\mu_0 - c_\mu^2) + \frac{\mu_2}{2} x^2, \\ \boldsymbol{B} = \boldsymbol{C} = \boldsymbol{I}. \end{gathered} \tag{2.58}$$

Following Bagheri *et al.* [7], we set $\gamma = 1 - \mathrm{i}, \nu = 2 + 0.2\mathrm{i}, \mu_0 = 0.38, c_\mu = 0.2$, and $\mu_2 = -0.01$. These parameters ensure global stability and provide a large gain separation between the leading mode and the rest of the modes at the peak frequency [7]. To explicitly build the $\boldsymbol{A}$ operator, a central finite difference method is used to discretize $x \in [-100, 100]$ using $N = 500$ grid points. The domain is sufficiently extended in both $\pm x$ directions such that it resembles infinite boundaries [7], and the weight matrix $\boldsymbol{W}$ is set to the identity on account of the uniform grid.

2.10.1.1 RSVD-Δt validation: assessing the transient and truncation errors

The RSVD-Δt outcome must replicate the RSVD-LU outcome up to machine precision when cutting both sources of errors described in §2.7.2. Truncation error depends on the integration scheme and the time step, while the transient error depends on the length of the simulation. Therefore, using a tiny time step with a high-order integration scheme and a lengthy transient duration should eliminate the errors due to time integration.

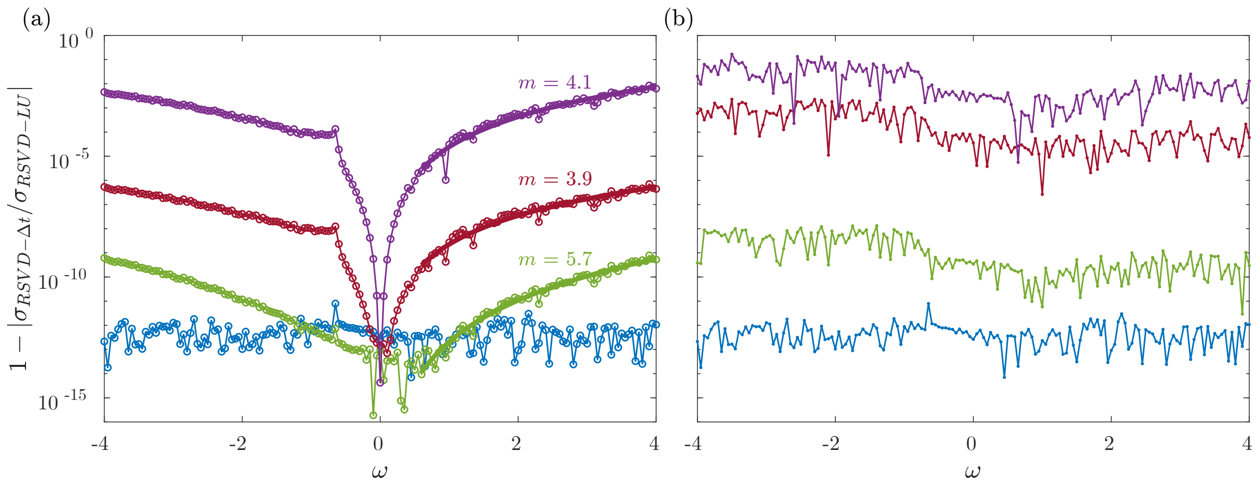


Figure 2.5: Relative error between gains computed using the RSVD-LU and RSVD-Δt algorithms for the Ginzburg-Landau problem: (a) $T_t = 5000$ and $\{\text{TSS}, dt\} = \{\text{BDF4}, 0.1\}$ (purple), $\{\text{BDF4}, 0.01\}$ (red), (BDF6, 0.01) (green), and $\{\text{BDF6}, 0.001\}$ (blue) varies; (b) $\{\text{BDF6}, 0.001\}$ is fixed and T_t varies as 500 (purple), 1000 (red), 2500 (green), and 5000 (blue). In (a), the exponents m are shown for the best-fit exponential within $\omega \in [0.6, 4]$.

Time-stepping errors are investigated by setting the number of test vectors to $k = 1$ and power iterations to $q = 0$. These minimal values are used since including additional test vectors or power iterations have no effect on the time-stepping error. The desired set of frequencies is $\Omega \in [-4, 4]$ with $\Delta\omega = 0.05$. The gains of the Ginzburg-Landau system are computed using RSVD and RSVD-Δt and the relative errors for various cases are shown in figure 2.5. The minimum error is near machine precision when BDF6, $dt = 10^{-3}$, and $T_t = 5000$ is used, validating the RSVD-Δt algorithm.

By decreasing the order of the integration scheme or increasing the time step, the truncation error becomes larger, and hence, the error in the computed gains becomes larger. In figure 2.5(a), the transient length is held fixed at $T_t = 5000$ and the gains are obtained using $\{\text{BDF6}, dt = 10^{-2}\}$, $\{\text{BDF4}, dt = 10^{-2}\}$, and $\{\text{BDF4}, dt = 10^{-1}\}$. For all four cases, the relative error is around $O(10^{-13})$ at $\omega = 0$, confirming that the transient effect is negligible. Moving away from zero frequency, the errors increase like $O(\omega^{\sim 4})$ and $O(\omega^{\sim 6})$ for the BDF4 and BDF6 schemes, respectively, consistent with the theoretical asymptotic estimates in §2.7.2.

Figure 2.5(b) displays how the length of time that the transient is allowed to decay can affect the accuracy of the gains as a function of frequency. This time, the time-stepping scheme of $\{\text{BDF6}, dt = 10^{-3}\}$ is held fixed, ensuring negligible truncation error, and the transient lengths are varied as $T_t = \{500, 1000, 2500, 5000\}$. Smaller values of T_t leave more transient residual in the steady-state response. The resulting relative gain errors show that

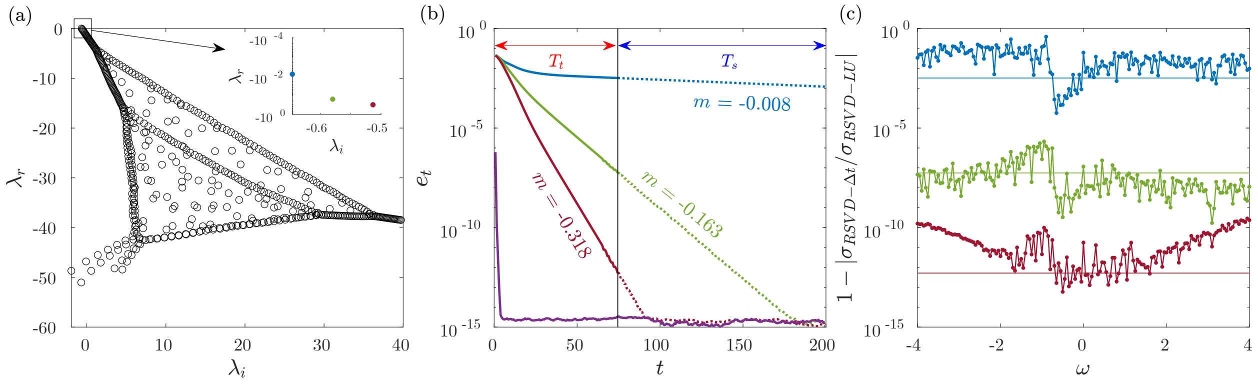


Figure 2.6: Transient-removal for the Ginzburg-Landau test problem: (a) Spectrum of Ginzburg-Landau operator with a zoomed-in view of the three least-damped eigenvalues. (b) Transient error measurement: blue curve represents original decay, while green, red, and purple curves depict decay using Galerkin projection with $\boldsymbol{\phi}$ of $\boldsymbol{V}_1$, $\{\boldsymbol{V}_1, \boldsymbol{V}_2\}$, and a matrix of snapshots, respectively. (c) Relative error comparison between the RSVD-Δt and RSVD-LU algorithms. Solid horizontal lines in (c) represent the expected transient error arising from the transient norm at the end of the T_t (the black vertical line in (b)).

the whole frequency spectrum is affected quite similarly. Longer transient lengths lead to smaller gain errors with a similar trend. The frequency distribution of the transient error depends on the eigenspectrum of the system. For example, a cluster of weakly damped modes around a specific frequency can lead to a peak transient error localized at the same frequency. In §2.10.2, the peak transient for the jet flows occurs near zero frequency.

2.10.1.2 Efficient transient removal

In this section, we demonstrate the transient removal strategy proposed in §2.8.2. We apply this strategy to the same Ginzburg-Landau system for the same Ω range described above and compare the results to the RSVD-LU results as a reference.

The eigenspectrum of the Ginzburg-Landau operator is shown in figure 2.6(a), and the three least-damped (and thus slowest decaying) modes have decays rates of $\lambda_{1,r} = -0.008, \lambda_{2,r} = -0.163$, and $\lambda_{3,r} = -0.318$, respectively, where the subscript r indicates the real part of the eigenvalue λ. Figure 2.6(b) depicts the transient norm as a function of time, where ϵ is measured as follows: we initially obtain the *true* steady-state solution by integrating (2.16) for a very long time at $\omega = 0.5$ (similar results for other frequencies), ensuring that the natural decay has eliminated the transient response to machine precision and use the steady-state response to measure the transient errors.

The natural decay in this system occurs slowly, as illustrated in figure 2.6(b). By defining $\boldsymbol{\phi}_1$ as $\boldsymbol{V}_1$ and utilizing Galerkin projection, we remove the fraction of the transient decaying

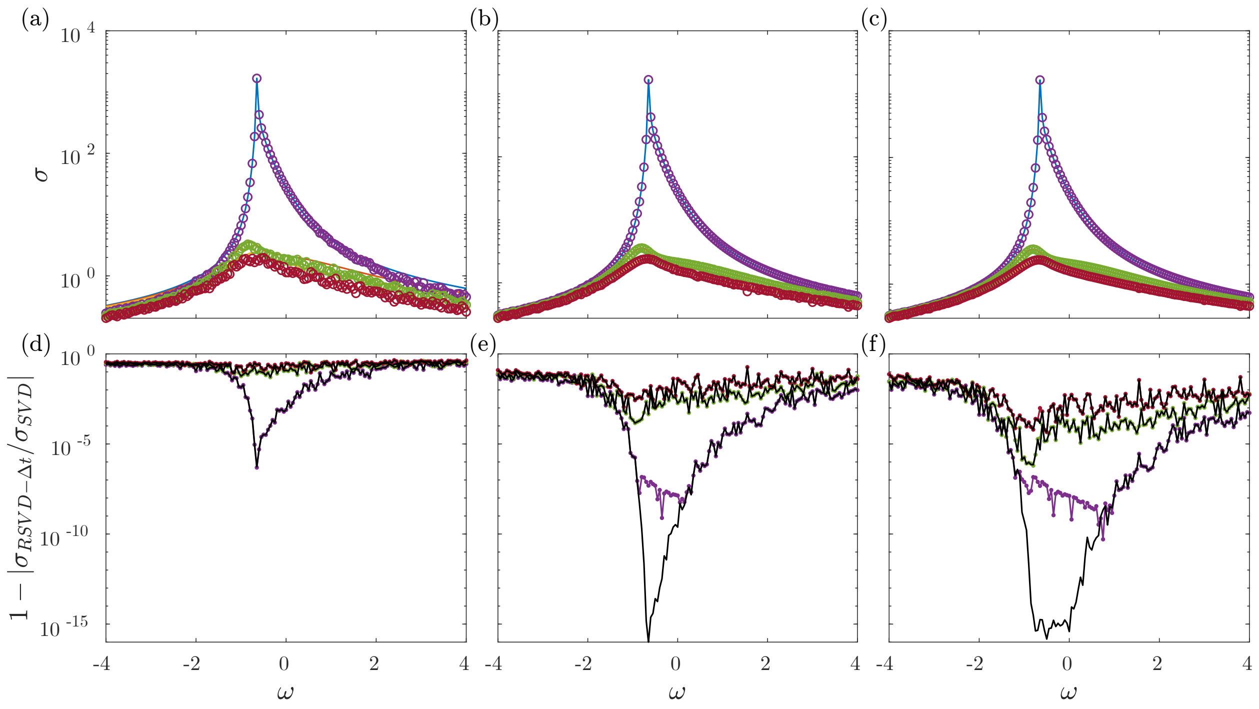


Figure 2.7: Impact of power iteration on the Ginzburg-Landau gains: (a-c) the gains of the first three optimal modes using SVD (line) and RSVD-Δt (circle); and (d-e) the relative error between them. (a,d), (b,e), and (c,f) correspond to q of 0, 1, and 2, respectively. Black lines in (d-f) show the relative error between the RSVD-LU algorithm and SVD for reference.

at the rate of $e^{\lambda_1 t}$, resulting in a noticeable change in the decay slope. Including the two least-damped modes with $\boldsymbol{\phi}_2 = \{\boldsymbol{V}_1, \boldsymbol{V}_2\}$ further steepens the decay rate, aligning closely with the corresponding least-damped eigenvalues shown in figure 2.6(a). However, it is the matrix of snapshots that proves to be the most effective, completely eliminating the transient within a short period of time.

We employ $\{\text{BDF6}, dt = 10^{-2}\}$ to compute gains using RSVD-Δt, considering three cases of transient removal that are halted at $T_t = 75$: (i) natural decay, (ii) Galerkin projection with $\boldsymbol{\phi}_1$, and (iii) Galerkin projection with $\boldsymbol{\phi}_2$. The error is measured as the relative difference in gain between the RSVD-LU and RSVD-Δt algorithms, as depicted in figure 2.6(c). The plot clearly illustrates that smaller transient errors lead to reduced gain errors. In the first two cases, the transient error dominates, while in the third case, the transient error balances with the truncation error at lower frequencies, with truncation dominating at higher frequencies. Our findings indicate that the matrix of snapshots is an effective basis for representing and removing the transient.

2.10.1.3 Impact of power iteration

Finally, we explore the impact of the number of power iterations q on the accuracy of the solution. For both the RSVD-LU and RSVD-Δt algorithms, we set $k = 6$ and vary q from 0 to 2. Additionally, RSVD-Δt uses a BDF4 integrator with $dt = 0.001$ and $T_t = 100$, and transients are reduced by removing the least-damped eigenvalue, leading to an expected overall time-stepping error of $O(10^{-8})$ according to Figure 2.6(c). A standard Arnoldi-based approach is used to provide a ground-truth reference for defining the error.

The leading three singular values and corresponding relative errors are shown in figure 2.7. One power iteration leads to a noticeable accuracy improvement. As expected, using one or more power iterations substantially improves the accuracy of both the RSVD-LU and RSVD-Δt algorithms. The optimal singular value in particular improves dramatically for frequencies with a large gap between the optimal and suboptimal modes. The RSVD-LU errors approach machine precision near the peak frequency, while the RSVD-Δt errors saturate at the floor set by the choice of integration parameters. For the rest of the modes and frequencies, the relative error between the RSVD-LU and RSVD-Δt algorithms is smaller than the relative error between the RSVD-LU algorithm and the ground truth, so the relative errors are identical. We have found using one power iteration to be sufficient for most problems, and we recommend this as a default value for our algorithm.

2.10.2 Round turbulent jet

A round jet is used to demonstrate the reduced cost and improved scaling of our algorithm. The mean flow is obtained from a large eddy simulation (LES) using the "CharLES" compressible flow solver developed by Cascade Technologies [13, 14], for Mach number $M = \frac{U_j}{a} = 0.4$ and Reynolds number $Re = \frac{U_j D_j}{\nu_j} = 0.45 \times 10^6$. Here, U_j is the mean centerline velocity at the nozzle exit, a is the ambient speed of sound, ν_j is the kinematic viscosity at the nozzle exit, and D_j is the diameter of the nozzle. Validation of the LES simulation against experimental results and more details on the numerical setup are available in Brès *et al.* [14].

The computation of the three-dimensional resolvent modes is performed within a region of interest defined by $x \in [0, 20]$ and $y \times z \in [-4, 4] \times [-4, 4]$. The spatial discretization of this region is accomplished using a grid with dimensions of $400 \times 140 \times 140$, respectively. The mean flow is obtained by revolving the axisymmetric mean flow around the streamwise axis, as depicted in figure 2.8. The domain is large enough to accommodate sizable low-frequency structures, and the mesh is resolved to capture structures that emerge in the response modes up to Strouhal (St) number of 1, where $St = \frac{\omega D_j}{2\pi U_j}$ is the non-dimensional form of frequency.

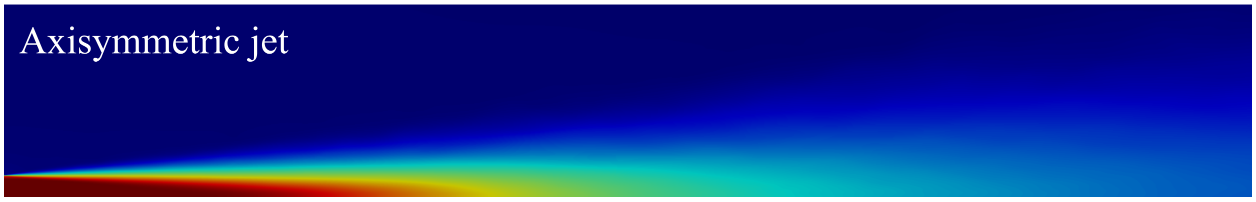


Figure 2.8: The mean streamwise velocity of the axisymmetric jet.

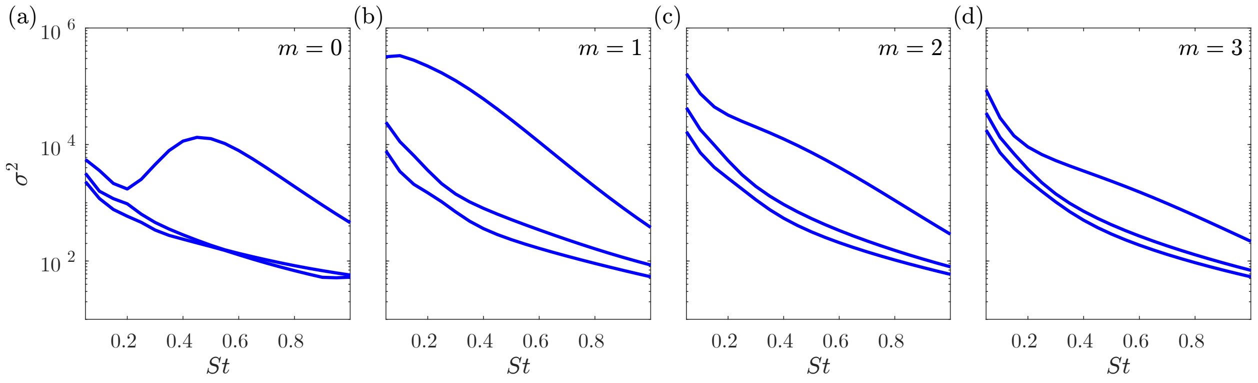


Figure 2.9: Three leading gains of the axisymmetric jet for four azimuthal wavenumbers.

The range of $St \in [0, 1]$ is wide enough to include the most important physical phenomena captured by resolvent analysis [137]. The effective Re is reduced to 1000 to account for unmodeled Reynolds stresses [112] and the effect of Re is thoroughly investigated and reported in Schmidt *et al.* [136].

The LNS equations are expressed in terms of specific volume, the three velocity components, and pressure, which can be compactly represented as $\boldsymbol{q}(\boldsymbol{x}, t) = [\boldsymbol{\xi}, \boldsymbol{u}_x, \boldsymbol{u}_r, \boldsymbol{u}_\theta, \boldsymbol{p}]^T(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{\theta}, t)$. The three-dimensional state in the frequency domain is

$$\boldsymbol{q}'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t) = \sum_{\omega} \hat{\boldsymbol{q}}_\omega(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) e^{\mathrm{i}\omega t}, \tag{2.59}$$

and each mode is characterized by its frequency ω.

To validate our three-dimensional results, we also perform a axisymmetric resolvent analysis of the same jet for a set of azimuthal wavenumbers in which the symmetry in the azimuthal direction is exploited. The mean flow is obtained on the symmetry plane with cylindrical coordinates (x, r). The axisymmetric state

$$\boldsymbol{q}'(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{\theta}, t) = \sum_{m,\omega} \hat{\boldsymbol{q}}_{m,\omega}(\boldsymbol{x}, \boldsymbol{r}) e^{\mathrm{i}m\boldsymbol{\theta}} e^{\mathrm{i}\omega t} \tag{2.60}$$

is characterized by the pair (m, ω), where m denotes azimuthal wavenumber. The domain

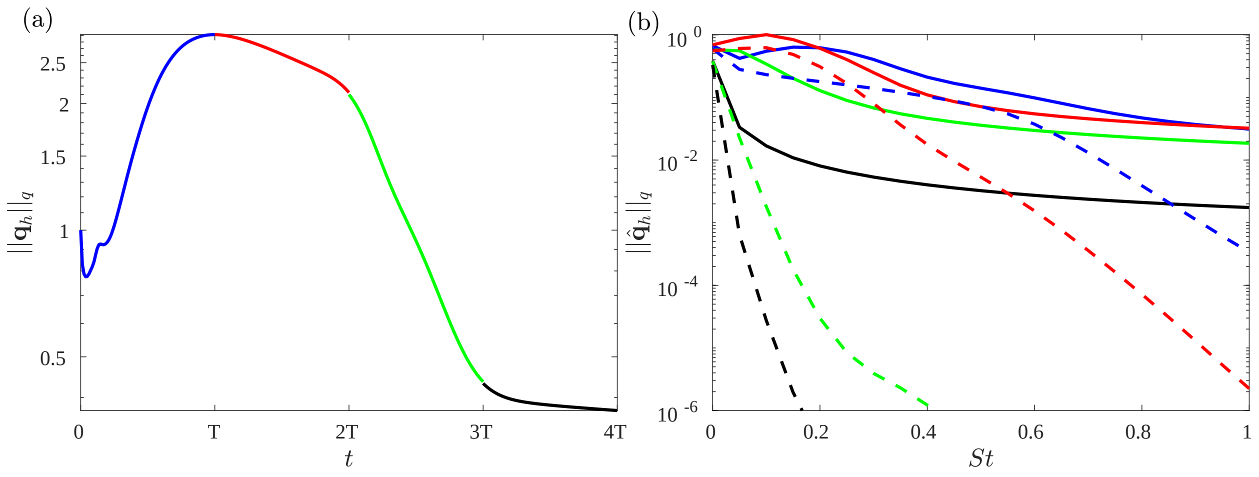


Figure 2.10: Transient error estimates for the jet in (a) the time domain and (b) the frequency domain. Each colored period represents the duration utilized for obtaining norms in the frequency domain as shown in (b). Solid lines represent the natural decay, while dashed lines correspond to the transient removal strategy using Galerkin projection with the matrix of snapshots.

of interest for resolvent analysis is $x \times r \in [0, 20] \times [0, 4]$ surrounded by a sponge region and is spatially discretized using fourth-order summation by parts finite differences [93] with 400×100 grid points in the streamwise and radial directions, respectively. A grid-convergence study verifies the relative error between gains with this mesh and twice the number of grid points is less than 1-10% for $0 \leq St \leq 1$. The remaining parameters are kept the same as in the three-dimensional discretization of the jet.

Figure 2.9 shows the gains (squared singular values) for $m = 0, 1, 2, 3$. The dominant mechanisms for each wavenumber are analyzed in detail in Schmidt *et al.* [136] and Pickering *et al.* [112]. The optimal mode when $m = 0, St \geq 0.2$ corresponds to KH instability. At $m = 0$, the KH modes are overtaken by Orr-type modes for $St < 0.2$. At $m > 0$, streaks become the dominant response and continue to prevail as the primary instability at low frequencies $St \to 0$. The KH modes remain the most amplified response for the higher St-range when $m > 0$, causing the large separation between the leading mode and suboptimal modes.

Similar gain trends are found in Schmidt *et al.* [137] and Pickering *et al.* [111] for the same wavenumbers demonstrating the robustness of the outcome even though the computational domains, Re, state vector, sponge regions, and boundary conditions are slightly different. The gains and corresponding modes of the axisymmetric jet are used as a baseline for comparison to the three-dimensional jet.

2.10.2.1 Resolvent modes for the jet

Resolvent modes for the three-dimensional round jet are computed for the same range of $St \in [0,1]$ with $\Delta St = 0.05$. The six leading modes are of interest, so we set $k = 10$ and $q = 1$. For the RSVD-Δt algorithm, we use the classical 4^{th} order Runge–Kutta (RK4) integrator with $dt = 0.00625$. The steady-state interval is $T_s = 20$. Figure 2.10 shows the expected transient error in the time and frequency domains. The transient initially grows in time before slowly decaying in figure 2.10(a). The resulting error in the frequency domain obtained from selecting each colored segment for computing resolvent modes is shown in figure 2.10(b). Our transient removal strategy, using Galerkin projection with the matrix of snapshots, drastically reduces these errors for $St > 0$, as indicated by the dashed lines. We select a transient duration of $T_t \approx 2T_s$ (green segment), for which the transient removal strategy brings the transient error below 1% for $St > 0$.

Figure 2.11 compares the gains of two-dimensional and three-dimensional discretizations of the jet. Due to the azimuthal symmetry of the problem, the gains of the three-dimensional problem are expected to be the union of the gains from the axisymmetric problem [141]. Since higher wavenumbers ($m > 3$) have lower gains [112], the union of the first four azimuthal wavenumbers is enough to match the leading modes of the three-dimensional system. The azimuthal symmetry makes modes corresponding to $m \neq 0$ appear in pairs for the three-dimensional problem. The six computed modes appear in pairs for $St \leq 0.3$, after which the gain of the $m = 0$ mode becomes large enough to appear for the three-dimensional problem. Up to $St = 0.8$, the largest gains are associated with $m = \pm 1$. All of the modes that appear for the three-dimensional problem are KH modes; many more resolvent modes would need to be computed to capture Orr modes that are buried beneath a slew of KH modes for each azimuthal wavenumber. The close match between the computed three-dimensional modes and the set of two-dimensional modes verifies that the three-dimensional calculations are properly capturing the known physics for this problem. The small mismatch at frequencies close to $St = 1$ is due to mild under-resolution of the grid for the compact structures that appear at these frequencies.

Figure 2.12 shows the pressure response modes at four (St, m) pairs (other components such as velocity yield similar observations). Each panel shows, for one (St, m) pair, contours of the two-dimensional mode computed leveraging symmetry, isocontours of the corresponding three-dimensional mode, and contours for cross sections of the three-dimensional mode in the $x - y$ and $y - z$ planes. These images show the wavepacket form of the modes, confirm the classification of each three-dimensional mode with a particular azimuthal wavenumber, and illustrate the match between the symmetric and three-dimensional results. As noted by Martini *et al.* [92], symmetries such as the azimuthal homogeneity of the jet produce pairs of

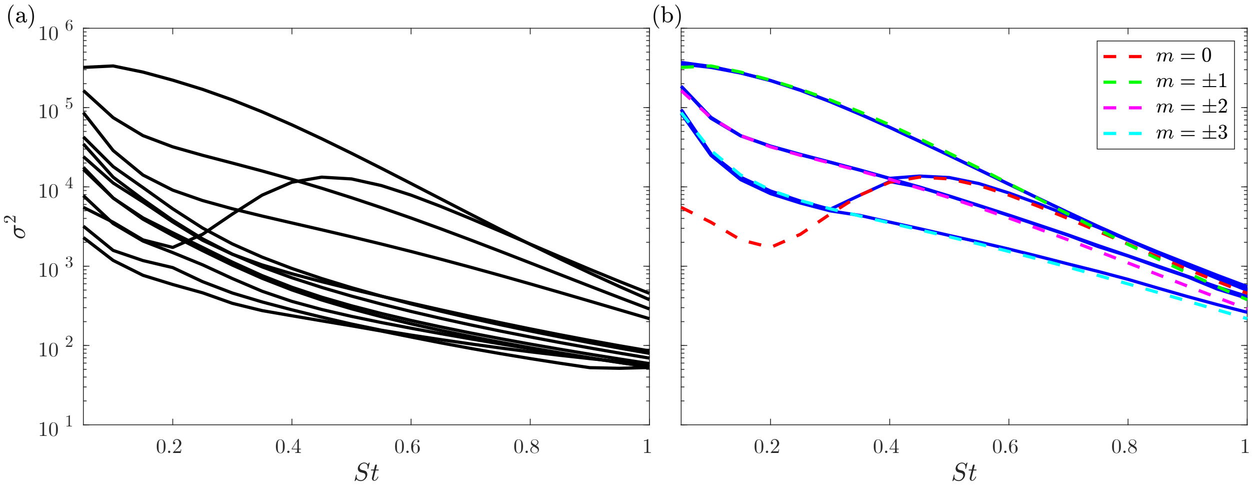


Figure 2.11: Resolvent gains for the jet: (a) the union of the axisymmetric jet gains; (b) the optimal gains of the axisymmetric jet corresponding to various values of m (dashed lines) overlaid on top of the six leading gains for the three-dimensional discretization (solid lines).

modes with equal gain that can be arbitrarily combined (under the constraint of orthogonality) to produce equally valid mode pairs. For visualization purposes, we have adjusted the phase and summed the mode pairs to best match those of the modes from the axisymmetric calculations.

2.10.2.2 Computational complexity comparison

We showcase the superior computational efficiency and scalability of the RSVD-Δt algorithm compared to the RSVD-LU algorithm using the three-dimensional jet by varying the discretized state dimension N. We set $k = 10$, $N_\omega = 21$, and $q = 0$ for both algorithms and $dt = 0.00625$, $T_t = 2T_s$, and $T_s = 20$ in the RSVD-Δt algorithm as in §2.10.2.1. The reported costs for the RSVD-LU algorithm includes only a single LU decomposition and the two solutions of the LU decomposed system (once for the direct system and once for the adjoint system) at each frequency of interest, highlighting the LU decomposition as the primary bottleneck in the RSVD-LU algorithm and similar methods utilizing LU decomposition to solve (2.30). The reported costs encompasses the entire RSVD-Δt algorithm with a total integration length of $4T_s$ per action, including one extra period to account for the transient removal strategy, as explained in §2.8.2. The RSVD-Δt algorithm is implemented using PETSc [8], while the LU decomposition in the RSVD-LU algorithm utilizes PETSc in conjunction with the MUMPS [3] external package. All calculations are performed on one processor such that wall-time functions as a proxy for CPU time.

The measured CPU time for both algorithms are shown in figure 2.13(a) as a function of

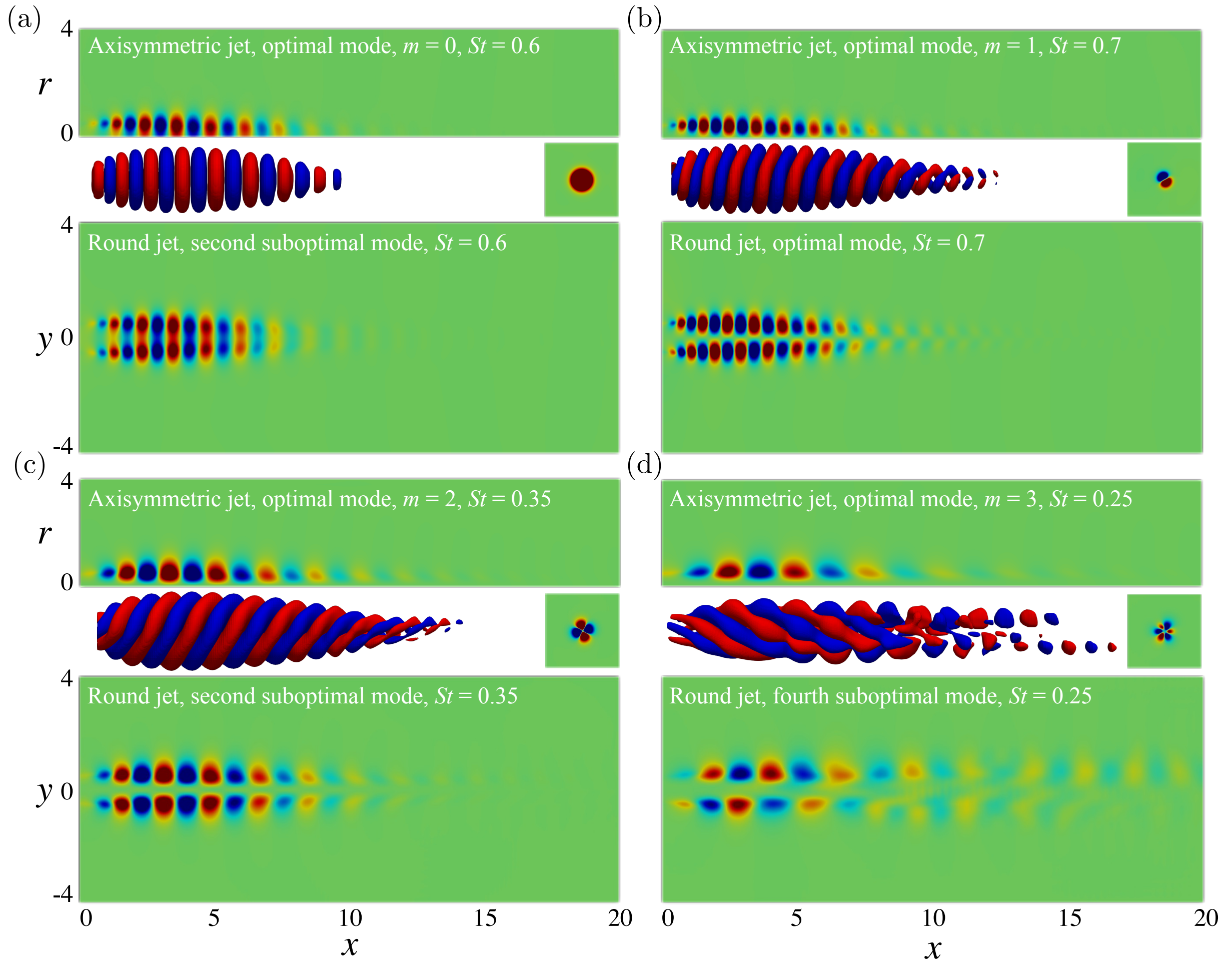


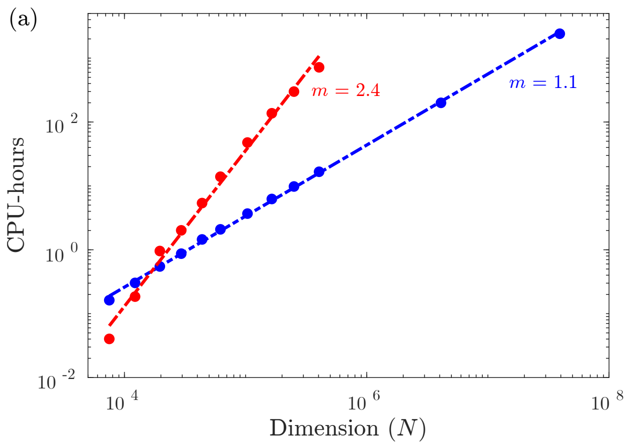
Figure 2.12: Four groups of axisymmetric and three-dimensional pressure modes are shown, including axisymmetric views, three-dimensional iso-volume representations, and $x-y$ plane snapshots of the round jet. Cross-sections at x = 5 confirm the azimuthal wavenumber of the three-dimensional results. Color bar ranges are adjusted for visualization.

the state dimension N. The RSVD-LU algorithm scales poorly, in fact exceeding the theoretical scaling of $O(N^2)$ for three-dimensional flows (refer to §2.6) due to poor performance at low frequencies that has also been noted in other studies [111]. In contrast, the RSVD-Δt algorithm achieves (near) linear scaling, $O(N^{1.1})$, confirming its scalability to large problems. The calculations could not be performed using RSVD-LU for the largest two grids dimensions exceeding 1 million require an excessive amount of memory beyond our cluster limits when employing RSVD-LU. Therefore, these two data points are exclusively reserved for RSVD-Δt to validate the linear scaling retention during the transition to more realistic dimensions. The final point corresponds to the same grid utilized in our three-dimensional jet flow test case.

Similar observations can be made about the memory requirements of the two algorithms, shown in figure 2.13(b). The observed $O(N^{1.5})$ memory scaling for the RSVD-LU algorithm is better than the CPU counterpart, but it is still the main barrier to applying the RSVD-LU algorithm when the state dimension is of the order of 10 million or higher. The RAM peak usage is determined entirely by LU decomposition and drops after the decomposed matrices are obtained. On the other hand, the memory scaling for the RSVD-Δt algorithm is exactly linear with the state dimension N, consistent with the theoretic scaling determined in §2.6. Results could not be obtained for the largest two grids using RSVD-LU due to its memory requirements exceeding that of our cluster.

Most of the grids considered in figure 2.13 are under-resolved to make the RSVD-LU calculations tractable. To compare the two algorithms for a realistic grid, Table 4.1 compares the costs of RSVD-LU and RSVD-Δt for $N \approx 39$ million (5 state variables $\times$ a $[400 \times 140^2]$ grid), which was used for the three-dimensional calculations in §2.10.2.2, and $N_\omega = 21$, $k = 10$, and $q = 1$. The CPU and memory requirements of the RSVD-LU algorithm are intractable for this problem, so we estimate these costs by extrapolating the best-fit lines in figure 2.13. Computing the action of the resolvent operator in the RSVD-LU algorithm involves both LU decomposition and solving the decomposed system, with both being extrapolated but the latter not depicted in figure 2.13. This implies that for $q = 1$, the CPU time includes a single LU decomposition and four times solving the LU-decomposed system. On the other hand, for RSVD-Δt, the CPU time and memory usage are directly taken from our simulation, which employed 300 cores.

The RSVD-LU algorithm exhibits a CPU time that is more than three orders of magnitude higher than that of the RSVD-Δt algorithm. Specifically, using 300 cores, the wall-time for RSVD-Δt is approximately 61 hours (< 3 days), while the RSVD-LU algorithm requires over 75 300 000 CPU-hours, which translates to around 251 000 hours (~ 28 years) wall-time, assuming perfect linear speed-up using 300 parallel cores. Of course, this wall-time can



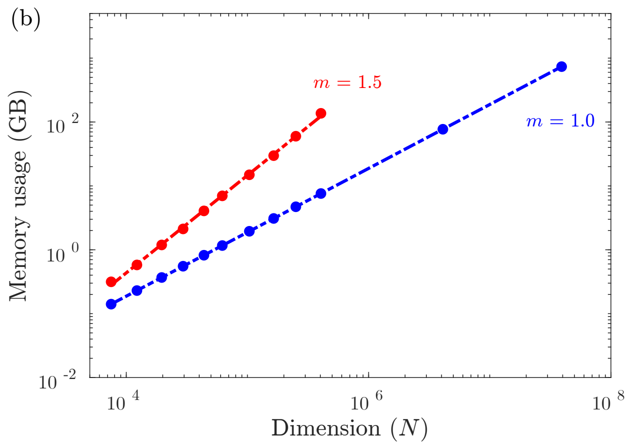


Figure 2.13: Computational cost as a function of the state dimension N for the three-dimensional jet: (a) CPU-hours and (b) memory usage for the RSVD-LU (red) and RSVD-Δt (blue) algorithms.

be brought down by increasing the number of cores, but it is clear that super-computing resources would be required to make the wall-time acceptable, which is antithetical to role of resolvent analysis as a reduced-order model. This disparity becomes even more pronounced as N increases due to the linear CPU scaling of RSVD-Δt and the quadratic scaling of the RSVD-LU algorithm for three-dimensional problems. Table 4.1 confirms that the time-stepping process accounts for nearly all of the CPU time in RSVD-Δt.

The memory improvements of the RSVD-Δt algorithm are arguably even more important. The memory usage in the RSVD-LU algorithm exceeds that of RSVD-Δt by more than two orders of magnitude. The minimum memory requirement for LU calculations surpasses 130 TB for the three-dimensional jet flow. This amount of memory is more than can be accessed even on most high-performance-computing clusters. In contrast, the memory usage in RSVD-Δt is optimized to store only three matrices of size $N \times k \times N_\omega$, which can be accurately estimated based on the size of each float number in C/C++. For instance, with $N \approx 39$ million, $k = 10$, and $N_\omega = 21$, the RAM consumption for these matrices amounts to ~ 0.75 TB (using double precision with 64-bit indices). Moreover, the RAM requirements of our algorithm can be further reduced at the expense of higher CPU cost if necessary as proposed in §2.8.1.3, while no such trade-off exists for the RSVD-LU algorithm.

2.11 Chapter summary

In conclusion, this chapter has highlighted the effectiveness of the RSVD-Δt algorithm in advancing resolvent analysis, particularly for large-scale fluid flows. By validating its accuracy and demonstrating its scalability through three-dimensional flow simulations, we have shown that the algorithm offers significant improvements in computational efficiency. These advancements position the RSVD-Δt algorithm as a practical tool for researchers and engineers working on complex flow problems, paving the way for more efficient and accurate analyses in fluid dynamics.

CHAPTER 3

Harmonic Resolvent Analysis

Harmonic resolvent analysis is a mathematical method used to study the dynamics of fluid flows that are linearized around a periodic mean flow. This technique helps examine interactions between different frequency components, offering insights into the behavior of fluid flows. The harmonic resolvent operator, a key element of this method, linearly maps forcing inputs to perturbations, uncovering the underlying mechanisms driving flow dynamics. Although it can be computationally intensive, harmonic resolvent analysis is valuable for understanding complex fluid dynamics. Challenges include managing the coupling of multiple frequencies into a single operator and addressing potential singularities or near-singularities. Variations such as cross-frequency amplification and subharmonic resolvent further refine the analysis by focusing on specific aspects of these interactions.

In this chapter, we address the challenges of harmonic resolvent analysis using the RSVD-Δt algorithm. This algorithm effectively handles the computational difficulties related to the coupling of multiple frequencies and potential singularities by employing a time-stepping approach for the time-periodic linearized Navier-Stokes operator. While the memory optimization strategy follows a concept similar to that used in resolvent analysis, we introduce a novel approach for removing unwanted transients to minimize the simulation length. Our algorithm is validated against ground truth results from a periodic Ginzburg-Landau system, and we demonstrate its effectiveness by computing harmonic resolvent modes for the flow over an airfoil, matching the test case from Padovan *et al.* [106]. Empirical results confirm the scalability of our algorithm, with CPU and memory costs scaling linearly with the problem dimension.

3.1 Derivation

In this section, we present the formulation of harmonic resolvent analysis and derive its operator by truncating the inherently infinite-dimensional system. We also explore various aspects of the analysis, including its singular nature and key variations.

3.1.1 Formulation

Analogous to resolvent analysis, by inserting the generalized Reynolds decomposition $\boldsymbol{q}(t) = \bar{\boldsymbol{q}}(t) + \boldsymbol{q}'(t)$ into the Navier-Stokes equations, the linearization around a periodic base flow $\bar{\boldsymbol{q}}(t) = \bar{\boldsymbol{q}}(t+T)$ leads to the linear time-periodic system

$$\frac{d\boldsymbol{q}'}{dt} = \boldsymbol{A}_p\boldsymbol{q}' + \boldsymbol{f}, \tag{3.1}$$

where $\boldsymbol{A}_p(t) = \boldsymbol{A}_p(t+T) \in \mathbb{C}^{N\times N}$ is the periodic LNS operator, ω_f is the fundamental frequency, and $T = 2\pi/\omega_f$ is defined as the fundamental cycle length. As both $\boldsymbol{A}_p$ and $\boldsymbol{q}'$ are time-periodic, the Fourier transform of (3.1) incorporates interactions between all pairs of frequencies. For simplicity, we drop the prime notation for fluctuations from here on out.

Expanding $\boldsymbol{A}_p(t)$ and $\boldsymbol{q}(t)$ in terms of the Fourier series

$$\begin{aligned} \boldsymbol{A}_p(t) &= \sum_{j=-\infty}^{\infty} \hat{\boldsymbol{A}}_{p,j} e^{\mathrm{i}j\omega_f t}, \\ \boldsymbol{q}(t) &= \sum_{j=-\infty}^{\infty} \hat{\boldsymbol{q}}_j e^{\mathrm{i}j\omega_f t}, \end{aligned} \tag{3.2}$$

where $(\cdot)_j$ denotes j^{th} harmonic of ω_f, and substituting these expansions into (3.1), we obtain

$$[\boldsymbol{T}\hat{\boldsymbol{q}}]_k = \mathrm{i}k\omega_f\hat{\boldsymbol{q}}_k - \sum_{j=-\infty}^{\infty} \hat{\boldsymbol{A}}_{k-j}\hat{\boldsymbol{q}}_j = \hat{\boldsymbol{f}}_k. \tag{3.3}$$

Here, $\boldsymbol{T}$ is an infinite dimensional block matrix of the form

$$\boldsymbol{T} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \dots & \boldsymbol{L}_{-1} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & -\hat{\boldsymbol{A}}_{-3} & \dots \\ \dots & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{0} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & \dots \\ \dots & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{1} & -\hat{\boldsymbol{A}}_{-1} & \dots \\ \dots & -\hat{\boldsymbol{A}}_{3} & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{2} & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{3.4}$$

where the diagonal entries are in fact the inverse of resolvent operators at various frequencies,

$$\boldsymbol{L}_j = \boldsymbol{R}_j^{-1} = \mathrm{i}j\omega_f\boldsymbol{I} - \hat{\boldsymbol{A}}_{p,0}, \forall j \in \mathbb{Z}. \tag{3.5}$$

The harmonic resolvent operator

$$\boldsymbol{H} = \boldsymbol{T}^{-1} \tag{3.6}$$

transfers the harmonic inputs to the outputs for periodic base flows,

$$\hat{\boldsymbol{q}} = \boldsymbol{H}\hat{\boldsymbol{f}}. \tag{3.7}$$

The SVD of the harmonic resolvent operator

$$\boldsymbol{H} = \mathbf{U}_H \boldsymbol{\Sigma}_H \mathbf{V}_H^* \tag{3.8}$$

unveils the most amplified responses $\mathbf{U}_H$ that correspond to optimal forcing $\mathbf{V}_H$, each accompanied by associated amplification magnitudes $\boldsymbol{\Sigma}_H$. The matrix $\boldsymbol{H}$ encompasses the base flow, the fundamental frequency, and higher harmonics. Whereas individual frequencies can be analyzed separately in a standard resolvent analysis, all frequencies are coupled within $\boldsymbol{H}$ in harmonic resolvent analysis and must be considered simultaneously. Furthermore, the modes of the modified harmonic resolvent operator, defined as

$$\tilde{\boldsymbol{H}} = \boldsymbol{W}_q^{1/2} \boldsymbol{C} \boldsymbol{H} \boldsymbol{B} \boldsymbol{W}_f^{-1/2}, \tag{3.9}$$

can be computed using our algorithm with the same minor adjustments as detailed in Farghadan *et al.* [44]. In a special case where the base flow is time-independent, *i.e.*, $\hat{\boldsymbol{A}}_{p,j} = 0$ for $j \neq 0$, $\boldsymbol{H}$ reduces to

$$\boldsymbol{H} = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & ⋰ \\ \dots & \boldsymbol{R}_{-1} & 0 & 0 & \dots \\ \dots & 0 & \boldsymbol{R}_0 & 0 & \dots \\ \dots & 0 & 0 & \boldsymbol{R}_1 & \dots \\ ⋰ & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{3.10}$$

so each frequency is decoupled. Therefore, resolvent analysis can be perceived as a special case of harmonic resolvent analysis.

3.1.2 Singular or near-singular nature of $\boldsymbol{T}$

The operator $\boldsymbol{T}$ tends to be ill-conditioned for many periodic systems [106]. A system is considered ill-conditioned when the ratio of the largest and smallest singular values is large ($\kappa = \sigma_{\max}/\sigma_{\min} \sim O(10^4)$ or higher). The smallest singular value of T can be shown to reach machine precision zero if the periodic base flow satisfies the Navier-Stokes equations [106, 81], *i.e.*, when

$$\frac{d\bar{\boldsymbol{q}}}{dt} = \boldsymbol{\mathcal{N}}(\bar{\boldsymbol{q}}), \tag{3.11}$$

where $\boldsymbol{\mathcal{N}}$ is the nonlinear Navier-Stokes operator. Taking a time derivative of both sides,

$$\frac{d\left(\frac{d\bar{\boldsymbol{q}}}{dt}\right)}{dt} = \frac{d\left(\boldsymbol{\mathcal{N}}(\bar{\boldsymbol{q}})\right)}{dt} = \frac{\partial \boldsymbol{\mathcal{N}}}{\partial \bar{\boldsymbol{q}}}\frac{d\bar{\boldsymbol{q}}}{dt} = \boldsymbol{A}_p\frac{d\bar{\boldsymbol{q}}}{dt}. \tag{3.12}$$

Taking a Fourier transform of (3.12) and recalling that $\boldsymbol{T}$ is obtained as shown in (3.3), $\widehat{d\bar{\boldsymbol{q}}/dt}$ is a non-trivial solution of

$$\boldsymbol{T}\boldsymbol{q} = 0, \tag{3.13}$$

residing within the null space of $\boldsymbol{T}$, thereby proving the singularity of $\boldsymbol{T}$. In systems where the base flow approximately satisfies (3.11), $\boldsymbol{T}$ becomes nearly singular. Consequently, irrespective of the specific periodic system under consideration, $\boldsymbol{T}$ might be poorly conditioned, presenting even more challenges when dealing with large systems.

Padovan *et al.* [106] demonstrated that defining the harmonic resolvent operator in a manner that eliminates the phase shift imparted by the Fourier coefficients of $d\bar{\boldsymbol{q}}/dt$ is effective, without adversely affecting the other dominant amplification mechanisms. By constraining $\boldsymbol{T}$ to a subspace $\boldsymbol{U}^{\perp}$ that is orthogonal to the direction of the phase shift $\widehat{d\bar{\boldsymbol{q}}/dt}$, the range of the harmonic resolvent operator is limited to $\boldsymbol{U}^{\perp}$. The key concept here is to eliminate the singular vectors associated with the smallest singular value of $\boldsymbol{T}$ without affecting the other singular values and vectors of $\boldsymbol{T}$. Given that the phase shift resides in the null space of $\boldsymbol{T}$, the desired right singular vector $\boldsymbol{v} = \widehat{\frac{d\bar{\boldsymbol{q}}}{dt}}/\|\widehat{\frac{d\bar{\boldsymbol{q}}}{dt}}\|$ is readily available. We can also solve

$$\boldsymbol{T}^{*}\tilde{\boldsymbol{u}} = \boldsymbol{v} \tag{3.14}$$

and compute the corresponding left singular vector as $\boldsymbol{u} = \tilde{\boldsymbol{u}}/\|\tilde{\boldsymbol{u}}\|$. Upon obtaining $\boldsymbol{v}$ and $\boldsymbol{u}$, the process becomes straightforward: projecting out these modes from the forcing and response terms effectively restricts $\boldsymbol{T}$ to $\boldsymbol{U}^{\perp}$. These steps are elucidated in detail in Padovan *et al.* [106].

To compute the action of the harmonic resolvent operator on a given vector (or matrix), our goal is to determine the vector (or matrix) $\boldsymbol{q}$ that satisfies $\boldsymbol{q} = \boldsymbol{H}\boldsymbol{f}$. In brief, to compute the action of a singular $\boldsymbol{H}$ on a vector $\tilde{\boldsymbol{f}}$, the component along $\boldsymbol{u}$ must be projected out as

$$\tilde{\boldsymbol{f}}_{in} = \tilde{\boldsymbol{f}} - \boldsymbol{u}(\boldsymbol{u}^{*}\tilde{\boldsymbol{f}}), \tag{3.15}$$

and this will provide a response

$$\boldsymbol{T}\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{f}}_{in} \rightarrow \tilde{\boldsymbol{q}} = \boldsymbol{H}\tilde{\boldsymbol{f}}_{in}, \tag{3.16}$$

which is orthogonal to the direction of the phase shift. Nevertheless, to mitigate potential round-off errors, it is advisable to refine the output by projecting out the component along $\boldsymbol{v}$ as

$$\tilde{\boldsymbol{q}}_{out} = \tilde{\boldsymbol{q}} - \boldsymbol{v}(\boldsymbol{v}^*\tilde{\boldsymbol{q}}). \tag{3.17}$$

Similarly, when computing the action of singular $\boldsymbol{H}^*$ on a vector, the initial step involves projecting out the component along $\boldsymbol{v}$. After the response is obtained, to mitigate round-off errors, the component along $\boldsymbol{u}$ is projected out.

3.2 Truncating the harmonic resolvent operator

Computing harmonic resolvent modes is contingent upon $\boldsymbol{T}$ being of finite size. The size and block sparsity pattern of $\boldsymbol{T}$ are determined by two key variables: the number of frequencies within the base flow, N_b, and within the response, N_ω. Here, we elucidate the impact of each variable on the operator.

The rows of $\boldsymbol{T}$ consist of N_b-stencil blocks, dictated by the presence of non-zero elements in $\hat{\boldsymbol{A}}_{p,j}$. In the context of periodic flows, the time-dependent base flow can be expressed using a Fourier series expansion

$$\bar{\boldsymbol{q}}(t) = \sum_{j\omega_f \in \Omega_{\bar{q}}} \hat{\bar{\boldsymbol{q}}}_j e^{\mathrm{i}j\omega_f t}. \tag{3.18}$$

The set $\Omega_{\bar{q}}$ encompasses the relevant frequencies associated with the dominant flow structures. Typically, $\Omega_{\bar{q}}$ is a subset of $j\omega_f$ with $j \in \mathbb{Z}$ and encapsulates the majority of energy within the periodic flows. By retaining the base periodicity and a limited number of higher harmonics, it becomes possible to simplify the base flow representation, resulting in N_b frequencies within $\Omega_{\bar{q}}$ and N_b non-zero elements within each row of $\boldsymbol{T}$.

On the other hand, we aim to resolve perturbations occurring at temporal frequencies $\omega \in \Omega_q$, where Ω_q represents a subset of $j\omega_f$ with $j \in \mathbb{Z}$, although the specific choice of harmonics for Ω_q may vary, as further elucidated in §3.4. Typically, $\Omega_{\bar{q}}$ is a subset of Ω_q, *i.e.*, the perturbation frequency content spans a wider range of harmonics [106]. The number of frequencies N_ω expanding both $\boldsymbol{f}$ and $\boldsymbol{q}$ determines the dimensions of the block matrices, yielding a size of $(N_\omega N) \times (N_\omega N)$ for $\boldsymbol{T}$.

We provide a simple example to visually observe the harmonic resolvent operator. Suppose we use $\Omega_{\bar{q}} = \{-2,-1,0,1,2\}\omega_f$ to expand the LNS operator and $\Omega_q = \{-4,-3,\dots,3,4\}\omega_f$ to expand the perturbations. In this case, the stencil length would be $N_b = 5$, and the size of the block matrix would be $(9N)\times(9N)$. The harmonic resolvent operator is then

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{L}_{-4} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{-3} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 & 0 & 0 & 0 & 0 \\ -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{-2} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 & 0 & 0 & 0 \\ 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{-1} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 & 0 & 0 \\ 0 & 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{0} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 & 0 \\ 0 & 0 & 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{1} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} & 0 \\ 0 & 0 & 0 & 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{2} & -\hat{\boldsymbol{A}}_{-1} & -\hat{\boldsymbol{A}}_{-2} \\ 0 & 0 & 0 & 0 & 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{3} & -\hat{\boldsymbol{A}}_{-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\hat{\boldsymbol{A}}_{2} & -\hat{\boldsymbol{A}}_{1} & \boldsymbol{L}_{4} \end{bmatrix}^{-1}. \tag{3.19}$$

The forcing and response vectors can be represented as

$$\hat{\boldsymbol{f}} = \begin{bmatrix} \hat{\boldsymbol{f}}_{-4} \\ \hat{\boldsymbol{f}}_{-3} \\ \hat{\boldsymbol{f}}_{-2} \\ \hat{\boldsymbol{f}}_{-1} \\ \hat{\boldsymbol{f}}_{0} \\ \hat{\boldsymbol{f}}_{1} \\ \hat{\boldsymbol{f}}_{2} \\ \hat{\boldsymbol{f}}_{3} \\ \hat{\boldsymbol{f}}_{4} \end{bmatrix}, \hat{\boldsymbol{q}} = \begin{bmatrix} \hat{\boldsymbol{q}}_{-4} \\ \hat{\boldsymbol{q}}_{-3} \\ \hat{\boldsymbol{q}}_{-2} \\ \hat{\boldsymbol{q}}_{-1} \\ \hat{\boldsymbol{q}}_{0} \\ \hat{\boldsymbol{q}}_{1} \\ \hat{\boldsymbol{q}}_{2} \\ \hat{\boldsymbol{q}}_{3} \\ \hat{\boldsymbol{q}}_{4} \end{bmatrix}. \tag{3.20}$$

The accuracy of the obtained resolvent system in Fourier space depends on how well the chosen frequencies capture the relevant flow information, and it is affected by the truncation limit imposed on the frequency range. Although it is feasible to reduce the infinite-dimensional harmonic operator to a finite dimension based on perturbation frequencies, the choice of frequency range is consequential. To demonstrate the influence of truncation on response modes, consider a periodic flow with a dominant first harmonic, such that the base flow can be expressed as

$$\bar{\boldsymbol{q}}(t) = \hat{\bar{\boldsymbol{q}}}_0 + \hat{\bar{\boldsymbol{q}}}_1 e^{\mathrm{i}\omega t} + \hat{\bar{\boldsymbol{q}}}_{-1} e^{-\mathrm{i}\omega t}. \tag{3.21}$$

Assume that the LNS equations are subjected to a constant forcing $\boldsymbol{f}(t) = \hat{\boldsymbol{f}}_0$. The LNS operator exhibits frequency content at 0 and $\pm\omega$, indicating that a constant forcing will directly induce a response at these frequencies. However, higher harmonics are also indirectly stimulated, and the triggering of responses at higher harmonics follows a cascade pattern, with norms gradually decaying towards zero, *i.e.*, $\lim_{j\to\infty} \|\hat{\boldsymbol{q}}_j\| = 0$. Hence, the truncation must be set at a sufficiently high level to preserve the response norms.

In practice, by defining Ω_q for both input perturbation and the response, we compute accurate harmonic resolvent modes as long as the norms of higher frequency modes become less relevant within the spectrum. In other words, if we set both forcing and response frequencies sufficiently high and, upon computing harmonic resolvent modes, find that the norms of input and output modes with higher frequency content are relatively small, it indicates convergence, and including higher frequencies has negligible impact on the results. In a case, for instance, where response modes with higher frequency content remain relatively important, extending the frequency range of the output modes is necessary without changing the input frequency content. In our algorithm, the input and output frequency ranges are adjustable as needed.

3.3 Cross-frequency harmonic resolvent analysis

Harmonic resolvent analysis captures the interaction between various frequencies within $\Omega_{\bar{q}}$ and Ω_q, and the typical objective is to identify the optimal forcing mode, potentially spanning various frequencies, that produces the optimal response across a range of frequencies. Alternatively, one can compute the unit-norm forcing at one frequency $\omega_1 \in \Omega_q$ that triggers the most amplified response at another frequency $\omega_2 \in \Omega_q$. This is achieved by defining

$$\boldsymbol{H}_{\omega_2,\omega_1} = \boldsymbol{C}\boldsymbol{H}\boldsymbol{B} \tag{3.22}$$

and computing the modes and gains following the same procedure as before. Here, matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ are constructed to extract the ω_1 and ω_2 frequencies from the forcing and response modes, respectively. This represents a special case of the modified harmonic resolvent operator in (3.9).

3.4 Subharmonic resolvent analysis

The frequency content of the forcing inputs is typically composed of harmonics of the base flow frequency. However, Padovan & Rowley [107] have shown that the harmonic resolvent system may exhibit sensitivity to subharmonic inputs. For instance, in the case where the fundamental base flow frequency is denoted by ω_f, the linearized system may be sensitive to forcing inputs with $\gamma < \omega_f$, such as $\omega_f/2$. More generally, if the frequency content of a periodic LNS operator is limited to the frequency range $\Omega_{\bar{q}}$, it may be desirable to compute input-output modes with frequencies outside of this range, *i.e.*, $\gamma \notin \Omega_{\bar{q}}$. In such cases, the subharmonic resolvent modes become relevant.

To identify unique subharmonic inputs, we can take advantage of the linear nature of the harmonic resolvent operator, which allows interaction with the fundamental frequency ω_f and its harmonics. It can be demonstrated that the interval $(-\omega_f/2, \omega_f/2]$ encompasses all sets of subharmonic frequencies where the input-output modes are unique. As discussed in §3.2, a forcing with a frequency $\gamma \in (-\omega_f/2, \omega_f/2]$ can only trigger responses at frequencies $\gamma + j\omega_f$, where j is an integer. Therefore, the input and output modes exist within the frequency range $\Omega_\gamma = \gamma \oplus \Omega_{\bar{q}}$, where $\oplus$ denotes element-wise addition. Harmonics of γ trigger a unique set of outputs as long as $j\gamma \leq |\omega_f|$. Due to the linear relationship, we can show that all harmonics are isolated sets that need to be studied separately. Note that $-\omega_f/2$ is excluded since $\Omega_{-\omega_f/2} = \Omega_{\omega_f/2}$, resulting in redundancy. For $\gamma \in \Omega_\gamma$, one can derive the subharmonic resolvent system as [107]

$$\hat{\boldsymbol{q}} = (\mathrm{i}\gamma \boldsymbol{I} - \boldsymbol{T})^{-1} \hat{\boldsymbol{f}}. \tag{3.23}$$

To further elucidate the point, consider an example where the subharmonic frequency of interest is $\omega_f/5$ along with its harmonics. Note that the valid harmonics are the ones that fulfill the condition $j\omega_f/5 \in (-\omega_f/2, \omega_f/2]$, which includes the set $\{-2/5, -1/5, 0, 1/5, 2/5\}$. While $\Omega_0 = 0 \oplus \{-N_b/2, \cdots, N_b/2\}$, where N_b is the number of active frequencies within the base flow, is identical to harmonic inputs, the sets $\Omega_{-2/5}$, $\Omega_{-1/5}$, $\Omega_{1/5}$, and $\Omega_{2/5}$ represent distinct input-output modes. In general, for any two distinct subharmonic frequencies γ_1 and γ_2 within the interval $(-\omega_f/2, \omega_f/2]$, the corresponding subharmonic modes in Ω_{γ_1} are decoupled from those in Ω_{γ_2} as there exists no common frequency $\gamma_1 - \gamma_2 \in \Omega_{\bar{q}}$. While our algorithm described in §3.6.2 is described for harmonic resolvent analysis, it can be easily extended to compute subharmonic resolvent modes, as described in §3.6.3.

The approach used to eliminate the singularity of $\boldsymbol{T}$ described in §3.1.2 is not applicable for subharmonic resolvent analysis ($\gamma \neq 0$). Padovan & Rowley [107] formulated an alternative approach in which the domain of $\boldsymbol{T}$ is constrained to a physically meaningful subspace for all choices of γ. In this approach, an oblique projection operator is defined as

$$\boldsymbol{P}(t) = \boldsymbol{I} - \boldsymbol{v}(t)\boldsymbol{u}^*(t), \tag{3.24}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are defined the same as in §3.1.2. The restricted operator can be written as

$$\boldsymbol{T}_p = \boldsymbol{T}\hat{\boldsymbol{P}}. \tag{3.25}$$

The resulting range of $\boldsymbol{T}_p$ is deliberately designed to be invariant under T. This projection facilitates steps within the RSVD-LU algorithm, effectively addressing singularities. For more details on oblique projection properties and the efficient computation of $\boldsymbol{u}$ and $\boldsymbol{v}$, we refer the reader to Padovan & Rowley [107].

3.5 Computing harmonic resolvent modes using RSVD-LU

The RSVD algorithm [53] is a randomized linear algebra technique developed to efficiently identify the singular vectors with the highest gains in a given matrix. In the context of harmonic resolvent analysis, the matrix of interest is the harmonic resolvent operator. This operator, similar to the resolvent operator, relies on the inverse of a sparse matrix, as shown in (3.6), making the computation of modes challenging. Modifying the original RSVD algorithm is necessary to address these challenges, enabling harmonic resolvent analysis. The modifications for resolvent analysis have been extensively documented in the literature by Ribeiro *et al.* [118] and Farghadan *et al.* [44], while the outline for the harmonic resolvent operator can be found in the appendix of Padovan *et al.* [106].

In brief, the computation involves computing the action of $\boldsymbol{H}$ and $\boldsymbol{H}^*$ for sketching the range and image of the harmonic resolvent operator, respectively, and approximating the leading harmonic modes. Computing actions of $\boldsymbol{H}$ and $\boldsymbol{H}^*$, however, requires solving linear systems

$$\begin{aligned} \boldsymbol{T}\hat{\boldsymbol{q}} &= \hat{\boldsymbol{f}}, \\ \boldsymbol{T}^*\hat{\boldsymbol{q}} &= \hat{\boldsymbol{f}}, \end{aligned} \tag{3.26}$$

respectively, in Fourier space. These linear systems are solved via LU-decomposition of $\boldsymbol{T}$, but due to its large size and ill-conditioned nature, computing its LU decomposition can be a formidable computational obstacle, particularly for flows with three inhomogeneous directions. We propose an alternative way in the following section.

3.6 Computing harmonic resolvent modes using time stepping

In this section, we show how time stepping can be used to efficiently compute harmonic resolvent modes. The same concept was first used by Monokrousos *et al.* [98] in the context of the resolvent analysis and further optimized by Martini *et al.* [92] and Farghadan *et al.*

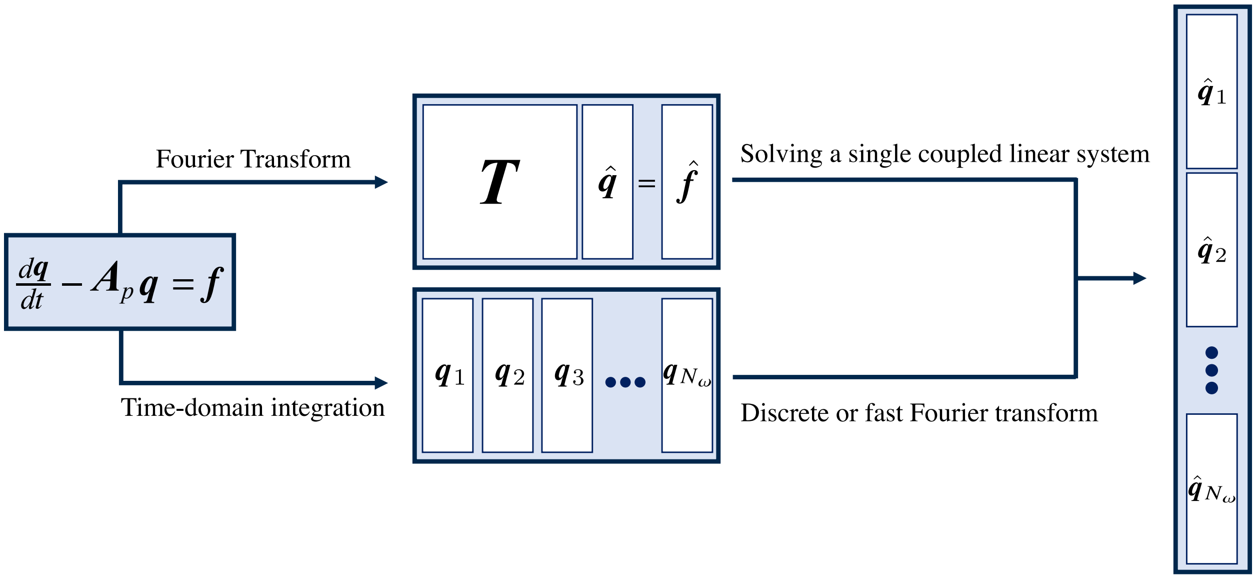

Fourier Transform
$\frac{d\boldsymbol{q}}{dt} - \boldsymbol{A}_p\boldsymbol{q} = \boldsymbol{f}$
Time-domain integration
$\boldsymbol{T}$
$\hat{\boldsymbol{q}} = \hat{\boldsymbol{f}}$
$\boldsymbol{q}_1$ $\boldsymbol{q}_2$ $\boldsymbol{q}_3$ ••• $\boldsymbol{q}_{N_\omega}$
Solving a single coupled linear system
Discrete or fast Fourier transform
$\hat{\boldsymbol{q}}_1$
$\hat{\boldsymbol{q}}_2$
$\hat{\boldsymbol{q}}_{N_\omega}$

Figure 3.1: Flowchart depicting the action of $\boldsymbol{H}$ on a forcing comprised of N_ω frequencies within the RSVD-LU (top route) and the RSVD-Δt (bottom route) algorithms. Both routes produce the same result, but the bottom route is computationally advantageous for large systems.

[44]. Here, we introduce the extension of our algorithm to compute the harmonic resolvent modes and gains.

3.6.1 Computing the action of $\boldsymbol{H}$ using time stepping

Computing the action of $\boldsymbol{H}$ in the Fourier space poses a bottleneck within the RSVD-LU algorithm. To overcome this limitation, we propose to use time-stepping as an effective surrogate approach. Given that $\Omega_{\bar{q}}$ represents the frequency content of the base flow around which $\boldsymbol{T}$ is constructed, we define the time-domain linearized operator as

$$\boldsymbol{A}_p(t) = \sum_{j\omega_f \in \Omega_{\bar{q}}} \hat{\boldsymbol{A}}_{p,j} e^{\mathrm{i}j\omega_f t}. \tag{3.27}$$

The action of $\boldsymbol{H}$ on $\hat{\boldsymbol{f}}$ in Fourier space is expressed in (3.7), where the frequency content of the forcing lies within Ω_q.

The steady-state solution of (3.1) is given by

$$\hat{\boldsymbol{q}}_s = \boldsymbol{H}\hat{\boldsymbol{f}}, \tag{3.28}$$

when subjected to the forcing represented as

$$\boldsymbol{f} = \sum_{j\omega_f \in \Omega_q} \hat{\boldsymbol{f}}_j e^{\mathrm{i}j\omega_f t}. \tag{3.29}$$

The steady-state response can be obtained through time-stepping before taking a Fourier transform. The time-domain forcing is constructed to include frequencies identical to those present in $\hat{\boldsymbol{f}}$. Since the time-domain and frequency-domain frequencies are the same, both approaches yield identical results, *i.e.*, $\hat{\boldsymbol{q}}_s = \hat{\boldsymbol{q}}$. In practice, the time-stepping method is inherently discrete, and the frequencies are retained up to the maximum limit determined by the chosen time step (below the Nyquist frequency [104]). By employing a suitably small time step, we can capture frequencies up to the desired limit.

The equivalence between computing the action of $\boldsymbol{H}$ in both the RSVD-LU and RSVD-Δt algorithms is depicted in figure 3.1. In the upper route, within the RSVD-LU algorithm, the LNS equations are first transformed into Fourier space before solving a coupled linear system in the frequency domain for a given forcing. In the lower route, within the RSVD-Δt algorithm, the LNS equations are initially integrated in the time domain before being transformed into Fourier space. Both routes produce identical outputs so long as numerical artifacts are minimized.

3.6.2 RSVD-Δt for harmonic resolvent analysis

By recognizing the equivalence between time stepping and solving linear systems in the frequency domain, we can effectively address the limitations of the RSVD-LU algorithm, leading to the development of RSVD-Δt. The time-stepping aspect of RSVD-Δt relies on the linearized operator $\boldsymbol{A}_p$ without explicitly constructing $\boldsymbol{H}$ in Fourier space. The algorithmic steps for extending RSVD-Δt to harmonic systems are presented in Algorithm 4.

The algorithm proposed in this study is based on the RSVD-Δt algorithm introduced by Farghadan *et al.* [44], and we will provide a concise overview of the key steps. Initially, a random matrix $\hat{\boldsymbol{\Theta}} \in \mathbb{C}^{NN_\omega \times k}$ is generated (line 2) to compute the sketch of $\boldsymbol{H}$ through time stepping (line 3). The dimensions of the random matrix are determined by the state dimension N, the number of frequencies to resolve N_ω, and the desired number of modes to compute k. `DirectAction` is a function that computes the action of $\boldsymbol{H}$ onto a given forcing using time-stepping and transforms the steady-state responses to Fourier space. As outlined in §3.6.1, `DirectAction` solves (3.1) with zero initial condition over a sufficiently long time interval, causing the initial transient response to dissipate. T_t represents the duration of this interval before the transient diminishes. The forcing term in (3.1) is constructed via inverse

Algorithm 4 RSVD-Δt for harmonic resolvent analysis

```
1:  Inputs: $\boldsymbol{A}_p, k, q, \Omega_q, \Omega_{\bar{q}}$,TSS, $dt, T_t$
2:  $\hat{\boldsymbol{\Theta}} \leftarrow \text{randn}(NN_\omega, k)$                     ▷ Create random test matrices
3:  $\hat{\boldsymbol{Y}} \leftarrow$ DirectAction($\boldsymbol{A}_p, \hat{\boldsymbol{\Theta}}$,TSS, $dt, T_t$)              ▷ Sample the range of $\boldsymbol{H}$
4:  if $q > 0$ then                                            ▷ Optional power iteration
5:      $\hat{\boldsymbol{Y}} \leftarrow$ PI($\boldsymbol{A}_p, \hat{\boldsymbol{Y}}, q$,TSS, $dt, T_t$)        ▷ Power iteration with time-stepping
6:  $\hat{\boldsymbol{Q}}_\Omega \leftarrow \text{qr}(\hat{\boldsymbol{Y}}_\Omega)$                  ▷ Build the orthonormal subspace $\hat{\boldsymbol{Q}}_\Omega$
7:  $\hat{\boldsymbol{S}} \leftarrow$ AdjointAction($\boldsymbol{A}_p^*, \hat{\boldsymbol{Q}}$,TSS, $dt, T_t$)          ▷ Sample the image of $\boldsymbol{H}$
8:  $(\tilde{\boldsymbol{U}}_H, \boldsymbol{\Sigma}_H, \boldsymbol{V}_H) \leftarrow \text{svd}(\hat{\boldsymbol{S}}_H)$                        ▷ Obtain $\boldsymbol{\Sigma}_H, \boldsymbol{V}_H$
9:  $\boldsymbol{U}_H \leftarrow \hat{\boldsymbol{Q}}_\Omega \tilde{\boldsymbol{U}}_H$                                    ▷ Recover $\boldsymbol{U}_H$
10: Outputs: $\boldsymbol{U}_H, \boldsymbol{\Sigma}_H, \boldsymbol{V}_H$
```

Algorithm 1. Inputs: linearized operator $\boldsymbol{A}_p$, number of modes k, number of power iterations q, frequency range of the perturbations and LNS operator Ω_q and $\Omega_{\bar{q}}$, respectively, time-stepping scheme abbreviated as TSS (*e.g.*, backward Euler), time step dt, and the transient length T_t. Outputs: k response modes $\boldsymbol{U}_H$, k forcing modes $\boldsymbol{V}_H$ and the corresponding gains $\boldsymbol{\Sigma}_H$. Here, $k, q, \Omega_q, \Omega_{\bar{q}}$ are common parameters with RSVD-LU. $(\cdot)_\Omega$ indicates all frequencies are merged into a single column. `DirectAction` and `AdjointAction` are functions that solve the direct and adjoint LNS equations, respectively, with a predefined forcing. `PI` is a function that performs the power iteration.

Fourier transform of $\hat{\boldsymbol{\Theta}}$ and the steady-state response undergoes a Fourier transform to obtain $\hat{\boldsymbol{Y}}$. Next, an optional power iteration (`PI`) is performed (lines 4 and 5) via q successive applications of `DirectAction` and `AdjointAction`, enhancing the accuracy of harmonic resolvent modes. QR decomposition is performed to obtain $\hat{\boldsymbol{Q}}_\Omega$ (line 6) before constructing $\boldsymbol{S}$ via time stepping (line 7). `AdjointAction` is the function that computes the action of $\boldsymbol{H}^*$ onto a given forcing using time-stepping and transforms the steady-state responses to Fourier space. This function is similar to `DirectAction`, but it evolves the adjoint system. The forcing term in `AdjointAction` is constructed via the inverse Fourier transform of $\hat{\boldsymbol{Q}}$. An SVD (line 8) is conducted to obtain the optimal forcing modes $\boldsymbol{V}_H \in \mathbb{C}^{NN_\omega \times k}$ and gains $\boldsymbol{\Sigma}_H \in \mathbb{R}^{k\times k}$ of $\boldsymbol{H}$. Lastly, the optimal response modes $\boldsymbol{U}_H \in \mathbb{C}^{NN_\omega \times k}$ are recovered in line 9.

Compared to the RSVD-Δt algorithm for resolvent analysis [44], two notable differences are present. In the extension of RSVD-Δt, the response modes at various frequencies are merged prior to both the QR decomposition and the SVD steps. This deviation from the original algorithm is motivated by the nature of the problem that requires interactions between frequencies, unlike in resolvent analysis where the QR decomposition and SVD are performed separately for each individual frequency of interest. The second difference is the generation of LNS operators over time, whereas in resolvent analysis, the LNS operator remains constant throughout the integration.

If $\boldsymbol{T}$ approaches machine precision singularity, it is necessary to project out the forcing along the $\boldsymbol{u}$ direction before the `DirectAction` and along $\boldsymbol{v}$ before the `AdjointAction`, as described in §3.1.2. Moreover, to mitigate numerical artifacts, it is advisable to project out the response along $\boldsymbol{v}$ and $\boldsymbol{u}$ after the `DirectAction` and `AdjointAction`, respectively.

3.6.3 RSVD-Δt for the subharmonic resolvent operator

In §3.4, we provided an overview of subharmonic resolvent analysis. In this appendix, we briefly outline the application of RSVD-Δt in computing subharmonic resolvent modes and gains. Consider the frequency of interest as $\gamma \in \Omega_\gamma$. This set specifies the perturbation frequency, while the base flow frequency content is confined to $\Omega_{\bar{q}}$.

To compute the actions of $\boldsymbol{H}$ and $\boldsymbol{H}^*$ using time-stepping, we must adhere to both the base flow frequency, enforcing a duration of $T = 2\pi/\omega_f$, and the perturbation frequency, enforcing a duration of $T_p = 2\pi/\gamma$, in order to obtain N_ω steady-state solutions to (3.1). Thus, we need to integrate for T_{sub} such that $T_{sub}/T \in \mathbb{N}$ and $T_{sub}/T_p \in \mathbb{N}$, during which N_ω equidistant snapshots are saved. For instance, if we consider $\gamma = \omega_f/5$, requiring $T_p = 2\pi/(\omega_f/5) = 5T$, integrating for $T_{sub} = 5T$ is sufficient to obtain the steady-state solutions. All the other steps remain the same as introduced in Algorithm 3.

3.7 Computational complexity

A brief comparison is conducted between memory requirements and CPU cost of the RSVD-LU and RSVD-Δt algorithms. We also propose strategies to minimize the memory consumption. Throughout, we assume that the linearized operator $\boldsymbol{A}_p$ is sparse, *i.e.*, that its number of non-zero entries scales as $\texttt{nnz}(\boldsymbol{A}_p) \sim O(N)$. Sparse operators are obtained when using sparse discretization schemes such as finite differences, finite volume, or finite elements. For more detailed information on the CPU and memory scaling of the LU decomposition of the resolvent operator, we refer readers to Farghadan *et al.* [44]. To offer empirical insights and confirmation of the theoretical scalings discussed below, we present a two-dimensional test case in §3.9.

3.7.1 Memory benefits of RSVD-Δt

In the case of resolvent analysis, the memory scaling of the LU decomposition is empirically $O(N^{1.2})$ and $O(N^{1.6})$ for two- and three-dimensional systems, respectively [159, 44]. The memory scaling of LU decomposition is expected to be worse for the harmonic resolvent operator, as the lower- and upper-triangular matrices are denser. When $\boldsymbol{T}$ is singular or nearly singular, the computational task becomes even more challenging, as discussed in Padovan *et al.* [106]. An alternative approach for solving (3.26) involves leveraging iterative solvers, such as Krylov subspace methods [119, 169]. This approach eliminates the need for computing the LU decomposition and becomes more memory-efficient, especially for large systems. However, finding a suitable preconditioner remains a challenging task.

On the other hand, the memory usage of RSVD-Δt for harmonic resolvent analysis is contingent on the sizes of various matrices, specifically the sparse LNS operators $\hat{\boldsymbol{A}}_p \in \mathbb{C}^{N\times N\times N_b}$, and the dense forcing matrix $\hat{\boldsymbol{F}} \in \mathbb{C}^{NN_\omega \times k}$ and response matrix $\hat{\boldsymbol{Q}} \in \mathbb{C}^{NN_\omega\times k}$, all of which are stored in Fourier space. The scaling for resolvent analysis is $O(NN_\omega)$, and the sole extra space is for the storage of LNS operators.

One approach to generating LNS operators for all time points involves retaining N_b coefficients in Fourier space and subsequently constructing $\boldsymbol{A}_\Omega = \{\boldsymbol{A}_{p,1}, \boldsymbol{A}_{p,2}, \boldsymbol{A}_{p,3}, \dots, \boldsymbol{A}_{p,N_t}\}$ once for all time steps within a period. However, this approach can be memory-intensive for large systems, especially when $N_t \sim O(10^3 - 10^5)$. An alternative strategy to reduce memory consumption is to create LNS operators on the fly, as shown in figure 3.2. As a result, no time-domain LNS operator is permanently stored in memory. Each LNS operator is created on the fly using the discrete Fourier transform (DFT) of N_b Fourier coefficients,

$$\boldsymbol{A}_p(t) = \sum_{j=-N_b/2}^{N_b/2} \hat{\boldsymbol{A}}_{p,j} e^{\mathrm{i}j\omega_f t}. \tag{3.30}$$

Since $\boldsymbol{A}_p$ is very sparse, $\texttt{nnz}(\boldsymbol{A}_p) \sim O(N)$, resulting in an overall memory consumption of $O(NN_b)$. The methods employed for generating LNS operators, both in the streaming and memory-intensive approaches, mirror those used for creating forcing terms in resolvent analysis, as elaborated in Farghadan *et al.* [44]. Additionally, the CPU cost of this procedure scales as $O(\texttt{nnz}(\boldsymbol{A}_p)N_b)$ or $O(NN_b)$.

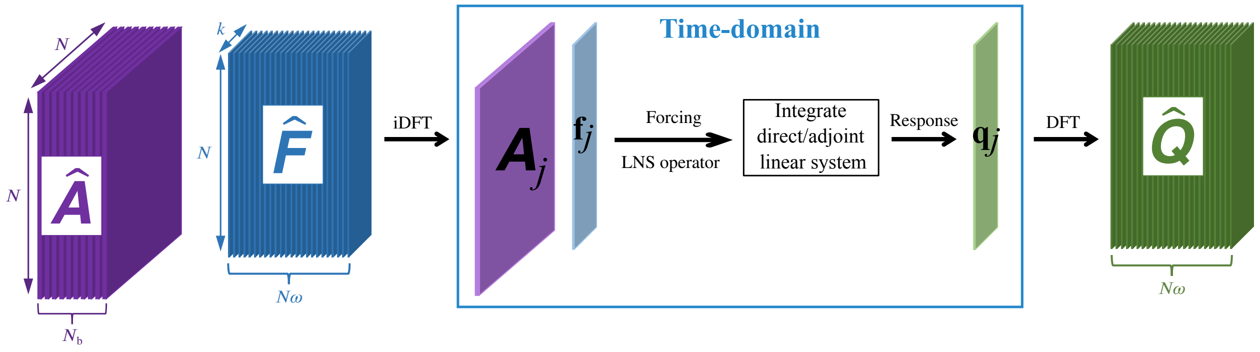


Figure 3.2: Schematic of the action of $\boldsymbol{H}$ with streaming DFT/iDFT to transform between the Fourier and time domains.

Streaming DFT is indeed effective in reducing memory consumption; nevertheless, additional memory savings can be specifically achieved for real-valued LNS operators. In many cases, $\boldsymbol{A}_p$ is real-valued, *i.e.*, confined to $\mathbb{R}^{N\times N}$, yielding $\hat{\boldsymbol{A}}_{p,j} = \bar{\hat{\boldsymbol{A}}}_{p,-j}$, where $\bar{(\cdot)}$ denotes the complex conjugate. Hence, $\hat{\boldsymbol{A}}_{p,j}$ with $j \geq 0$ suffices to construct $\boldsymbol{A}_p$ as

$$\boldsymbol{A}_p(t) = \hat{\boldsymbol{A}}_{p,0} + 2\mathcal{R}e\left(\sum_{j=1}^{N_b/2} \hat{\boldsymbol{A}}_{p,j} e^{\mathrm{i}j\omega_f t}\right), \tag{3.31}$$

resulting in halving the number of Fourier coefficients of the LNS operator to $\lfloor N_b/2 \rfloor + 1$. Here, $\mathcal{R}e$ represents the real part.

Another opportunity to reduce memory usage for real-valued LNS matrices arises from the symmetry between positive and negative frequencies. Rewriting (3.3) for $-k\omega_f$ yields

$$\mathrm{i}(-k\omega_f)\hat{\boldsymbol{q}}_{-k} - \sum_{j=-\infty}^{\infty} \hat{\boldsymbol{A}}_{-k+j}\hat{\boldsymbol{q}}_{-j} = \hat{\boldsymbol{f}}_{-k}, \tag{3.32}$$

and the complex conjugate version of (3.3) becomes

$$-\mathrm{i}k\omega_f \bar{\hat{\boldsymbol{q}}}_k - \sum_{j=-\infty}^{\infty} \bar{\hat{\boldsymbol{A}}}_{k-j}\bar{\hat{\boldsymbol{q}}}_j = \bar{\hat{\boldsymbol{f}}}_k. \tag{3.33}$$

Both equations (3.32) and (3.33) yield the same output for a given $\bar{\hat{\boldsymbol{f}}}_k = \hat{\boldsymbol{f}}_{-k}$ since $\bar{\hat{\boldsymbol{A}}}_{k-j} = \hat{\boldsymbol{A}}_{-k+j}$, inducing the harmonic resolvent response and forcing modes containing $\boldsymbol{U}_{H,j} = \bar{\boldsymbol{U}}_{H,-j}$ and $\boldsymbol{V}_{H,j} = \bar{\boldsymbol{V}}_{H,-j}$ for $j \neq 0$, respectively. Hence, we only keep $\lfloor N_\omega/2 \rfloor + 1$ Fourier coefficients of the forcing and response modes, reducing memory consumption by half when storing these dense matrices.

3.7.2 CPU benefits of RSVD-Δt

The computational cost of computing harmonic resolvent modes using RSVD-LU is predominantly determined by the LU decomposition of $\boldsymbol{T}$. In the case of resolvent analysis, the CPU scaling of the LU decomposition is $O(N^{1.5})$ and $O(N^2)$ for two- and three-dimensional systems, respectively [159, 44]. When, in general, block matrix $\boldsymbol{T}$ contains $\hat{\boldsymbol{A}}_{p,i}$ terms, and yields a denser matrix than block diagonal with a complex sparsity pattern, the scaling is expected to be $O((N_\omega N)^{1.5})$ or $O((N_\omega N)^2)$ or worse. Alternatively, iterative solvers may outperform LU decomposition in terms of CPU scalability, contingent on improving the condition number of $\boldsymbol{T}$ via applying an appropriate preconditioner [119, 169], which itself may or may not be inexpensive to obtain.

The cost of using RSVD-Δt is directly tied to the cost of computing the actions of $\boldsymbol{H}$ and $\boldsymbol{H}^*$. In resolvent analysis, the total CPU cost is proportional to the state dimension

N. This cost can be broken down into three main components: (i) time-stepping, *e.g.*, Adams-Bashforth methods, which scales as $O(N)$ when $\boldsymbol{A}$ is sparse, (ii) creating forcing from Fourier space to the time domain, which scales as $O(NN_tN_\omega)$, (iii) taking the response from the time domain to Fourier space, which scales as $O(NN_\omega^2)$. Among these components, the time-stepping part is the most significant contributor to the overall cost. The CPU cost of harmonic resolvent analysis can be assessed similarly to resolvent analysis, with one exception. For periodic flows, the LNS operator varies over time, whereas it remains constant for statistically stationary flows. We showed in §3.7.1 that the cost of creating LNS operators over time scales as $O(NN_b)$ for sparse matrices.

In summary, the CPU and memory costs of RSVD-Δt exhibit linear scaling with respect to the state dimension N, regardless of whether the base flow is steady or periodic. Moreover, the dominant terms in the CPU cost are independent of the number of retained frequencies N_ω. Both of these properties make the RSVD-Δt algorithm considerably more scalable than RSVD-LU for harmonic resolvent analysis.

3.8 Minimizing the CPU cost of RSVD-Δt

Similar to time stepping in the context of resolvent analysis, periodic systems may also experience a slow decay of transient response, which is undesirable, since we require the steady-state response in isolation to compute the action of the harmonic resolvent operator using time stepping. This phenomenon might be linked to instances when $\boldsymbol{T}$ is near singularity. Regardless, in cases where the LNS equations are absolutely stable but exhibit slowly decaying modes, we propose a strategy that can effectively reduce the CPU cost of time stepping.

3.8.1 Stability analysis: Floquet theorem and transient response

The steady-state response of the LNS equations, subject to forcing as described in equation (3.1), is of interest for our analysis. In the case of a linear time-invariant system where $\boldsymbol{A}$ is independent of time, the decay rate is controlled by the least-damped eigenvalue of $\boldsymbol{A}$. This decay rate can be estimated by integrating a homogeneous system over a sufficiently long period. However, for a linear time-periodic system, the decay rate is determined by the least-damped Floquet exponent.

Assuming $\boldsymbol{\Phi}(t)$ is the fundamental solution of (3.1), the monodromy matrix is constructed as

$$\boldsymbol{M} = \boldsymbol{\Phi}(0)^{-1}\boldsymbol{\Phi}(T), \tag{3.34}$$

where T is the period of the flow. When $\boldsymbol{\Phi}(0) = \boldsymbol{I}, \boldsymbol{\Phi}(t)$ is referred to as the principal fundamental matrix, and the monodromy matrix simplifies to $\boldsymbol{\Phi}(T)$. In other words, we evaluate the principal fundamental matrix at the end of the first period. The eigenvalues of $\boldsymbol{M}$, known as Floquet multipliers μ_j, allow us to determine the Floquet exponents $\lambda_j = \log(\mu_j)/T$. The least-damped Floquet exponent has the smallest real part and it determines the decay rate of the transient response. If any mode is located in the unstable half-plane, *i.e.*, $\mathcal{R}e(\lambda_j) > 0$, the system is globally unstable.

It is worth noting that the transient length remains independent of the steady-state period. This characteristic is particularly important as it ensures that the length of integration does not rely on T, thereby avoiding potential deterioration in the performance of RSVD-Δt for flows with low periodicity. In practice, both finding the principal fundamental matrix and determining the Floquet exponents can be computationally expensive, especially for large-scale systems. As with autonomous cases, it is possible to run a homogeneous simulation to analyze the long-term behavior of the transient response for periodic flows. This equivalence has been illustrated on a test case in §3.9.

If $\boldsymbol{T}$ is singular but all other modes are stable, then the Floquet exponent with the smallest real part will be zero with $\widehat{d\bar{\boldsymbol{q}}/dt}$ as the corresponding Floquet mode. This is a well-known fact from Floquet theory [167] and can also be seen from (3.12), where the phase shift direction is a non-trivial solution of the homogenous system. Leclercq & Sipp [81] have also shown that $d\bar{\boldsymbol{q}}/dt$ is associated with a zero Floquet exponent under the assumption that the base flow satisfies the Navier-Stokes equations as in (3.11).

3.8.2 Efficient transient removal

Our strategy leverages the periodic nature of the steady-state part and the linear evolution of the transient part of the solution to directly compute and eliminate the undesired transient portion. For a given pair of solutions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ at times t_1 and $t_2 = t_1 + T$, respectively, they can be expressed as a sum of their steady-state and transient components as

$$\begin{aligned} \boldsymbol{q}_1 &= \boldsymbol{q}_{s,1} + \boldsymbol{q}_{t,1}, \\ \boldsymbol{q}_2 &= \boldsymbol{q}_{s,2} + \boldsymbol{q}_{t,2}, \end{aligned} \tag{3.35}$$

where $\boldsymbol{q}_{t,1}$ and $\boldsymbol{q}_{t,2}$ represent the transient part which decays to zero as $t \to \infty$ and $\boldsymbol{q}_{s,1}$ and $\boldsymbol{q}_{s,2}$ denote the steady-state part which evolves periodically. The time distance between two snapshots is one period, hence

$$\boldsymbol{q}_{s,2} = \boldsymbol{q}_{s,1}. \tag{3.36}$$

Also, the evolution of the transient part can be expressed as

$$\boldsymbol{q}_{t,2} = \boldsymbol{\Phi}(T)\boldsymbol{q}_{t,1}, \tag{3.37}$$

where $\boldsymbol{\Phi}$ is the principal fundamental matrix of (3.1). Simplifying (3.35), (3.36), and (3.37) leads to

$$(\boldsymbol{\Phi}(T) - \boldsymbol{I})\boldsymbol{q}_{t,1} = \boldsymbol{b}, \tag{3.38}$$

where $\boldsymbol{b} = \boldsymbol{q}_2 - \boldsymbol{q}_1$ is known.

By obtaining $\boldsymbol{q}_{t,1}$ from solving (3.38), the steady-state solution can be recovered as $\boldsymbol{q}_{s,1} = \boldsymbol{q}_1 - \boldsymbol{q}_{t,1}$. The central challenge in removing the transient is the computational cost associated with solving the linear system in (3.38). Instead, we employ Petrov-Galerkin (or Galerkin) projection to obtain an approximate solution to (3.38) in a more cost-effective manner.

A low-dimensional representation of the transient part can be expressed as

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}\boldsymbol{\beta}_1, \tag{3.39}$$

where $\boldsymbol{\phi} \in \mathbb{C}^{N\times r}$ is an orthonormal low-dimensional trial basis with $r \ll N$, and $\boldsymbol{\beta}_1 \in \mathbb{C}^r$ represents the coefficients describing the transient response in this basis. Substituting (3.39) into (3.38), the linear system

$$(\boldsymbol{\Phi}(T) - \boldsymbol{I})\boldsymbol{\phi}\boldsymbol{\beta}_1 = \boldsymbol{b} \tag{3.40}$$

is overdetermined. We use Petrov-Galerkin projection with a low-dimensional test basis $\boldsymbol{\psi} \in \mathbb{C}^{N\times r}$ to close (3.40), yielding

$$\tilde{\boldsymbol{M}}\boldsymbol{\beta}_1 = \boldsymbol{\psi}^*\boldsymbol{b}, \tag{3.41}$$

where

$$\tilde{\boldsymbol{M}} = \boldsymbol{\psi}^*(\boldsymbol{\Phi}(T) - \boldsymbol{I})\boldsymbol{\phi} \tag{3.42}$$

is the map between coefficients. Solving (3.41) for $\boldsymbol{\beta}_1$ is inexpensive due to its reduced dimension, and from (3.39) the transient estimate is obtained as

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}(\boldsymbol{\psi}^*\boldsymbol{\phi} - \tilde{\boldsymbol{M}})^{-1}\boldsymbol{\psi}^*\boldsymbol{b}. \tag{3.43}$$

The reduced operator $\tilde{\boldsymbol{M}}$ is obtained by integrating the columns of $\boldsymbol{\phi}$ for one period, giving $\boldsymbol{\Phi}(T)\boldsymbol{\phi}$, and then projecting $(\boldsymbol{\Phi}(T) - \boldsymbol{I})\boldsymbol{\phi}$ against the columns of $\boldsymbol{\psi}$. This integration happens once and its CPU cost is equivalent to integrating the LNS equations for r periods. The analogous process occurs for the adjoint equations.

How should the trial and test bases be selected? We have found a trial basis spanning

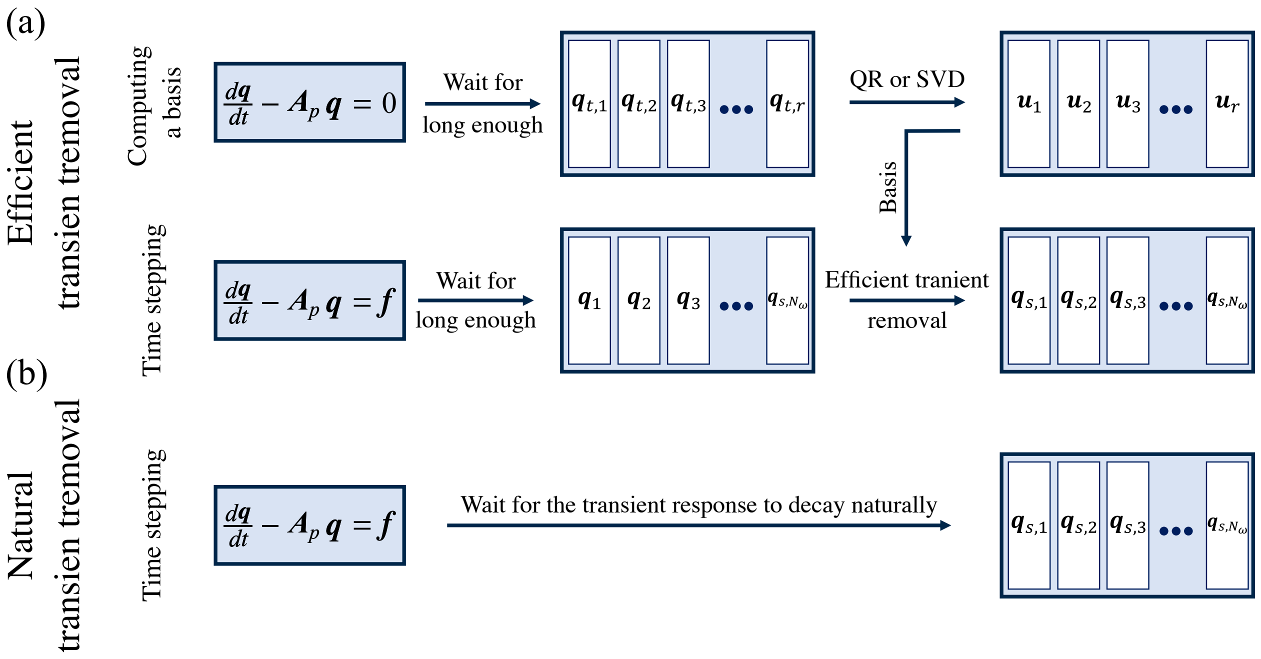

(a)
Efficient transien tremoval
Computing a basis
$\frac{dq}{dt} - A_p q = 0$
Wait for long enough
$q_{t,1}$ $q_{t,2}$ $q_{t,3}$ ••• $q_{t,r}$
QR or SVD
u_1 u_2 u_3 ••• u_r
Basis
Time stepping
$\frac{dq}{dt} - A_p q = f$
Wait for long enough
q_1 q_2 q_3 ••• q_{s,N_ω}
Efficient tranient removal
$q_{s,1}$ $q_{s,2}$ $q_{s,3}$ ••• q_{s,N_ω}
(b)
Natural transien tremoval
Time stepping
$\frac{dq}{dt} - A_p q = f$
Wait for the transient response to decay naturally
$q_{s,1}$ $q_{s,2}$ $q_{s,3}$ ••• q_{s,N_ω}

Figure 3.3: Flowchart depicting the action of $\boldsymbol{H}$ using time stepping (a) with efficient transient removal strategy and (b) with natural transient decay.

the least-damped modes to be effective. Rather than determining these modes through an exhaustive Floquet analysis, we employ a homogenous simulation as illustrated in the top row of the flowchart in figure 3.3. By initiating a sufficiently long simulation from a normalized random initial condition, the system asymptotically converges to the least-damped modes. The longer the integration length, the lower-dimensional the space becomes. As demonstrated in a test case in §3.9, the dimension of the trial bases can, in some cases, be as small as a single column. Our test cases employ Galerkin projection with identical trial and test bases.

The obtained trial basis effectively captures the transient response at a specific phase θ. Utilizing $\boldsymbol{\phi}$ and performing Galerkin projection, we obtain $\tilde{\boldsymbol{M}}$. The time stepping is then carried out for a sufficiently long duration to allow the initial transients to vanish before applying (3.41) to determine the transient at phase θ as shown in the middle row of the flowchart in figure 3.3. Once the steady-state response at the same phase is updated as $\boldsymbol{q}_{s,1} = \boldsymbol{q}_1 - \boldsymbol{q}_{t,1}$, it is used as a new initial condition, which is synchronized with the forcing and will not excite a transient response (within the span of the bases used to construct $\tilde{\boldsymbol{M}}$). Then, integration for one period is sufficient, and N_ω steady-state snapshots are collected as desired. The resulting steady-state snapshots are identical to those obtained by a prolonged wait in the case of natural decay, as illustrated in the bottom row of the flowchart in figure 3.3. This procedure is also applicable to the adjoint LNS equations. Given that the adjoint equations differ from the LNS equations, a new basis needs to be constructed for them. Implementing this strategy can reduce the integration length by a factor of 10 or more, depending on the desired accuracy.

3.9 Test cases

We assess the effectiveness of RSVD-Δt in periodic systems using two test cases. First, we use a modified Ginzburg-Landau equation to validate the RSVD-Δt algorithm and to illustrate the transient removal strategy. Second, we consider a periodic flow around an airfoil, similar to a test case from Padovan *et al.* [106], to evaluate the accuracy and cost-savings of the RSVD-Δt algorithm compared to RSVD-LU.

3.9.1 Periodically varying complex Ginzburg-Landau equation

The one-dimensional complex Ginzburg-Landau equations is a widely used model for understanding and controlling the non-modal growth of instabilities in transitional and turbulent shear flows [23, 65, 7, 22, 20]. Here, a modified system is used as an inexpensive test case to validate our algorithm. The original complex Ginzburg-Landau system follows the form of (2.3) with

$$\begin{aligned}\mathcal{A} &= -\nu\frac{\partial}{\partial x} + \gamma\frac{\partial^2}{\partial x^2} + \mu(x),\\ \mu(x) &= (\mu_0 - c_\mu^2) + \frac{\mu_2}{2}x^2,\\ \boldsymbol{B} &= \boldsymbol{C} = \boldsymbol{I}.\end{aligned} \tag{3.44}$$

Following Bagheri *et al.* [7], we set $c_\mu = 0.2, \mu_2 = -0.01, \gamma = 1 - \mathrm{i}$, and $\nu = 2 + 0.2\mathrm{i}$. This system is globally stable when $\mu_0 < \mu_{0,cr} \approx 0.3977$ [7]. By substituting $\mu_0(t) = \bar{\mu}_0 + \mu_p sin(\omega_f t - \pi/2)$ in place of μ_0, we transform the Ginzburg-Landau equations to a periodically varying system with fundamental frequency ω_f, following the form of (3.1). We set $\mu_0 = 0.395$, $\mu_p = 0.1$, and $\omega_f = 0.1$; the mean value of $\mu(t)$ is equal to μ_0, whose value is close to the critical value $\mu_{0,cr}$ to promote significant growth. The periodic linearized operators $\boldsymbol{A}_{p,j}$ are constructed using a central finite-difference method for $x \in [-50, 50]$ with $N = 1000$ grid points, and we set $\boldsymbol{W} = \boldsymbol{I}$ on account of the uniform grid.

Nine operators are built within the interval $[0, T)$, with $T = 2\pi/\omega_0 \approx 62.83$, such that $\boldsymbol{A}_p(t) = \boldsymbol{A}_p(t + T)$, before conducting a DFT to obtain $\hat{\boldsymbol{A}}_p$. Figure 3.4(a) depicts the Frobenius norm of $\hat{\boldsymbol{A}}_p$ up to the fourth harmonic, normalized relative to the maximum norm. The spectrum reveals that only the mean and the first harmonic are active, as expected, since $\mu_0(t)$ oscillates at a single frequency. Accordingly, $\boldsymbol{A}_p$ can be reconstructed at any given time using $\hat{\boldsymbol{A}}_{p,0}$, and $\hat{\boldsymbol{A}}_{p,\pm 1}$, and the harmonic resolvent operator is constructed around $\Omega_{\bar{q}} = \{-1, 0, 1\}\omega_f$, *i.e.*, $N_b = 3$. Notably, the first harmonic's norm is several orders of magnitude smaller than that of the base flow. Nonetheless, it should not be disregarded, as we will elucidate its significance in subsequent discussions.

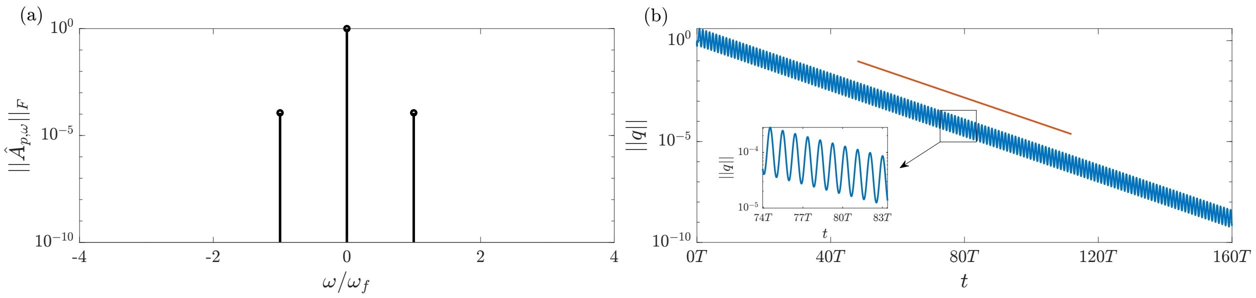


Figure 3.4: Ginzburg-Landau test case: (a) the normalized Frobenius norm of $\hat{\boldsymbol{A}}_{p,\omega}$ up to the fourth harmonic. (b) The norm of the transient response over time (blue) along with the expected decay rate from Floquet analysis (red).

An essential initial step involves ensuring the stability of $\boldsymbol{A}_p$. Typically, this is achieved by integrating the homogeneous system, *i.e.*, integrating (3.1) with zero forcing, starting from a normalized random initial condition. However, the compact size of this system allows us to validate the time-stepping approach for computing the least-damped decaying mode against a Floquet stability analysis. The least-damped Floquet exponent, computed as $\lambda_l = -0.0021 - 0.0477\mathrm{i}$, is expected to govern the decay rate of the transient response. Figure 3.4(b) displays the norm of snapshots of the homogeneous system response over time, along with a reference line representing $e^{\lambda_l t}$. The natural decay rate aligns with the reference line, as expected, ensuring that the transient run will ultimately converge to the least-damped Floquet mode. Additionally, the system is not singular, as the transient dynamics never reach a naturally stable state, and none of the Floquet exponents have a zero real value.

Before proceeding to the computation of harmonic resolvent modes and gains using RSVD-Δt and RSVD-LU, the transient removal strategy is demonstrated, and computing the actions of $\boldsymbol{H}$ and $\boldsymbol{H}^*$ are validated for a given random forcing containing $\Omega_q \in \{-10, -9, ..., 9, 10\}\omega_f$, *i.e.*, $\omega \in [-1, 1]$ with $\Delta\omega = 0.1$. The responses have been computed up to the 20th harmonic, *i.e.*, $\omega \in [-2, 2]$, using both time stepping and directly in Fourier space as outlined in (3.7). We used a 4th order Runge–Kutta (RK4) integration scheme and a time step of $dt = 0.001$ and $T_t = 15000$ for this purpose.

Figure 3.5(a) depicts the spectrum of response norms. In order to highlight the substantial difference in output generated by the harmonic resolvent operator around the base flow ($\Omega_{\bar{q}} = \{0\}\omega_f, N_b = 1$), the norm of the responses for the same input forcing is also presented.

Figure 3.5(b) displays the relative errors. The harmonic resolvent with $N_b = 1$ produces inaccurate results across the entire spectrum and has zero response for $|\omega| > 1$ since the modes at different frequencies are entirely decoupled. Therefore, neglecting $\hat{\boldsymbol{A}}_{p,\pm1}$ is impractical, as mentioned earlier, and induces significant alterations in the harmonic resolvent modes and gains. On the other hand, the time-stepping output exhibits almost machine-precision accuracy within $|\omega| \leq 1$, and its accuracy diminishes as response norms decrease within $|\omega| > 1$. The validation of $\boldsymbol{H}^*$ follows a similar procedure but is omitted here for brevity.

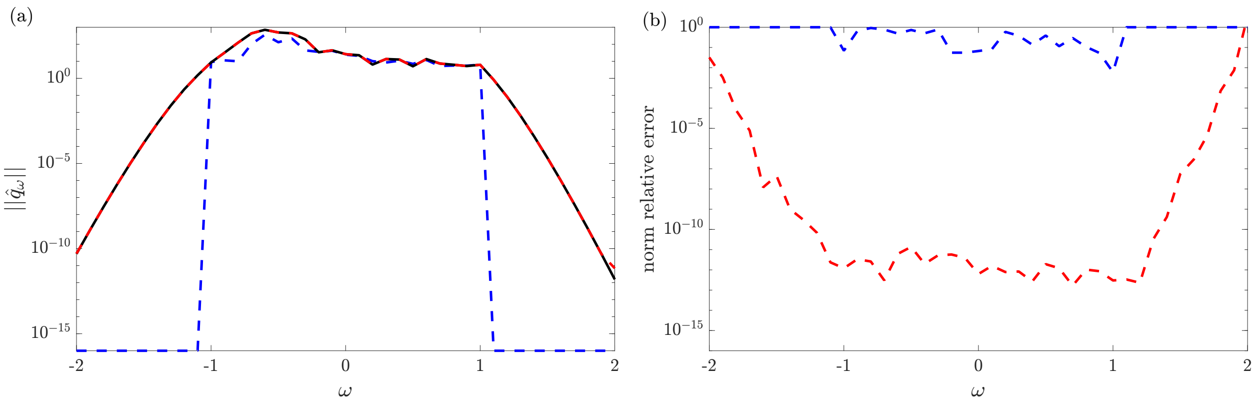


Figure 3.5: Ginzburg-Landau test case: (a) the norms of the Fourier space response for $N_b = 3$ (black) and $N_b = 1$ (blue), along with the norm of the steady-state response obtained via time-stepping (red). (b) The relative errors of the norms, with the norms from Fourier space with $N_b = 3$ (black line in (a)) serving as the reference.

The transient norm decay rate, as shown in figure 3.4(b), follows $e^{-0.0021t}$. This indicates that approximately 40 periods are necessary for the transient norm to decay to 1%. However, we can significantly reduce this duration by employing our transient removal strategy. To begin, we precompute the trial basis by conducting a sufficiently long homogeneous simulation, and we find that a single mode suffices. Next, we integrate the orthogonalized basis for one period to obtain $\tilde{M}$ and complete the preprocessing step.

Employing the same forcing as the validation case, we obtain the responses in time at the identical phase (initiated at $t = 0$ or $\theta = 0$ with a time interval of $\Delta t = T$), and compare them with the solutions after the removal of transients. Figure 3.6(a) presents the norm of the responses, which illustrates the effectiveness of our strategy commencing from the second period. This plot suggests that after two periods using our transient removal strategy with Galerkin projection, we obtain an accurate steady-state snapshot which can be used as the synchronized initial condition for the forced equations. Figure 3.6(b) demonstrates that the snapshots ultimately align perfectly with the updated snapshot. Hence, using our strategy, a total of three periods are required – two periods before transient removal and one period after the initial condition is obtained – to acquire all N_ω steady-state snapshots, a 10-fold speed-up compared to the natural decay. A similar observation was made regarding the adjoint equations. In brief, our strategy can significantly reduce computational costs, potentially by one or more orders of magnitude, depending on the desired level of accuracy.

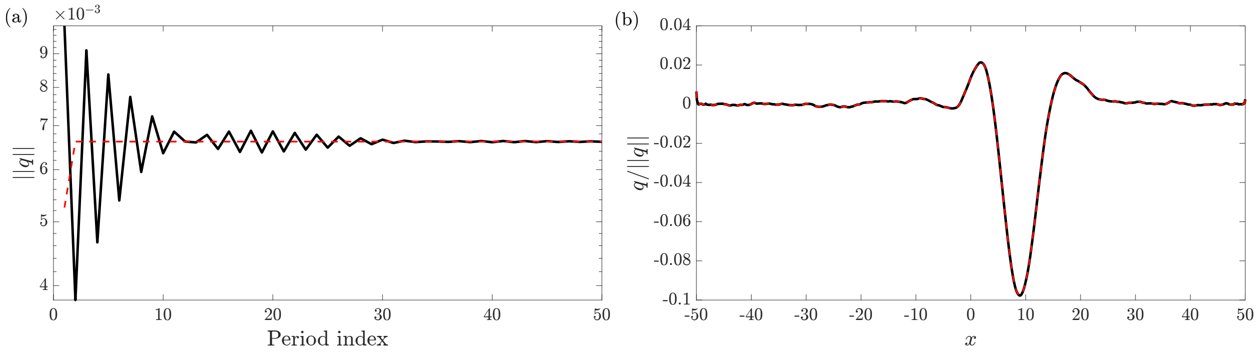


Figure 3.6: Ginzburg-Landau test case: (a) the response norms before (black) and after (red) transient removal at the same phase, *i.e.*, snapshots with $\Delta t = T$ interval. (b) Comparison of the normalized response at the second period in blue, the 50th period in black, and the second period after transient removal in red.

Finally, RSVD-Δt employing an RK4 integration scheme with $dt = 0.003$ and $T_t = 2T$ is carried out along with our efficient transient removal strategy to compute the harmonic resolvent modes for $k = 5$ and $q = 1$. To establish a reference, the harmonic resolvent modes are also computed using RSVD-LU with an equivalent number of test vectors and power iterations. Figure 3.7(a) illustrates that RSVD-Δt computes the five leading gains identically to RSVD-LU. The gain relative error in figure 3.7(b) confirms that relative errors remain below 10^{-9} across all modes. The inner products of response and forcing modes, computed via RSVD-Δt and RSVD-LU, demonstrate parallel directions as the relative errors remain below 10^{-8} across all modes. The inner-product error between two unit-norm vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is defined as

$$e_{ip} = 1 - \langle \boldsymbol{v_1}, \boldsymbol{v_2} \rangle. \tag{3.45}$$

3.9.2 Flow over an airfoil

Our second test case consists of the flow over a NACA0012 airfoil at a low Reynolds number of $Re = 200$ and a steep angle of attack of $\alpha = 20°$, which is dominated by periodic motions. This problem serves as a benchmark for computational cost and accuracy comparisons. A direct numerical simulation is conducted using the "CharLES" compressible flow solver. To mimic the incompressible simulations by Padovan *et al.* [106], we set the Mach number to

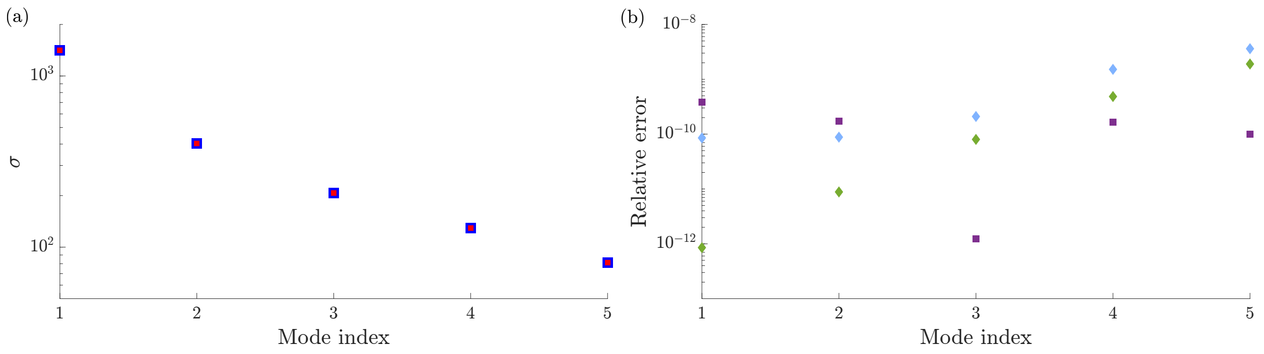


Figure 3.7: Ginzburg-Landau test case: (a) the five leading gains using RSVD-Δt (blue) and RSVD-LU (red). (b) The corresponding gain relative error (purple) and relative inner product errors (3.45) between the response modes (cyan) and the forcing modes (green) computed via RSVD-Δt and RSVD-LU.

0.05. The simulation employs a C-shaped mesh with a total of 62,000 cells. The leading edge of the airfoil is located at the origin $(x/L_c, y/L_c) = (0,0)$, where $L_c = 1$ is the chord length. The computational domain spans $x/L_c \times y/L_c \in [-49, 50] \times [-50, 50]$. A constant time step of $\Delta t U_\infty / L_c = 6.88 \times 10^{-5}$ is used, corresponding to a Courant-Friedrichs-Lewy number of 0.91, where U_∞ is the inflow streamwise velocity. Integration continues for a sufficient duration until the flow reaches a periodic state. Figure 3.8(a) displays the power spectral density computed from the transverse velocity at $(x/L_c, y/L_c) = (3.0, -0.43)$, where a pronounced vortex shedding occurs behind the trailing edge. The shedding frequency is $St_f = 0.114$, where the Strouhal number is defined as $St = \frac{\omega L_c sin(\alpha)}{2\pi U_\infty}$. The consistency between the dominant frequency identified in the power spectral density and obtained from our linearized code has been confirmed in our previous work [73].

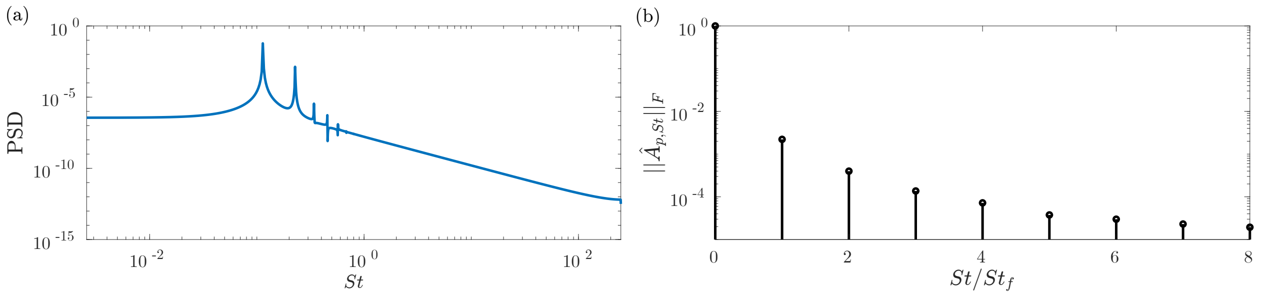


Figure 3.8: Airfoil test case: (a) the PSD spectrum based on transverse velocity at $(x/L_c, y/L_c) = (3.0, -0.43)$. (b) The normalized Frobenius norm of $\hat{\boldsymbol{A}}_{p,St}$ up to the eighth harmonic.

The linearized operators are constructed at 100 time points within $T = 2\pi/St_f \approx 55$. Performing a DFT on the $\boldsymbol{A}_i$, we generate 100 LNS operators $\hat{\boldsymbol{A}}_j$ in Fourier space to be used for harmonic resolvent analysis. The Frobenius norms $||\hat{\boldsymbol{A}}_{p,St}||_F$, depicted in figure 3.8, indicate that $\boldsymbol{A}_p(t)$ can be accurately reconstructed using up to the 5th harmonic. Given that the linearized operator $\boldsymbol{A}_p(t)$ is real-valued for this problem, we retained only $\lfloor N_b/2 \rfloor + 1 = 6$ coefficients, *i.e.*, the zeroth and 5 positive harmonics. For time stepping, $dt = 0.0045$ is chosen to ensure the stability and accuracy of the RK4 integration scheme. The input and output perturbation frequency range spans up to the 7th harmonic, *i.e.*, $N_\omega = 15$. The domain of interest is $x/L_c \times y/L_c \in [-4, 12] \times [-2.5, 2.5]$, identical to that in Padovan *et al.* [106]. We seek optimal forcing and response under the Chu energy norm [24], which has been used in several previous studies [160, 137, 54]. The number of test vectors is set to $k = 5$, and $q = 2$ power iteration proves sufficient for the convergence of both gains and modes.

A homogeneous simulation of the time-periodic system exhibits a slow decay rate, so we employ the transient removal strategy. Three snapshots constitute the trial basis, obtained after a sufficiently long time interval where most initial transients are eliminated. A random forcing, encompassing the set $\Omega_q \in \{-7, -6, ..., 6, 7\} St_f$ frequencies, is applied to the LNS equations for over 30 periods. The snapshot norms slowly approach towards the steady-state norm but remain far from converging, as illustrated in figure 3.9. Utilizing our strategy, the relative norm error falls below the 1% threshold after 20-30 periods in most cases with different random forcings. The efficacy of our strategy is further emphasized in figure 3.9(b), where the 1% relative error is achieved before 10 periods for the optimal forcing (from RSVD-Δt output). Ultimately, we set $T_t = 29T$ and conducted our simulations for a total duration of 30 periods to compute the actions of $\boldsymbol{H}$ and $\boldsymbol{H}^*$. Extrapolating the natural decay rate suggests that, in the absence of our transient-removal strategy, the norms reach the 1% threshold after around 2000 periods, demonstrating over a 60-fold acceleration in time stepping and overall algorithm performance with our strategy. This observation holds true for the adjoint equations as well.

The optimal forcing and response modes (apart from the phase shift mode) are forcing and response modes are shown in figure 3.10. The vorticity patterns of the first output mode of the harmonic resolvent closely predict vortical structures observed in simulations that are induced by sinusoidal perturbations to the periodic base flow across all frequencies[106]. This close match is attributed to the low-rank nature of the harmonic resolvent operator of the airfoil. This agreement also implies that regardless of the forcing type, resultant flow perturbations resemble the suboptimal output modes. Moreover, the proximity of forcing modes to the airfoil suggests sensitivity to perturbations near it. The significance of harmonic resolvent analysis lies in its ability to uncover flow structures that would remain hidden when linearizing around the temporal mean. It reveals the intricate interplay between different frequencies, the base frequency and its harmonics, which collectively contribute to the observed

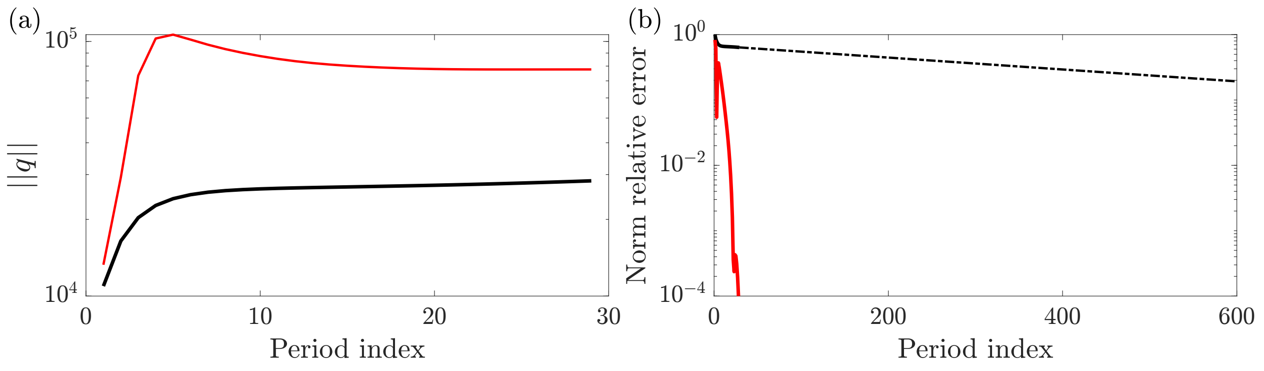


Figure 3.9: Airfoil test case: (a) the response norms before (black) and after (red) transient removal, and the true steady-state norm (blue), at the same phase, *i.e.*, snapshots with $\Delta t = T$ interval, for a random forcing. (b) The relative error between the steady-state response norm and the response norms before (black) and after (red) transient removal.

flow patterns during real perturbation analysis.

The results from RSVD-Δt closely align with the existing data obtained from RSVD-LU by Padovan *et al.* [106], despite variations in the CFD solver, boundary conditions, domain setup, and energy norm. The gain plot in figure 3.11(a) exhibits a similar pattern to Padovan *et al.* [106], with the exception that we have retained the optimal gain associated with the phase shift (the exact values differ due to differences in the problem setup). We noted that the leading mode and gain, *i.e.*, the mode associated with the phase shift, converged without power iterations due to a substantial two-orders-of-magnitude gap between the optimal and suboptimal gains.

From a computational standpoint, a fair comparison between the RSVD-LU and RSVD-Δt algorithms is feasible when both utilize the same parameters. However, the use of $N_b = 11$ proves to be excessively memory-intensive for the RSVD-LU algorithm, exceeding the available 3.5 TB of memory on our cluster. The CPU and memory scaling plots in figure 3.11(b) and (c), respectively, depict the LU decomposition of $\boldsymbol{T}$ with $N_b = 3$, maintaining a constant N while varying the number of blocks N_ω from 5 to 15 with $\Delta N_\omega = 2$. On the same plots, we present the cost of computing the action of $\boldsymbol{H}$ on a vector using time-stepping. The total duration is set to 30 periods to obtain an accurate solution. We only vary N_ω while keeping $N_b = 3$, the overall cost of RSVD-Δt remains constant and unaffected by changes in N_ω. This implies that the creation of LNS operators and the time-stepping process incur significantly higher costs compared to the transformations between Fourier space and the time domain for forcing and response. The memory of RSVD-Δt to store LNS operators remains independent of N_ω, as expected (see §3.7.1), while storing $\boldsymbol{Q}$ and $\boldsymbol{F}$ matrices grows linearly with the number of frequencies N_ω.

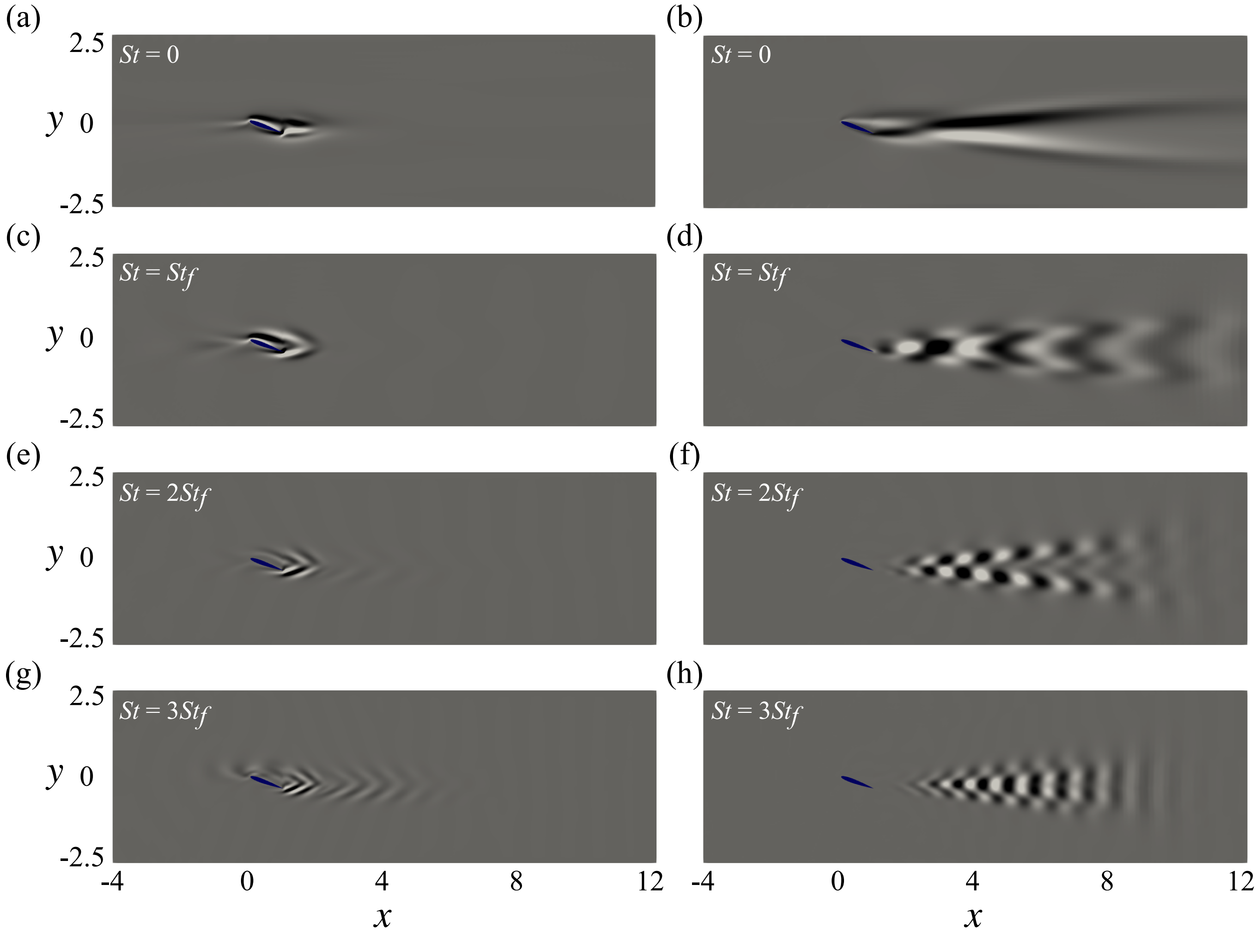


Figure 3.10: Airfoil test case: real part of the vorticity field computed from (a, c, e, g) the input mode and (b, d, f, h) the output mode associated with the first suboptimal gain.

RSVD-LU is estimated to require 1543 CPU-hours for the LU decomposition of $\boldsymbol{T}$, and an additional 15 CPU-hours for solving the LU-decomposed system for each test vector, yielding a total cost of $\mathrm{CPU}_{\text{RSVD-LU}} \approx 1543 + 15 \times 5 \times 2 \times 2 = 1843$ CPU-hours for $k = 5, q = 1$. On the other hand, employing RSVD-Δt with the aforementioned time-stepping parameters and $N_b = 11$ and $N_\omega = 15$ incurs a cost of approximately 7 CPU-hours per period for a test vector, resulting in a total cost of $\mathrm{CPU}_{\text{RSVD-}\Delta t} \approx 30 \times 7 \times 5 \times 2 \times 2 = 4200$ CPU-hours. The cost of QR decomposition and the final SVD are ignored as they are orders of magnitude faster. In terms of memory consumption, RSVD-LU peaks at $\mathrm{RAM}_{\text{RSVD-LU}} \approx 2036GB$ for the LU decomposition of $\boldsymbol{T}$. RSVD-Δt, on the other hand, stores $\lfloor N_b/2 \rfloor + 1 = 6$ sparse matrices, each of size 0.4 GB, and three dense matrices of size $Nk(\lfloor N_\omega/2 \rfloor + 1) = 313630 \times 5 \times 8$, totaling $\mathrm{RAM}_{\text{RSVD-}\Delta t} \approx 0.4 \times 6 + 0.185 \times 3 \approx 3GB$. This translates to a memory saving of approximately three orders of magnitude, even with an unbalanced number of base frequencies. Overcoming the memory consumption hurdle, which is typically the limiting factor in practice, RSVD-Δt

emerges as a viable tool for analyzing larger-scale flows compared to RSVD-LU.

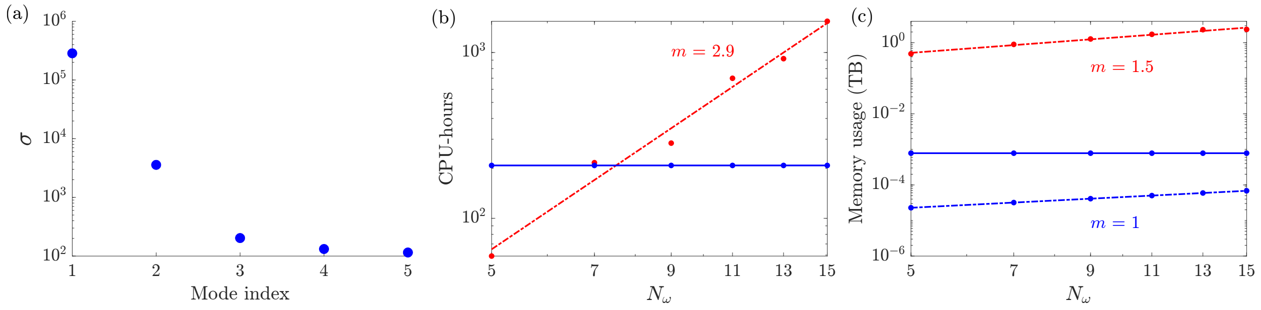


Figure 3.11: Airfoil test case: (a) the five leading gains using RSVD-Δt . (b) The CPU-hour, and (c) memory usage scaling of RSVD-LU (red) and RSVD-Δt (blue) to compute the action of $\boldsymbol{H}$ onto a vector, *i.e.*, $k = 1$. The memory usage of RSVD-Δt is decomposed into memory required to store LNS operators (solid) and forcing and response matrices (dashed).

3.10 Chapter summary

In summary, this chapter has demonstrated the robustness and scalability of the RSVD-Δt algorithm for harmonic resolvent analysis, particularly in addressing the complexities of multiple frequency coupling and singularities. By introducing a unique approach to optimizing simulation length, we have enhanced the algorithm's efficiency beyond traditional methods. The successful validation against the periodic Ginzburg-Landau system and the computation of harmonic resolvent modes for flow over an airfoil underscore its practical utility. These contributions significantly advance the application of harmonic resolvent analysis in fluid dynamics, offering a scalable solution for analyzing time-periodic flows.

CHAPTER 4

Applications

In earlier sections, we applied our algorithm to one-, two-, three-dimensional systems characterized by statistically stationary and periodic mean flows that had been previously studied in the literature. This effort was aimed at validating the algorithm, ensuring the reliability of the computed modes, and verifying theoretical scaling estimates for CPU time and memory requirements in large-scale flow simulations. In this chapter, we extend our algorithm's application to large-scale flows that have yet to be explored through resolvent or harmonic resolvent analyses due to computational limitations. The first two applications, discussed in §4.1 and §4.2, present preliminary results focused on computing resolvent modes without a deep dive into the underlying physics. The third application, detailed in §4.3, offers a more comprehensive analysis of a screeching supersonic jet, uncovering a connection between experimental observations and the interaction between response modes and the non-stationary motion of the mean flow. This discovery enhances our understanding of the jet screech, which continues to be a key area of research in aeroacoustics.

4.1 Jet with streaks

First, we apply the RSVD-Δt algorithm to study the impact of streaks on other coherent structures within a turbulent jet.

4.1.1 Background and motivation

Streaks – elongated regions of low-velocity fluid – have historically been observed and studied in turbulent channel flows (see McKeon [94] and Jiménez [67] and the references therein). More recently, in unbounded shear flows such as round jet flows, streaks have been shown to be generated via the evolution of optimal initial conditions that maximize the transient energy growth [68]. Nogueira *et al.* [102] and Pickering *et al.* [111] showed that streaks emerge as the dominant structures in the SPOD and resolvent spectra of jets at very low

Algorithm	CPU time (hours)			Memory (GB)
	Total	Action of $\boldsymbol{R}$ and $\boldsymbol{R}^*$	SVD/QR	
RSVD-LU	7.53×10^7	7.53×10^7	0.762	1.33×10^5
RSVD-Δt	1.83×10^4	1.83×10^4	0.762	7.36×10^2

Table 4.1: Comparison of the RSVD-LU and RSVD-Δt algorithms in terms of CPU time and memory usage for the three-dimensional jet with $N \approx 39M, N_\omega = 21, k = 10$, and $q = 1$. The action of $\boldsymbol{R}$ and $\boldsymbol{R}^*$ use time stepping for RSVD-Δt and a direct solver for the RSVD-LU algorithm.

frequencies when $m \geq 1$. Streaks are produced via a lift-up mechanism applied to the rolls or streamwise vortices that are usually excited near the nozzle exit. The presence of streaks within turbulence modifies the flow quite significantly. In particular, optimal streaks are shown to stabilize the KH wavepackets in a parallel plane shear layer [88] and Tollmien–Schlichting waves in the Blasius boundary layer [26]. Similar findings on a high-speed turbulent jet by Wang *et al.* [165] demonstrate the stabilizing effects of finite-amplitude streaks on KH wavepackets. In this study, we investigate the impact of streaks on the linear amplification and spatial structure of the KH wavepackets described by the leading resolvent modes via a secondary stability analysis.

4.1.2 Results and discussion

The streaks that will be added to the mean flow are obtained from an initial resolvent analysis of the mean flow; specifically, streaks are the optimal resolvent response at very low frequencies [111]. Due to the symmetry of the mean jet, streaks obtained from data via SPOD or computed using resolvent analysis are associated with a particular azimuthal wavenumber. Accordingly, we compute the streaks using our axisymmetric code, which produces the same results as the three-dimensional code but at a lower cost. We compute them for $(St, l) = (0, 6)$, where l denotes the azimuthal periodicity of the streaks. This choice of $l = 6$ corresponds to one of the main cases studied in Wang *et al.* [165].

The updated mean flow with the streaks added has 6-fold rotational symmetry and, following Sinha *et al.* [138], can be written as

$$\bar{\boldsymbol{q}}(x, r, \theta) = \sum_{j=-\infty}^{\infty} \hat{\bar{\boldsymbol{q}}}_{lj}(x, r) e^{ilj\theta}. \tag{4.1}$$

They proved that after plugging the Fourier ansatz of the resulting mean flow into the LNS

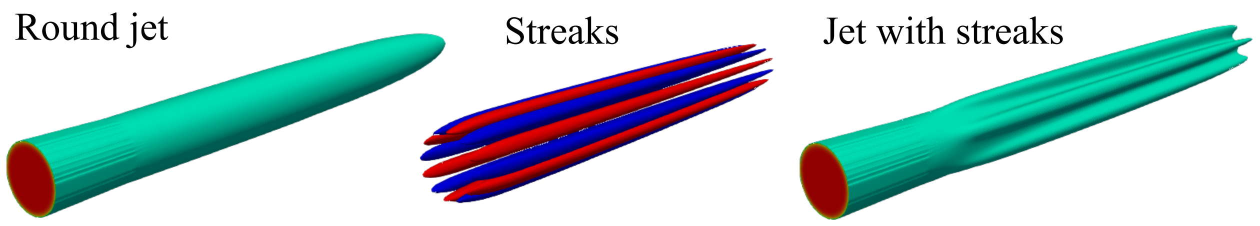


Figure 4.1: The mean streamwise velocity of the three-dimensional round jet and jet with streaks. The jet with streaks is obtained by adding the streaks with an azimuthal wavenumber of 6 to the mean flow of the round jet.

equations, given an azimuthal wavenumber m, the associated axisymmetric mode $\hat{\boldsymbol{q}}_{m,\omega}$ can only couple with $\hat{\boldsymbol{q}}_{m-lj,\omega}$ for $j \in \mathbb{Z}$. In our problem, $l = 6$ and sorting the modes with the lowest azimuthal modes, we expect coupling of modes in sets of $\boldsymbol{q}^L_\omega = \{\hat{\boldsymbol{q}}_{L-lj,\omega}\}_{l=-\infty}^{l=\infty}$, where $L = \{-2,-1,0,1,2,3\}$ includes all possibilities. Indexing in this manner implies that the modes with $L = 0,3$ are unpaired while $L = \pm1,\pm2$ will show up in pairs in the three-dimensional setup due to symmetry.

The streaks' shape and amplitude are sensitive to a few parameters including the viscosity (or equivalently turbulent Reynolds number or eddy-viscosity model if desired) and forcing region. In lieu of a more complex eddy-viscosity model, we use a constant turbulent Reynolds number of $Re = 1000$. This value is close to the optimal frequency-dependent value determined by Pickering *et al.* [112] for $St = 0$ as well as most of our frequency range of interest $St \in [0,1]$ for the secondary stability problem. Additionally, the forcing region of the resolvent analysis used to compute the streaks must be limited to obtain streaks of finite streamwise length. If the domain is not limited, the forcing rolls that generate these streaks sustain them throughout the domain. After some trial and error, we limited the forcing region to $x,r \in [0,1]\times[0,1]$, which produced streaks with a location of peak amplitude ($x \in [5,6]$) and overall shape consistent with the streak SPOD modes obtained by Nogueira *et al.* [102].

Once the axisymmetric streaks are computed, the three-dimensional streaks are obtained by revolving them around the $x-$axis with phase $e^{il\theta}$ as shown in figure 4.1. A tuning variable is the amplitude (or strength) of the streaks. The amplitude is defined as the ratio of the peak streamwise velocity of streaks over M. According to Wang *et al.* [165], the amplitude of these structures grows linearly over time. Therefore, no *correct* constant amplitude exists for our secondary analysis. The amplitude of streaks in our thesis is set to 40%, which is large enough to affect the modes compared to the round jet. The region of interest and grid points along with all the other parameters are the same as for the round jet.

RSVD-Δt is used to compute the resolvent modes for the modified mean flow. The number

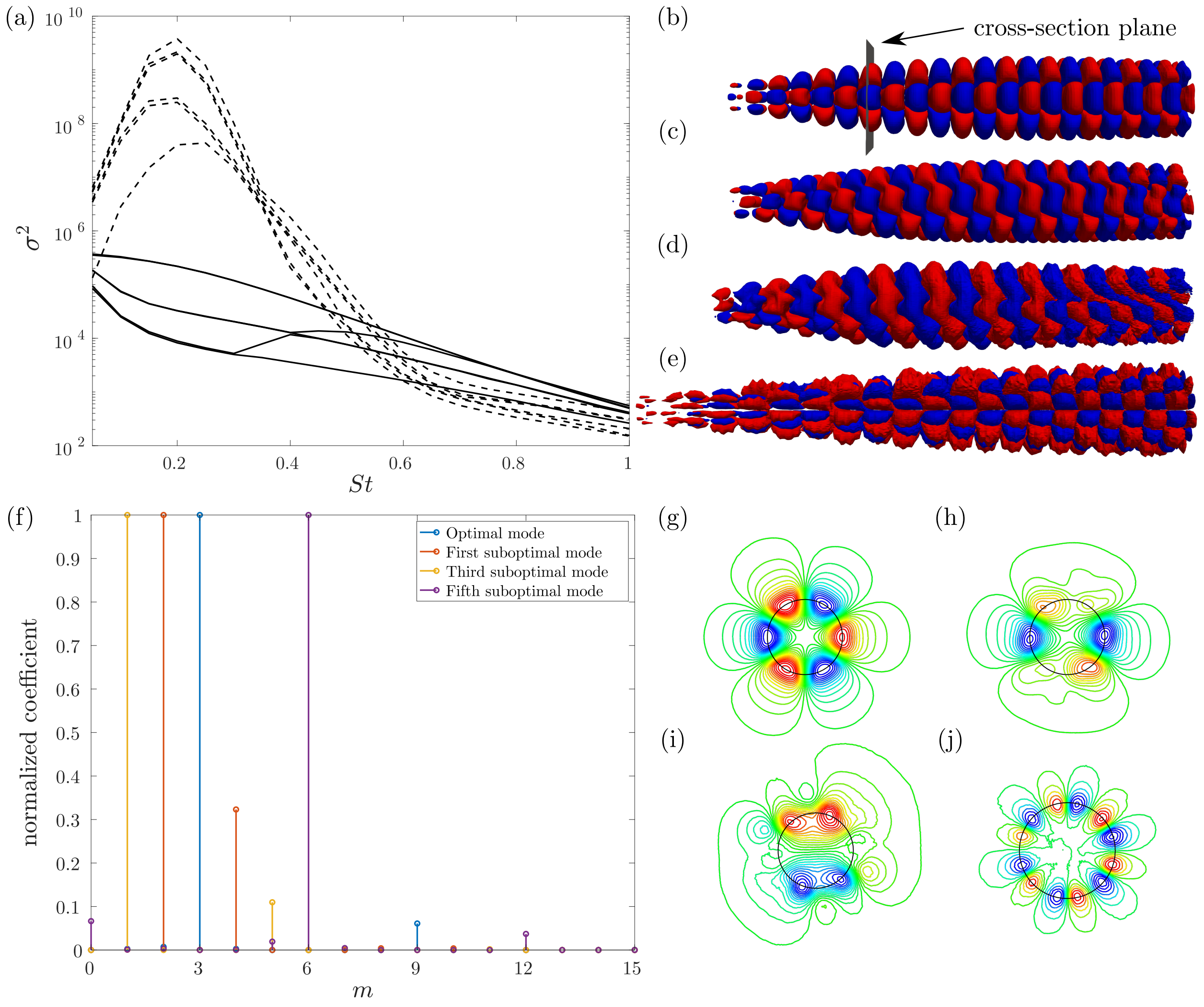


Figure 4.2: Results for the jet with streaks: (a) resolvent gains for the round jet (solid line) and jet with streaks (dashed line); (b-e) the optimal, first, third, and fifth suboptimal pressure responses at $St = 0.2$; (g-j) contours of the pressure responses on cross-section at $x = 8.5$ corresponding to (b-e), respectively. Fourier transforms are taken along the black circles shown to obtain the corresponding azimuthal wavenumber spectra for each mode shown in (f).

of test vectors is $k = 10$ and the gains are reported after $q = 2$ power iterations. The first few leading modes converged after the first power iteration, but an extra power iteration is performed to ensure convergence since no ground truth results are available for comparison. The frequency range $St \in [0, 1]$ and discretization $\Delta St = 0.05$ are the same as used for the round jet in §2.10.2. The time-stepping scheme is RK4 with $dt = 0.00625$. Transient errors are held below 1% for $St > 0$ via our transient removal strategy using Galerkin projection with the matrix of snapshots with a duration $T_t = 3T_s$.

The gains for the round jet and jet with streaks are compared in figure 4.2(a). The streaks have increased the gains by orders of magnitude for $St < 0.5$. Some of the gains appear in pairs, indicating mode pairs analogous to those described for the round jet, which arise due to the six-fold symmetry of the mean jet with streaks. The match occurs between the first and second suboptimal in addition to the third and fourth suboptimal modes. All modes almost coincide at $St = 0.35$ and continue decaying as St increases.

The optimal, first, third, and fifth suboptimal pressure response modes at $St = 0.2$, where the leading gain is maximum, are shown in figure 4.2. The second and fourth suboptimal modes are not shown since they are pairs with the first and third suboptimal modes, respectively. The three-dimensional iso-surfaces show KH wavepackets that are significantly altered by the streaks; characterizing the modes with the indexes defined earlier requires deeper investigation. To this end, cross-section contours at $x = 8.5$ are plotted. These plots are more complicated than the round jet due to the coupling between multiple azimuthal wavenumbers. We interpolate the pressure field on the circles shown on each contour plot to demonstrate the coupling azimuthal wavenumbers. Taking an FFT of the extracted data, the normalized coefficients are plotted against m in 4.2(f). This plot shows that the optimal mode is comprised of $L = 3$ with a larger weight and $L + l = 3 + 6 = 9$ with a smaller weight, which is consistent with our axisymmetric analysis. The first suboptimal mode includes $(L, L - l) = (2, -4)$, and its pair contains $(L, L + l) = (-2, 4)$, so both couplings and pairings are as expected. Similarly, the third mode is a coupling between $(L, L - l) = (1, -5)$, and the fourth mode is with $(L, L + l) = (-1, 5)$. Lastly, the fifth mode is unpaired and captures the $(L, L + l) = (0, 6)$ azimuthal wavenumbers with a small signature of $L + 2l = 12$.

From the perspective of computational cost, the jet with streaks is similar to the three-dimensional discretization of the round jet considered in §2.10.2.2. Utilizing the RSVD-LU algorithm for the same grid with state dimension $N \approx 39$ million, the anticipated CPU time surpasses 75 million hours. Nevertheless, leveraging RSVD-Δt with $q = 2$ enabled us to complete the analysis within 37 thousand CPU-hours. Our computations used 300 cores, which results in a wall time of 28 years for the RSVD-LU algorithm and 123 hours for our algorithm. Additionally, memory requirements amount to more than 130 TB for the RSVD-LU algorithm and 0.75 TB for ours. It is safe to say that this analysis would have been intractable using previous algorithms, demonstrating the promise of the RSVD-Δt algorithm for extending the applicability of resolvent analysis to new problems in fluid mechanics.

4.2 Twin jets

Second, we utilize the RSVD-Δt algorithm to examine the influence of nozzle-to-nozzle spacing on KH wavepackets within twin round turbulent jets.

4.2.1 Background and motivation

Twin jets, that is a pair of identical jets separated by some distance, have garnered significant attention due to their complex flow dynamics and relevance to engineering applications such as aerospace propulsion, environmental fluid mechanics, and jet-noise control. Understanding twin jet behavior is essential for optimizing designs in these fields. The computational challenges of accurately simulating these interactions have led to a focus on numerical methods, with pioneering studies including Goparaju & Gaitonde [51] on jet-to-jet interaction using LES and Gao *et al.* [47] on symmetrical flapping modes in twin circular jets.

Research into coherent structures within turbulent jets began with Crow & Champagne [27], who identified large-scale structures similar to instability waves in harmonically forced supersonic jets. This work has since guided efforts to model primary jet noise sources through linear stability analyses, particularly in subsonic and supersonic jets [60, 19]. Recent advancements, especially in SPOD, have improved our understanding of twin jets by identifying wavepackets in jet shear layers and analyzing far-field acoustic radiation [137]. Most studies have focused on axisymmetric configurations, but non-axisymmetric cases like twin jets introduce additional challenges due to reflection symmetries. Addressing these complexities, Esfahani *et al.* [40] developed an SPOD workflow that respects such symmetries, reducing computational costs by breaking down the problem into smaller components.

While single round jets have been extensively studied, twin jets present additional mathematical complexities. The inherent symmetry in single jets allows for simplified analyses using azimuthal Fourier modes, but twin jets require more sophisticated approaches to capture inter-jet interactions. Early studies like Morris [100] used bipolar coordinates to investigate inviscid instability in parallel jets, showing significant interactions when jet separation was less than three nozzle diameters. Subsequent research by Rodríguez *et al.* [121] and Nogueira & Edgington-Mitchell [103] applied two-dimensional cross-stream discretizations to explore

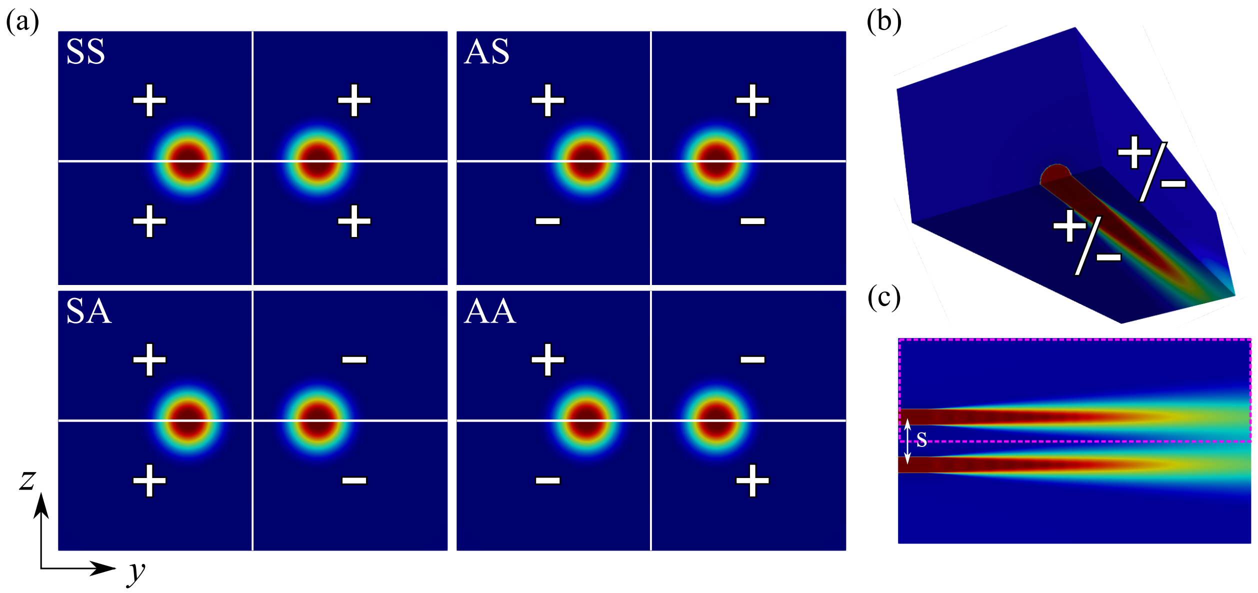


Figure 4.3: Illustration of the computational domain and boundary conditions for the twin jet. (a) The D_2 symmetry boundary condition is shown. (b) The computational domain is depicted as a quarter of the full domain, highlighting the surfaces where D_2 symmetry is applied. (c) The $y-z$ plane view shows the distance s/D between the two poles. Streamwise velocity is displayed in all plots.

wavepacket structures in twin jets. More recently, a plane-marching PSE method has been used to model wavepackets in turbulent twin jets [120]. Building on this, our research utilizes global resolvent analysis to examine how the spacing between nozzles, *i.e.*, center-to-center nozzle spacing, affects resolvent gains and response modes in twin jets.

While PSE provides a balance between accuracy and computational efficiency for predicting near-field wavepackets, more detailed approaches like global eigenmode and resolvent analyses, despite their higher computational cost, offer deeper insights into wavepacket dynamics. The main hindrance to doing global analyses of twin jets has been the high cost due to the high state dimension and poor scaling for three-dimensional problems. RSVD-Δt emerges as a viable tool in reducing the computational demands of computing resolvent modes of such systems.

4.2.2 Results and discussion

In this thesis, the mean flow of the twin jets is constructed by combining the mean flows of two separate single jets with a nozzle spacing s/D between their centers. The mean flow data of the single jet is sourced from the same database we used for our turbulent jet in §2.10.2 [13]. Specifically, we use LES results for a supersonic jet under isothermal and ideally expanded conditions, characterized by a Mach number of $M = 1.5$ and a Reynolds number of $Re = 1.76 \times 10^6$. For further details on the simulation methodology, including the grid setup, LES procedures, and subgrid-scale modeling, interested readers are referred to Brès *et al.* [13], where the "CharLES" compressible flow solver was used.

While prior studies have explored the influence of s/D on turbulence mixing [101], near-field pressure fluctuations of opposite flapping modes [1], velocity profiles [79], and local stability sensitivity [122], our research is uniquely focused on the impact of s/D on resolvent modes and gains. Given the time-independence of the mean flow, encompassing velocity components, pressure, and specific volume, resolvent analysis is performed to provide deeper insights into the flow dynamics.

To conduct our analysis, we utilize the D_2 symmetries inherent in the twin jet, following several studies [40, 172, 122, 144]. The symmetric (S) and anti-symmetric (A) conditions imposed at the $y = 0$ and $z = 0$ planes divide the analysis into four families: SS, AS, SA, and AA boundary conditions, as illustrated in figure 4.3(a). Due to the presence of two symmetry planes, only a quarter of the entire domain needs to be considered, effectively reducing the computational domain to one-fourth of the full domain. We solve for four boundary conditions, which is equivalent to solving for the entire domain without applying D_2 symmetry conditions. In practice, RSVD-Δt scales linearly with the domain size, so both scenarios would require approximately the same computational time. However, applying these symmetry conditions allows us to isolate modes corresponding to specific boundary conditions, facilitating direct comparisons with other stability analyses in the literature.

The resolvent modes and gains are computed for a domain extending from $x \in [0, 15]$ and $y \times z \in [0, 3] \times [0, 3]$ for the case where $s = 2D$, *i.e.*, $s/D = 2$, as illustrated in figure 4.3(b). As the spacing between the poles increases to $s/D = 3$ and $s/D = 5$, the domain size in the y direction expands correspondingly to $y \in [0, 4.5]$ and $y \in [0, 5.5]$. The farfield and inlet boundaries are surrounded by sponge layers. The domain is large enough to accommodate the sizeable structures of interest, and the mesh resolution is sufficient to capture these structures accurately over the frequency range of interest. Our objective is to compute the modes within the range of $St \in [0, 1]$ with $\Delta St = 0.05$, as this range is known for its low-rank behavior at certain frequencies where KH instability is predominant. For consistency with the analysis in §4.1, the effective Reynolds number is set to $Re = 1000$.

We employed RSVD-Δt for each combination of pole distance and boundary condition separately, resulting in a total of $3 \times 4 = 12$ simulations. For all cases, we set $k = 8$ and $q = 1$, as our primary interest lies in the top three modes. The transient length was set to $T_t = 4T_s$, where $T_s = 20$, using the RK4 integration scheme with a time step of $dt = 0.045$. Our transient removal strategy was applied after the fourth period; this reduced the simulation

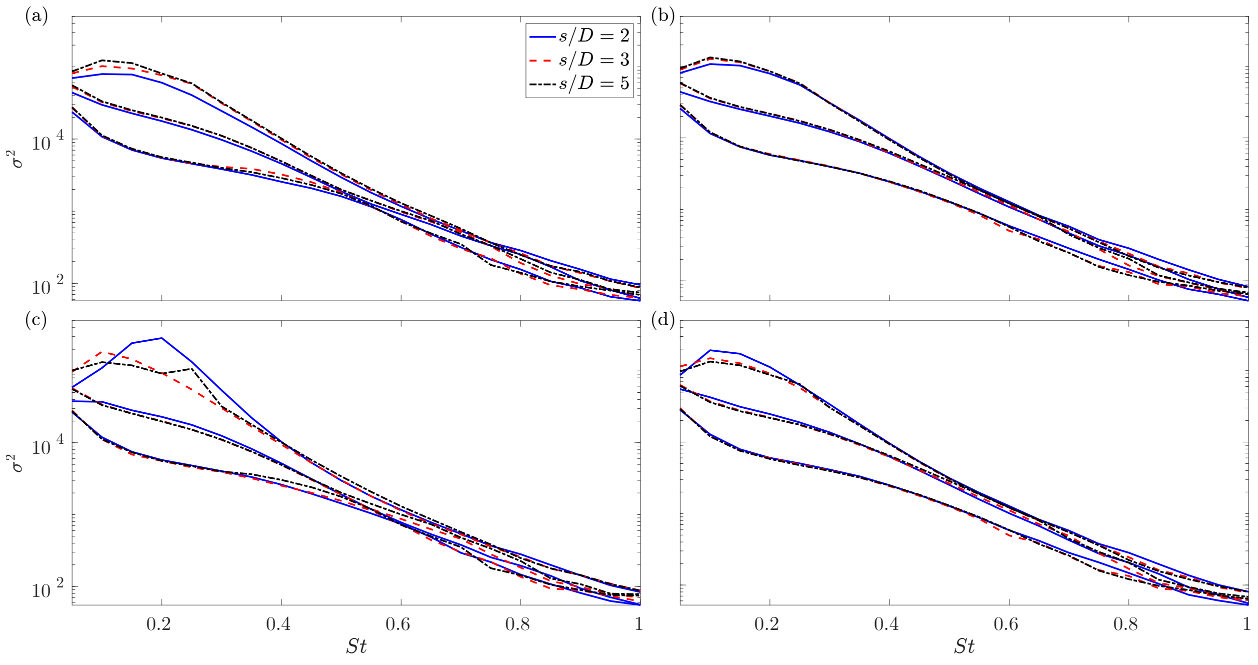


Figure 4.4: Resolvent gains for the twin jet: the three leading gains are plotted for three values of s/D for (a) SS, (b) AS, (c) SA, and (d) AA boundary conditions.

length by a factor of five for the same level of accuracy.

The gains for all cases are presented in figure 4.4. Two key observations can be drawn from these plots; first, for the SS, AS, and AA families, the spacing s/D does not significantly impact the results. The exception is the SA boundary condition, where at lower frequencies, $St \leq 0.3$, the dominant mode exhibits a pronounced peak, and the gap between the optimal and first suboptimal mode exceeds one order of magnitude. Second, as s/D increases, the gain plot increasingly resembles that of a single jet (compare with figure 2.11), as expected, since the twin jets begin to behave like two separate isolated jets.

The pressure modes of the optimal responses for $s/D = 3$ at the peak frequency of $St = 0.2$ are presented from various views in figure 4.5. As expected, the modes feature two KH instability waves interacting with each other. This interaction leads to the coupling of the unsteady pressure fields of the two jets but does not fundamentally alter the KH form observed in a single jet for this spacing ($s/D = 3$). The optimal mode for all four families corresponds to the $m = 1$ mode of a single jet, in the extreme case where the jets become decoupled due to increased spacing. The mode shapes differ from those obtained through resolvent analysis of a single jet, as illustrated in figure 2.12, due to the application of D_2 symmetry boundary conditions. These conditions cause the individual modes of the different solution families to correspond to flapping oscillations of the jet plumes rather than helical motions. To better compare the modes with those in figure 2.12, we combined the

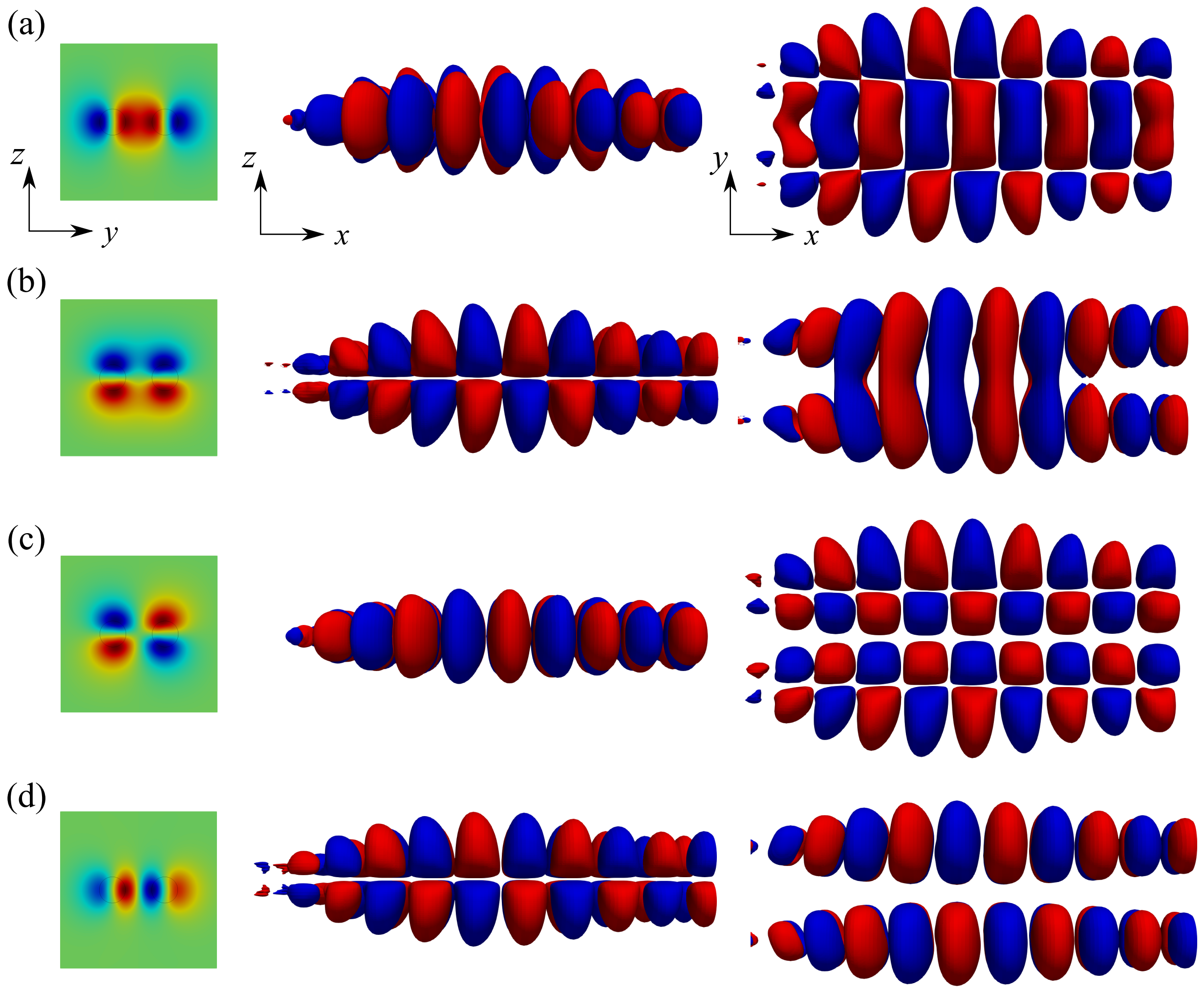


Figure 4.5: Resolvent optimal response modes for the twin jet with $s/D = 3$: the left panel shows the pressure field at $x \approx 5$, the middle panel displays isocontours of the pressure from the side view, and the right panel shows isocontours of the pressure from the top view for (a) SS, (b) AS, (c) SA, and (d) AA boundary conditions.

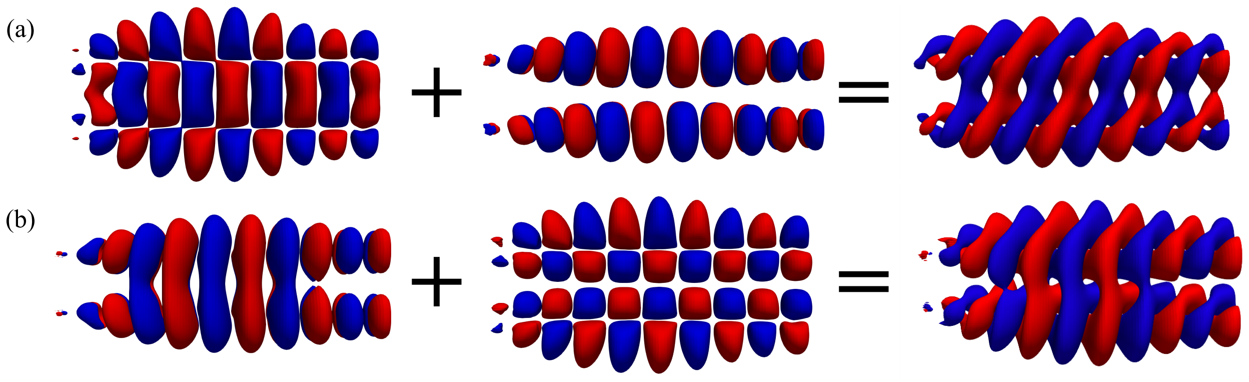


Figure 4.6: Resolvent optimal response modes for the twin jet with $s/D = 3$: the pressure isocontours are combined for (a) SS and AA, and (b) AS and SA, to reveal the helical modes in each row.

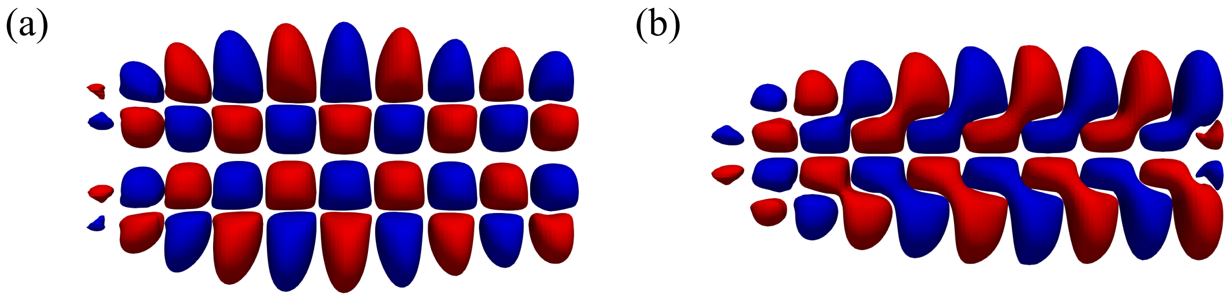


Figure 4.7: Resolvent optimal response modes for the twin jet: pressure isocontours are shown for (a) $s/D = 3$ and (b) $s/D = 2$ under SA boundary condition. The SA mode undergoes more significant alterations when $s/D = 2$.

phase-matched modes of AA and SS, and SA and AS, resulting in interconnected helical KH response modes, as shown in figure 4.6.

As the gain values indicate, most modes across different values of s/D appear similar, with the primary difference being that for $s/D = 5$, the interaction between the KH modes of the two jets is broader and weaker, while it becomes narrower and stronger for $s/D = 3$ and $s/D = 2$. This observation holds true across all gain values for the SS, AS, and AA cases. However, the SA mode undergoes more significant alterations and is particularly sensitive to the distance between the jets. Rodríguez *et al.* [122] observed a similar phenomenon in their analysis of a twin round jet with varying nozzle spacing. They conducted LST and found that the change of eigenvalues associated with the maximum growth modes was most pronounced for the SA modes, which involve counter-phase lateral flapping oscillations in the plane containing the jet axes. Based on their analysis, as the nozzle spacing decreases, this shift becomes significantly more pronounced. In figure 4.7, the mode associated with the optimal response at $St = 0.2$ for the SA case with $s/D = 2$ is compared to that obtained for $s/D = 3$, clearly showing the alteration of the KH mode at this flapping mode.

These preliminary results from computing three-dimensional global resolvent modes of twin jets with a synthetic mean flow suggest that the flapping mode associated with the SA boundary condition is highly sensitive to nozzle spacing. This sensitivity opens up an intriguing avenue for further research. A promising direction for future work is to explore resolvent modes for twin jets using a more realistic mean flow from simulations or experiments and to further investigate the mechanisms behind the significant changes observed in the SA mode.

4.3 Screeching jet

Finally, we employ the RSVD-Δt algorithm to investigate the underlying physics and nonlinear mechanisms critical to understanding jet screech.

4.3.1 Background and motivation

Screeching jets are characterized by intense acoustic emissions resulting from a feedback loop between downstream-traveling KH waves and upstream-traveling acoustic waves called guided jet modes, coupled by shock cells. They have been a topic of considerable interest due to their impact on aircraft noise and structural fatigue. The high-frequency noise generated by screeching jets poses significant challenges in aerospace engineering, particularly in the design of exhaust systems and noise mitigation strategies. The study of screeching mechanisms has evolved through both experimental and computational approaches, with foundational research by Tam & Tanna [151] on the acoustic feedback loop in under-expanded jets and Raman [115] on the categorization of screech tones in different jet configurations.

The identification of screech modes in turbulent jets began with Powell [114], who recognized discrete tones in the acoustic spectrum of supersonic jets and rightly attributed them to a resonance loop. This discovery has been pivotal in the development of models that predict screech frequencies as a function of flow parameters such as Mach number and nozzle geometry [150, 56]. Linear stability theory has been instrumental in understanding the conditions under which screeching occurs, offering predictions of instability growth rates and the influence of shock structures on sound generation. More recently, advanced computational techniques such as LES and DNS have provided detailed insights into the complex interactions that drive screeching in jets, albeit at significant computational cost [49, 34].

The physics behind screeching jets involve several mechanisms, including KH instability, shock wave interaction, and wave superposition. The KH wavepacket, formed due to shear layer instabilities, interacts with shock cells, generating both upstream and downstream-traveling waves. The combination of distinct wave structures, including downstream-traveling KH modes and upstream-traveling guided acoustic modes, leads to the periodic modulation of velocity fluctuations, thereby contributing to the screech phenomenon.

Despite advancements in linear stability theory, the validity of linear models in capturing the non-linear processes involved in jet screech remains uncertain [35]. While linear analyses can predict screech tones with reasonable accuracy, the complex interactions and turbulence dynamics complicate modeling efforts. Ongoing research aims to bridge the gap between experimental observations and theoretical predictions, addressing open questions regarding frequency selection mechanisms, mode competition, and energy transfer pathways.

In this thesis, we employ harmonic resolvent analysis as a tool to understand the mechanisms leading to the screech phenomenon. This approach allows the decomposition of jet dynamics into harmonic resolvent modes, facilitating the identification of dominant linear mechanisms and underlying energetic structures contributing to the screech process. Harmonic resolvent analysis captures triadic interactions, which can be crucial in transferring energy between modes or interacting with periodic motions of the mean flow, thereby opening new avenues for examining mode shapes and the energy spectral distribution.

4.3.2 Results and discussion

To construct the mean flow field, we utilize a two-dimensional planar velocity field in the streamwise and radial directions, corresponding to the axisymmetric mode ($m = 0$). This data is obtained through Particle Image Velocimetry (PIV) from Monash University, which captures the presence of weak shock cells within the flow. The PIV data include mean streamwise and spanwise velocities for the screeching jet (A1 mode) at a Mach number of 1 and a nozzle pressure ratio of 2.1. The computational domain of interest spans $x \times r \in [0, 15] \times [0, 3]$, with an extension in the streamwise direction from $x = 10$, based on the experimental data to ensure a self-similar velocity profile. Other flow variables, such as pressure and specific volume, are obtained assuming a zero mean pressure gradient and using the Crocco-Busemann relation and the state equation. The domain is discretized using a grid resolution of $N_r \times N_x = 200 \times 400$ in the radial and streamwise directions, respectively. The LNS operator is then constructed by linearizing the mean flow and applying fourth-order summation-by-parts finite difference schemes for spatial derivatives, following methodologies consistent with prior jet flow studies in this work. The global states are functions of x and

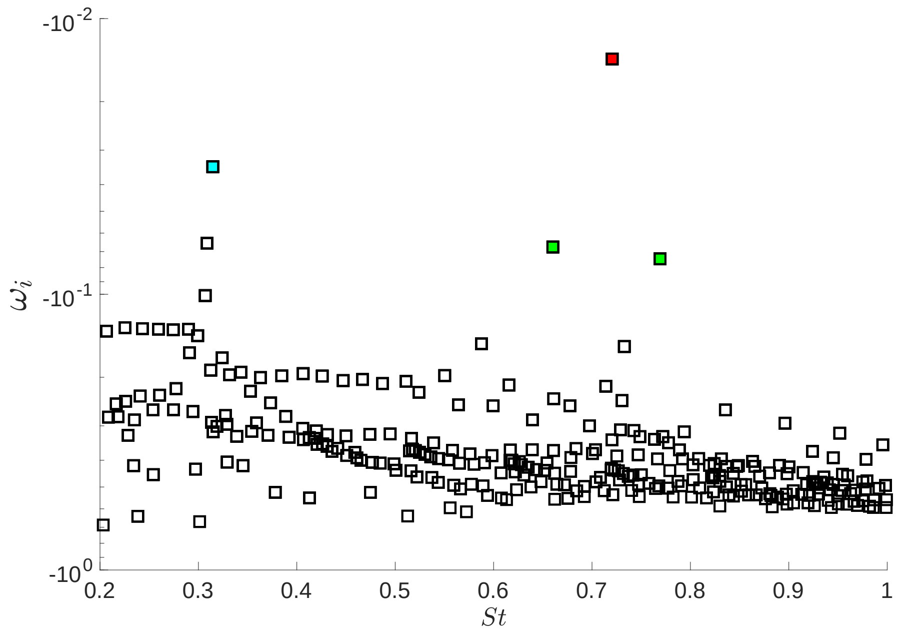


Figure 4.8: Eigenspectrum of the linearized operator around the mean flow with $Ma = 1$. The red square is the least-stable mode. The green and cyan squares are the other isolated modes.

r and can be written in the form of

$$\boldsymbol{q}(x, r, \theta, t) = \hat{\boldsymbol{q}}(x, r)e^{\mathrm{i}\omega t + \mathrm{i}m\theta}. \tag{4.2}$$

We begin our analysis by computing the eigenmode associated with the screech phenomenon using global stability analysis. The eigenspectrum of the LNS operator around the original base flow is shown in figure 4.8. The screech mode is characterized by a downstream-traveling KH wavepacket and an upstream-traveling guided jet mode, as described by Edgington-Mitchell *et al.* [36]. This mode is expected to occur at a St close to the experimentally observed screech frequency. The reported screech frequency is $St = 0.7140$, and our analysis identifies the least stable mode at $St = 0.72$, with a relative error of less than 1%. The mode exhibits the anticipated characteristics, as depicted in figure 4.9(e, f). The pressure field shows the presence of both the KH wavepacket near the shear layer and the guided jet mode within the core region, confirming that this mode corresponds to the screech phenomenon.

In addition to the primary screech mode, we also identify an eigenmode at $St \approx 0.31$, which is associated with resonance between two guided jet modes. As the pressure field indicates, this mode is confined to the core region. Furthermore, two other modes around the screech frequency, at $St = 0.66$ and $St = 0.77$, exhibit characteristics similar to those

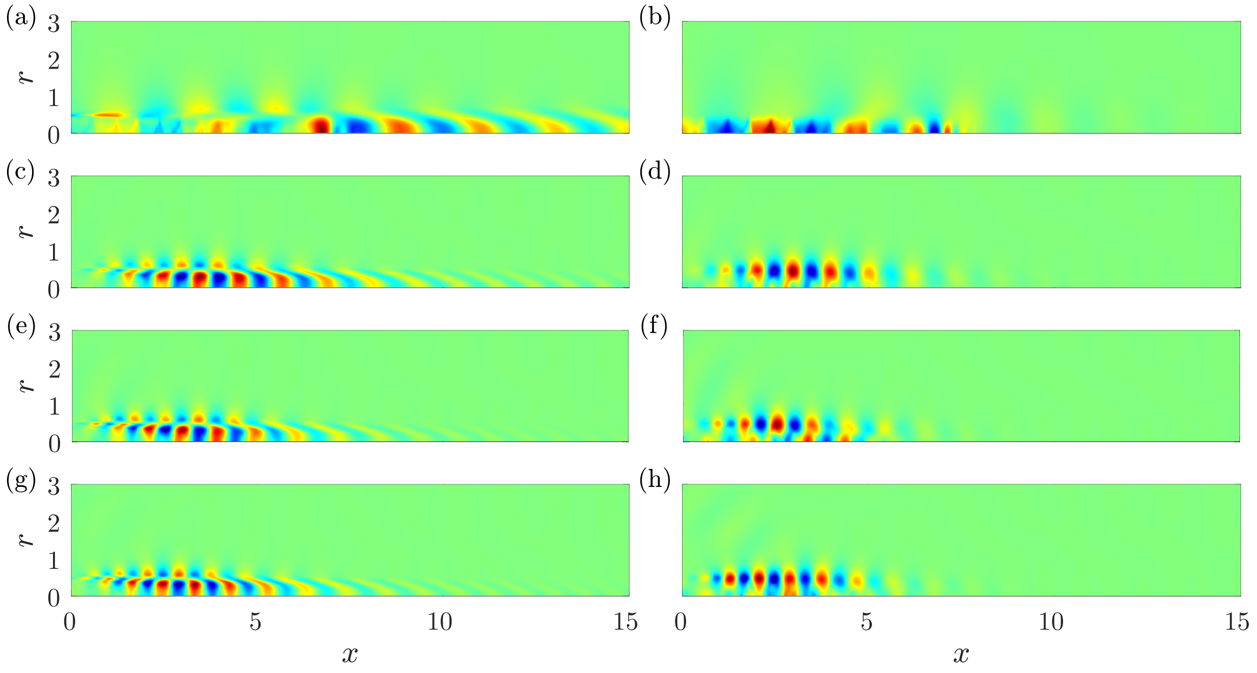


Figure 4.9: The eigenmodes associated with isolated eigenvalues of the screeching jet. Panels (a, c, e, g) show the streamwise velocity, and panels (b, d, f, h) show the pressure of the eigenmodes at frequencies of 0.31 (cyan), 0.66 (green), 0.72 (red), and 0.77 (green), respectively. The color coding corresponds to the spectrum shown in figure 4.8.

of the screech mode. These modes are also likely interactions between KH instabilities and shock cells but at higher wavenumbers, possibly corresponding to shock cells with smaller spacing, such as those further from the nozzle. Further analysis to categorize and understand these two modes is part of our future work, but it falls outside the scope of this thesis.

Next, we compute the resolvent modes for the same range of $St \in [0.2, 1]$ with a resolution of $\Delta St = 0.01$ for the same LNS operator. Unlike the gain distribution observed for a turbulent jet (see figure 2.11), we identify three distinct peaks at $St = 0.66$, 0.72, and 0.77, which aligns with the analysis of the eigenspectrum. As anticipated, the resolvent response modes closely resemble the eigenmode at the screech frequency. A comparison of the velocity components with POD data derived from experimental snapshots shows a strong agreement between the modes, as depicted in figure 4.11. This consistency not only validates the accuracy of our computations but also reinforces the use of the resolvent mode as the screech mode for subsequent analysis.

Following this, a secondary analysis was conducted using harmonic resolvent analysis. Unlike streaks in §4.1, the screech mode oscillates at a specific frequency. Accordingly, the modified mean flow $\bar{\boldsymbol{q}}_{mod}$ was constructed by superimposing the screech mode $\boldsymbol{q}_{res}$ onto the

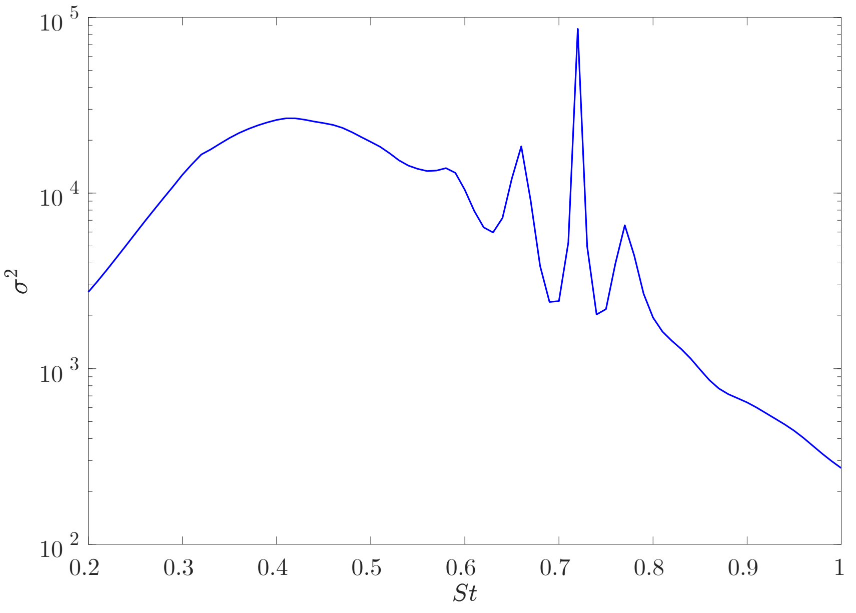


Figure 4.10: The optimal resolvent gain plot of the screeching jet.

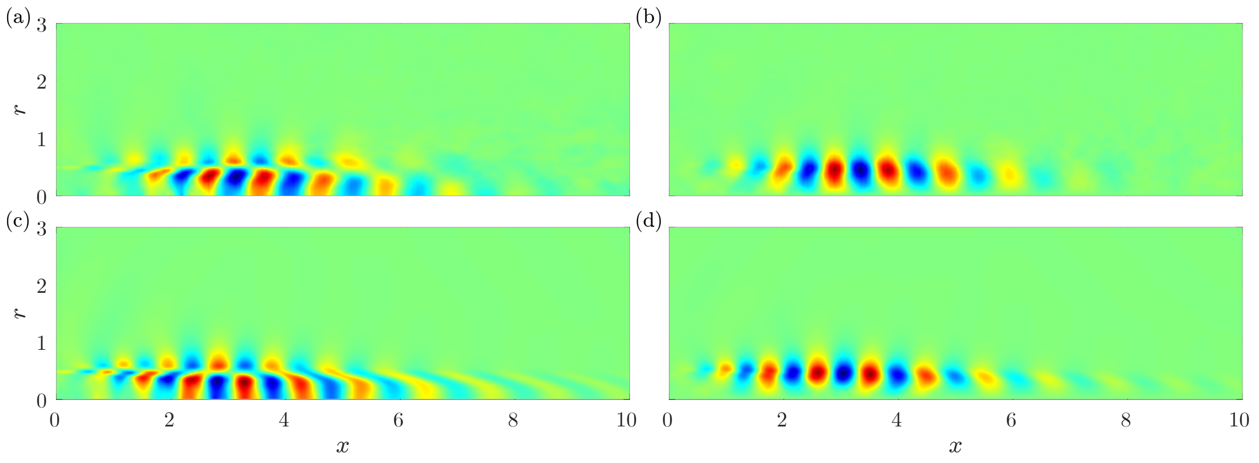


Figure 4.11: Screeching jet results: (a, b) POD modes from experimental data and (c, d) optimal resolvent response modes at $St = St_s$. Panels (a, c) show streamwise velocity, and panels (b, d) show radial velocity components.

original mean flow $\bar{\boldsymbol{q}}_{org}$, expressed as

$$\bar{\boldsymbol{q}}_{mod} = \bar{\boldsymbol{q}}_{org} + A_s \boldsymbol{q}_{res} e^{\mathrm{i} St_s 2\pi t}, \tag{4.3}$$

where $St_s = 0.72$ represents the screech frequency, and $A_s \approx 0.25$ denotes the relative amplitude with respect to the mean flow, determined through experimental matching. The time-periodic LNS operator is constructed around the modified mean flow. Given that the mean flow incorporates only a single non-zero mode at the screech frequency and the linearization process is first order, the resulting $\boldsymbol{T}$ matrix contains only the non-zero components $\hat{A}_0$ and $\hat{A}_{\pm 1}$. The input forcing is constrained to $r \leq 1$ excluding the sponge, and the output response domain excludes only the sponge regions.

The harmonic resolvent modes are computed using RSVD-Δt with parameters $k = 10$ and $q = 1$. The RK4 integration scheme is used with a time step of $dt = 0.01$ and a transient length of $T_t = 200$ for time-stepping. This integration period is sufficient to obtain accurate results, thus obviating the need for a transient removal strategy. However, if required, the transient length can be truncated to $T_t \approx 150$ without compromising accuracy. Given the real-valued nature of the LNS operator, $\hat{A}_0$ and $\hat{A}_1$ are retained in memory, and positive frequencies for the perturbation frequency range include $\Omega_q = \{0, St_s, 2St_s, 3St_s\}$. Results are obtained up to the 4^{th} harmonic, with convergence observed by the 3^{rd} harmonic. This indicates that the energy contribution beyond the third harmonic is negligible, and thus, higher harmonics do not influence the outcome. It is noted that while RSVD-LU could potentially offer similar computational efficiency, the associated memory requirements would be prohibitively large (exceeding 3.5 TB on the High Performance Computing (HPC) cluster), as extrapolated from data on smaller grid setups with the same frequency resolution.

A comparison is carried out by computing resolvent modes and gains at the corresponding frequencies. The LNS operator is derived by linearizing the compressible Navier-Stokes equations around the original mean flow, which is equivalent to $\hat{A}_0$. Given the moderate size of each block of the resolvent operator, the modes are computed using the Arnoldi method. The optimal gain values for $St = 0$ and $St = St_f$ are determined to be 293.6 and 74.81, respectively. For the second and third harmonics, the optimal gain values are significantly lower, at 5.33 and 4.05, respectively, indicating a decrease of two orders of magnitude compared to the maximum gain at $St = \pm St_f$. The corresponding resolvent modes are illustrated in figure 4.12. The optimal gain of the harmonic resolvent operator is 149.3, with the relative energy content distributed across the frequency range, as shown in figure 4.13(b). Note that the resolvent modes are confined to a single frequency, whereas the response modes from harmonic resolvent analysis are not; instead, they exhibit a broader

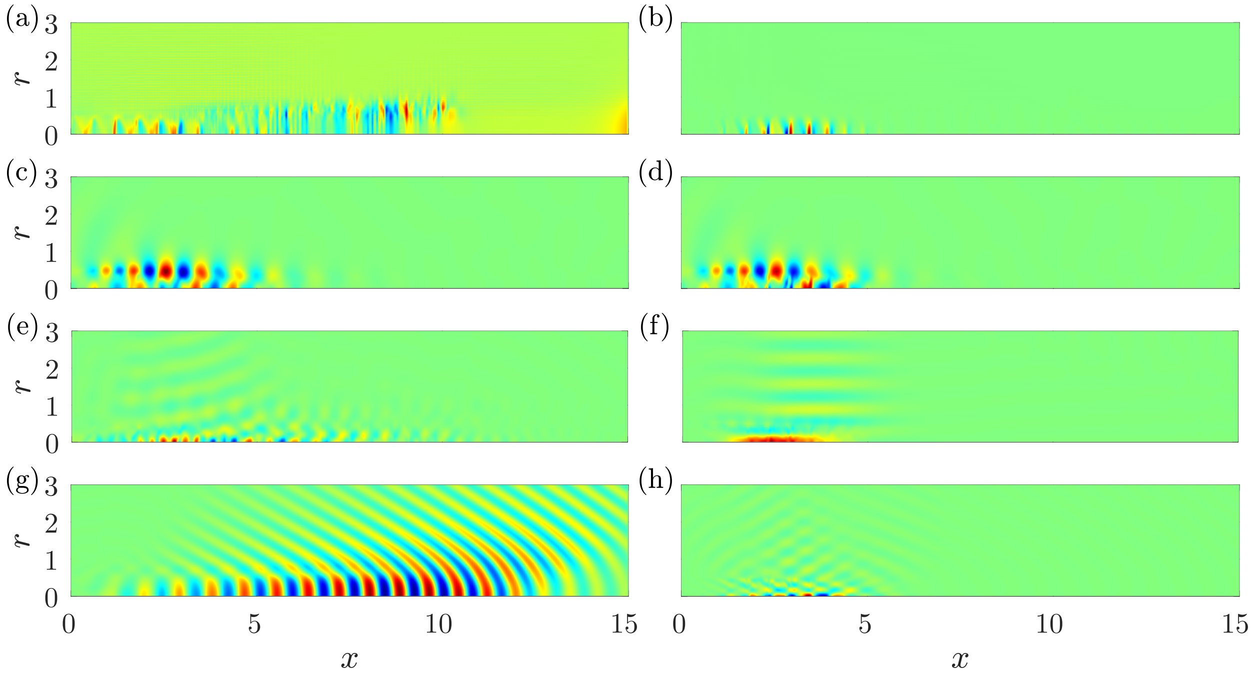


Figure 4.12: Screeching jet results: (a, c, e, g) optimal resolvent pressure response modes and (b, d, f, h) optimal harmonic pressure response modes at frequencies of $0, St_s, 2St_s$, and $3St_s$, respectively.

energy spectrum.

The optimal resolvent response at $St = 0$ exhibits some similarities to the optimal harmonic response mode at the same frequency, but it is more corrupted. The harmonic mode represents a modification to the shock cells, as the wavepackets are stationary. The modes at $St = St_s$ show the closest match, both containing a downstream-traveling KH wavepacket and an upstream-traveling guided jet wave. The modes at $St = 2St_s$ and $St = 3St_s$ are not well captured by resolvent analysis, suggesting that these modes arise from nonlinear interactions between the response modes and the screech mode, a topic that will be discussed further in the following sections.

From another perspective, to ensure consistency in comparison with the harmonic resolvent gains, the leading resolvent gains were recomputed after removing $\hat{A}_{\pm 1}$ from the $\boldsymbol{T}$ matrix. Figure 4.13(a) presents both the resolvent and harmonic resolvent gains for the six leading modes. The values for mode indices one, two, and three matched those reported from the individual blocks of the resolvent analysis, indicating that the three leading modes are associated with $St = \pm St_s$ and $St = 0$, respectively. A further analysis of the response energy spectrum confirmed these matching values and revealed that the first two resolvent gains correspond to $St = \pm St_s$, while the remaining gains are associated with $St = 0$. A key observation is the significant gap between the two leading resolvent gains and the harmonic

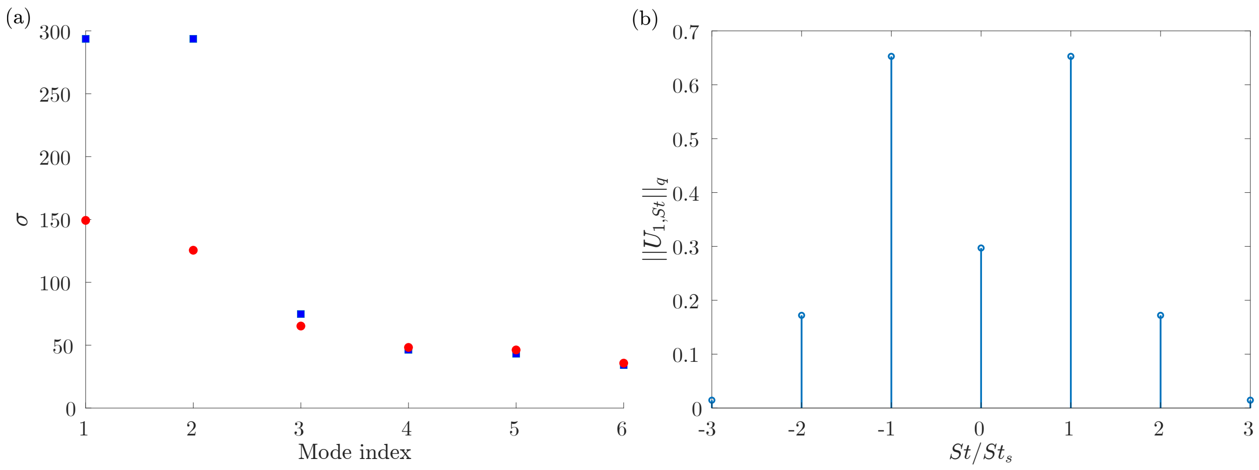


Figure 4.13: (a) The six leading gains using RSVD-Δt for both resolvent (blue) and harmonic resolvent (red) analyses of the screeching jet. (b) The energy spectrum of the optimal harmonic resolvent response mode of the screeching jet. The q-norm of the response (one column) is 1.

resolvent gains, nearly halving for the first two modes and to a lesser extent for the third mode. This discrepancy warrants further investigation, which will be explored in subsequent discussions.

To gain a deeper understanding of the amplification mechanisms, the process is categorized into two types: linear and nonlinear amplifications. The governing equation (3.7) can be rewritten as

$$(\boldsymbol{D} + \tilde{\boldsymbol{D}})\hat{\boldsymbol{q}} = \hat{\boldsymbol{f}}, \tag{4.4}$$

where $\boldsymbol{D}$ represents the diagonal components of the inverse of resolvent operators $\boldsymbol{L}_\omega$ for $\omega \in \Omega_q$, and $\tilde{\boldsymbol{D}}$ contains the off-diagonal terms of $\boldsymbol{T}$ such that $\boldsymbol{D} + \tilde{\boldsymbol{D}} = \boldsymbol{T}$. Moving the nonlinear terms to the right-hand side gives

$$\underbrace{\boldsymbol{D}}_{\text{Linear (resolvent) operators}} \hat{\boldsymbol{q}} = \underbrace{-\tilde{\boldsymbol{D}}\hat{\boldsymbol{q}}}_{\text{Nonlinear forcing}} + \underbrace{\hat{\boldsymbol{f}}}_{\text{Linear forcing}}. \tag{4.5}$$

or

$$\hat{\boldsymbol{q}} = \underbrace{-\boldsymbol{D}^{-1}\tilde{\boldsymbol{D}}\hat{\boldsymbol{q}}}_{\text{Nonlinear response}} + \underbrace{\boldsymbol{D}^{-1}\hat{\boldsymbol{f}}}_{\text{Linear response}} = \hat{\boldsymbol{q}}_{nl} + \hat{\boldsymbol{q}}_l. \tag{4.6}$$

This formulation indicates that the overall response $\hat{\boldsymbol{q}}$ is composed of both linear $\hat{\boldsymbol{q}}_l$ and nonlinear $\hat{\boldsymbol{q}}_{nl}$ components. This can be further decomposed as

$$\hat{\boldsymbol{q}}_k = \boldsymbol{D}_k^{-1}\left(\sum_{i+j=k} \tilde{\boldsymbol{D}}_i \hat{\boldsymbol{q}}_j\right) + \boldsymbol{D}^{-1}\hat{\boldsymbol{f}}_k, \tag{4.7}$$

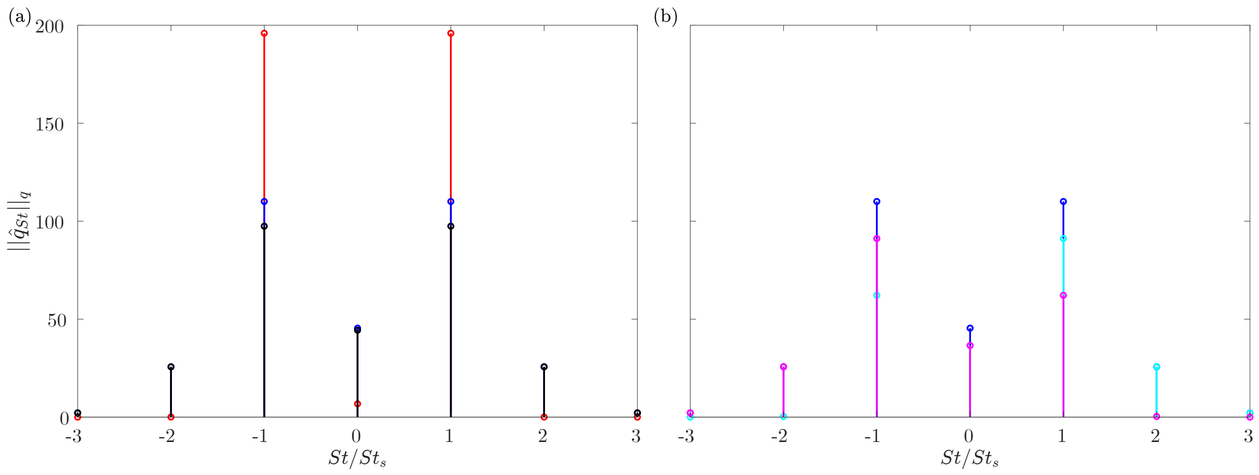


Figure 4.14: Screeching jet results: (a) the energy spectrum of $\boldsymbol{U}_1$ (black), $\boldsymbol{U}_{1,l}$ (red), and $\boldsymbol{U}_{1,nl}$ (blue), and (b) the energy spectrum of $\boldsymbol{U}_{1,nl}$ (blue), $\boldsymbol{U}_{1,nl,-1}$ (pink), and $\boldsymbol{U}_{1,nl,1}$ (cyan).

where the k^{th} harmonic response is determined by interactions between the i^{th} harmonic of the mean flow and the j^{th} harmonic of the response, such that $i + j = k$. In the case of the screeching jet, $i = \pm 1$ is the primary interaction, leading to

$$\hat{\boldsymbol{f}}_{nl} = \begin{bmatrix} \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{-2} \\ \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{-1} \\ \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{0} \\ \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{1} \\ \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{2} \\ \hat{\boldsymbol{A}}_{-1}\hat{\boldsymbol{q}}_{3} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{-3} \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{-2} \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{-1} \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{0} \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{1} \\ \hat{\boldsymbol{A}}_{1}\hat{\boldsymbol{q}}_{2} \end{bmatrix} = \hat{\boldsymbol{f}}_{nl,-1} + \hat{\boldsymbol{f}}_{nl,1}. \tag{4.8}$$

Both the optimal harmonic forcing $\boldsymbol{V}_1$ and response $\boldsymbol{U}_1$ modes are computed. By substituting $\hat{\boldsymbol{f}} = \boldsymbol{V}_1$ and $\hat{\boldsymbol{q}} = \boldsymbol{U}_1$, the components and interactions between various harmonics are analyzed. The action of $\boldsymbol{D}^{-1}$ on $\hat{\boldsymbol{V}}_{1,nl,-1}$, $\hat{\boldsymbol{V}}_{1,nl,1}$, and $\hat{\boldsymbol{V}}_{1,l}$ is computed using time-stepping with the same parameters to obtain $\hat{\boldsymbol{U}}_{1,nl,-1}$, $\hat{\boldsymbol{U}}_{1,nl,1}$, and $\hat{\boldsymbol{U}}_{1,l}$, respectively. This computation takes approximately three minutes on a cluster with 300 cores.

The energy spectrum of $\boldsymbol{U}_1$ is decomposed into its linear $\boldsymbol{U}_{1,l}$ and nonlinear $\boldsymbol{U}_{1,nl}$ components, as shown in figure 4.14(a). The linear forcing is primarily amplified at $St = \pm St_s$. However, the nonlinear response at $St = \pm St_s$ significantly reduces the optimal gain of the harmonic resolvent operator, effectively halving the amplification by canceling out the linear response. The energy spectrum of responses at other frequencies, including 0, $2St_s$, and $3St_s$, indicates that these modes are generated by nonlinear interactions. Further examination, breaking down $\boldsymbol{U}_{1,nl}$ into $\boldsymbol{U}_{1,nl,-1}$ and $\boldsymbol{U}_{1,nl,1}$, as illustrated in figure 4.14(b), reveals that the

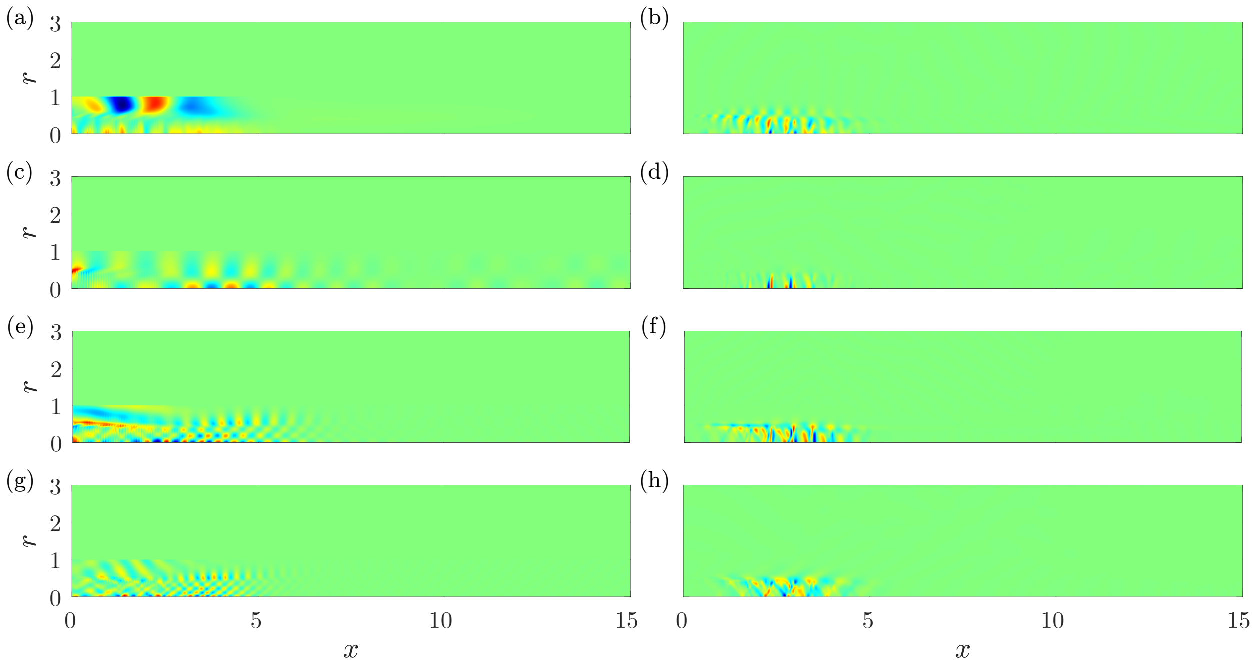


Figure 4.15: Screeching jet results: (a, c, e, g) $\boldsymbol{V}_{1,l}$ and (b, d, f, h) $\boldsymbol{V}_{1,nl}$ at frequencies of $0, St_s, 2St_s$, and $3St_s$, respectively.

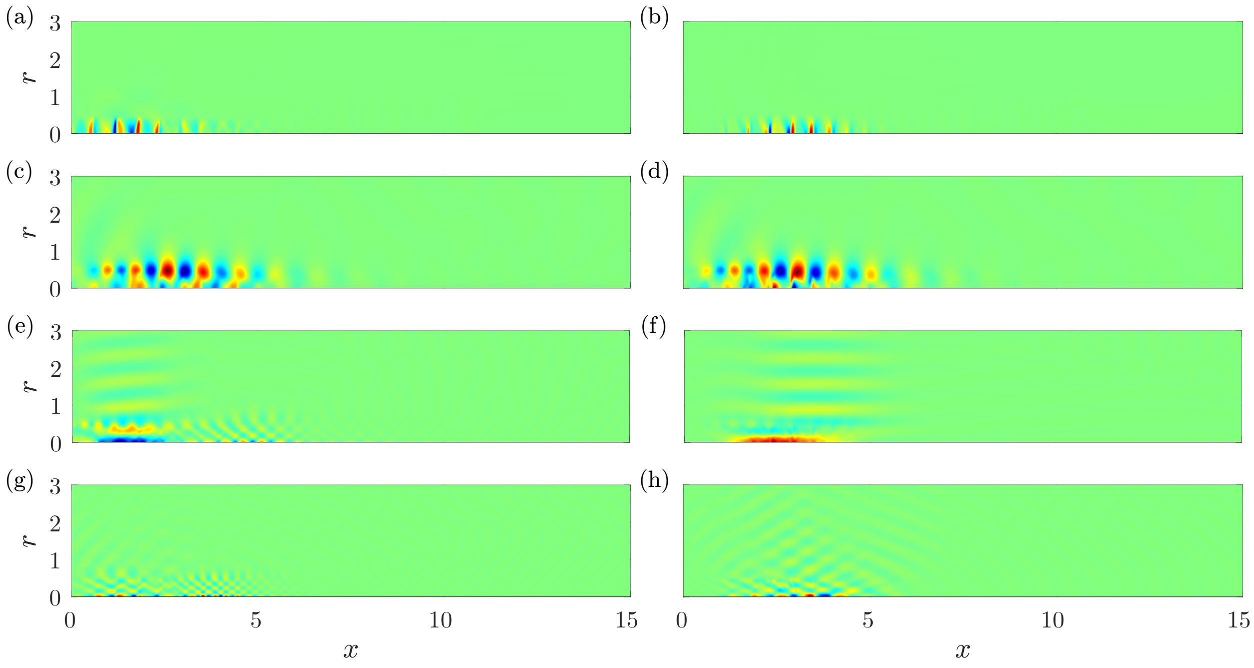


Figure 4.16: Screeching jet results: (a, c, e, g) $\boldsymbol{U}_{1,l}$ and (b, d, f, h) $\boldsymbol{U}_{1,nl}$ at frequencies of $0, St_s, 2St_s$, and $3St_s$, respectively.

response at the second harmonic arises primarily from the interaction between $\hat{\boldsymbol{A}}_1$ and $\hat{\boldsymbol{q}}_1$, while the interaction between $\hat{\boldsymbol{A}}_{-1}$ and $\hat{\boldsymbol{q}}_2$ has minimal effect. Similarly, the third harmonic response results from the interaction between $\hat{\boldsymbol{A}}_1$ and $\hat{\boldsymbol{q}}_2$, rather than $\hat{\boldsymbol{A}}_{-1}$ and $\hat{\boldsymbol{q}}_3$. For the negative frequencies, due to symmetry, interactions between $\hat{\boldsymbol{A}}_{-1}$ and $\hat{\boldsymbol{q}}_{-1}$ produce a response at $St = -2St_s$, while interactions between $\hat{\boldsymbol{A}}_{-1}$ and $\hat{\boldsymbol{q}}_{-2}$ generate the response at $St = -3St_s$.

The linear and nonlinear forcing modes are presented in figure 4.16. From the energy distribution at each frequency, we observe that the linear forcing at $St \in \{0, 2St_s, 3St_s\}$ is not significantly amplified through the linear operator $\boldsymbol{D}$. By comparing the nonlinear responses shown in figure 4.16 with the optimal harmonic response in figure 4.12, it becomes evident that at those three frequencies, only the nonlinear component is active, confirming that these response modes are generated by nonlinear interactions, which are not captured by resolvent analysis.

At $St = St_s$, both linear and nonlinear responses are active and nearly orthogonal to one another. The KH mode is generated in both cases, but they cancel each other out as they are in phase. The guided jet modes, however, differ. This results in a mode with reduced amplification, explaining why the amplification of the optimal harmonic resolvent mode is approximately half of the peak resolvent gain at $St = \pm St_s$. Interestingly, the forcing that generates very similar KH patterns differs; for the nonlinear forcing at $St = St_s$, the forcing is localized at the core, whereas the linear forcing is primarily located in the nozzle shear layer.

The nonlinear response and forcing modes corresponding to $\boldsymbol{U}_{1,nl,-1}$ and $\boldsymbol{U}_{1,nl,1}$, and $\boldsymbol{V}_{1,nl,-1}$ and $\boldsymbol{V}_{1,nl,1}$ are also displayed in figures 4.18 and 4.17, respectively. At all frequencies, the nonlinear forcing components are mostly localized in the core region, partly due to the presence of shock cells. The corresponding nonlinear response modes reveal which forcing terms are most amplified and which are less significant. This analysis clarifies the contributions of linear and nonlinear interactions to the amplification mechanisms and the generation of specific modes. The final aspect of this investigation focuses on the relationships between the wavenumbers associated with some of these coherent structures.

4.3.2.1 Wavenumber spectra analysis

The wavenumber analysis provides a means to elucidate the interactions between different wavepackets that contribute to the screeching phenomenon. The analysis begins with an appropriate normal mode ansatz, expressed as

$$\boldsymbol{q}(x, r, \theta, t) = \hat{\boldsymbol{q}}(r) e^{\mathrm{i}\omega t + \mathrm{i}m\theta + \mathrm{i}kx}, \tag{4.9}$$

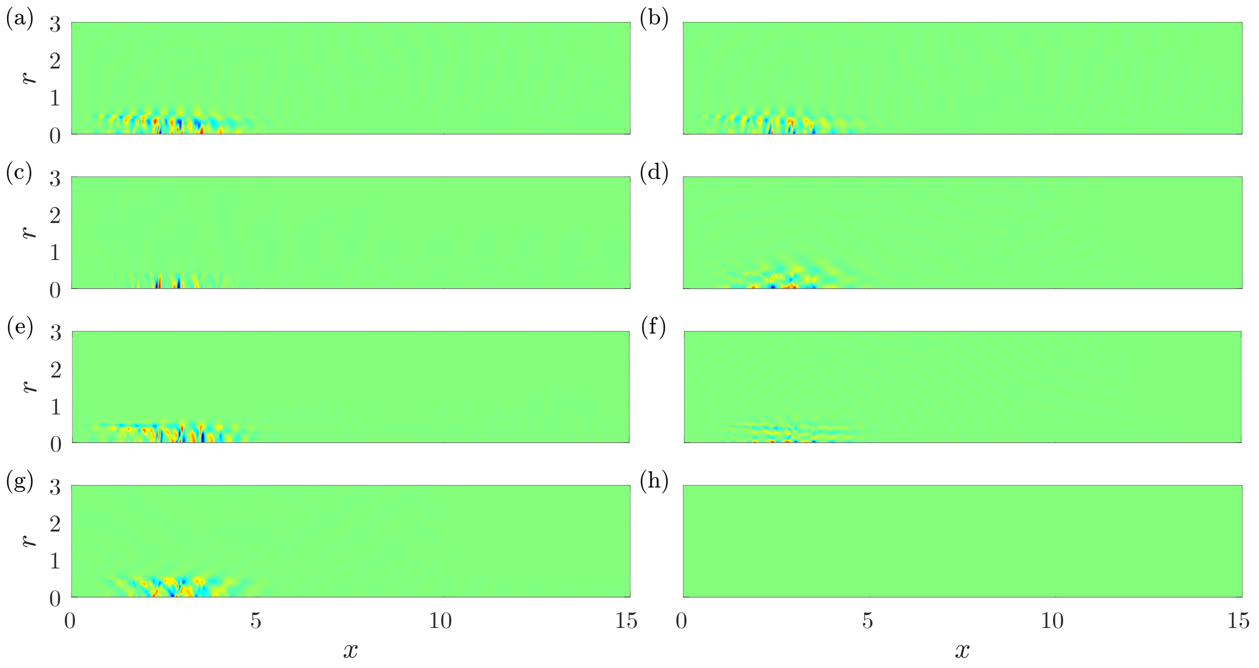


Figure 4.17: Screeching jet results: (a, c, e, g) $\boldsymbol{V}_{1,nl,1}$ and (b, d, f, h) $\boldsymbol{V}_{1,nl,-1}$ at frequencies of $0, St_s, 2St_s$, and $3St_s$, respectively.

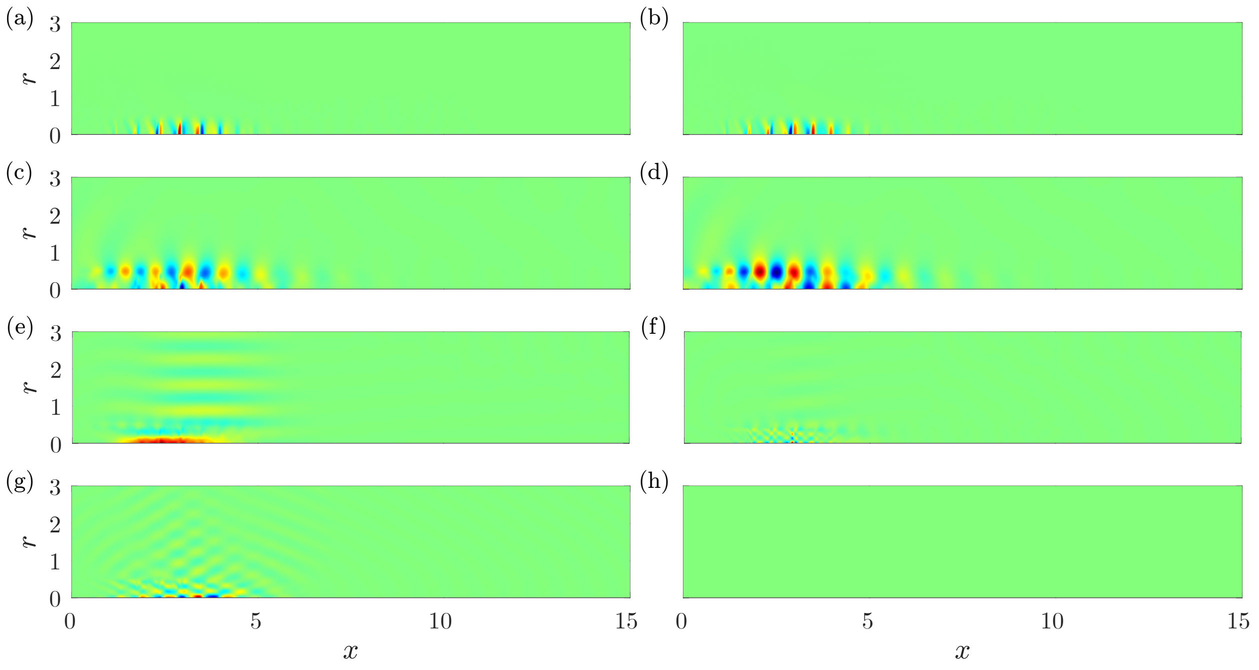


Figure 4.18: Screeching jet results: (a, c, e, g) $\boldsymbol{U}_{1,nl,1}$ and (b, d, f, h) $\boldsymbol{U}_{1,nl,-1}$ at frequencies of $0, St_s, 2St_s$, and $3St_s$, respectively.

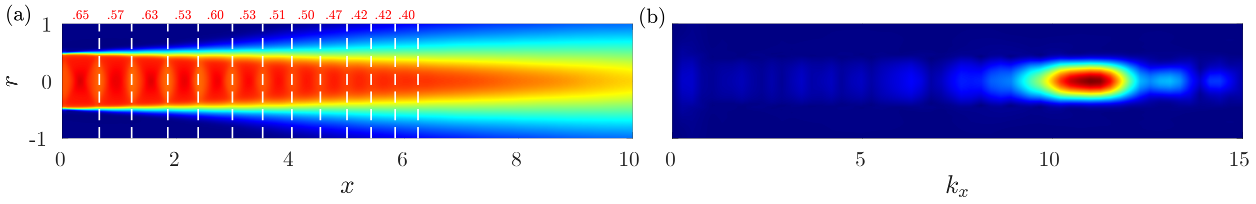


Figure 4.19: (a) Mean streamwise velocity with weak shock cells, with dashed lines indicating shock cell spacing. (b) Wavenumber spectrum of the mean streamwise velocity.

where k represents the streamwise wavenumber. This representation is valid under the local stability analysis framework, assuming parallel flow with minimal variation in the streamwise direction. However, instead of conducting a local stability analysis, a Fourier transform is applied in the streamwise direction across all $r \in [0, 3]$ to obtain the wavenumber spectra of the modes under investigation. As documented in [35], the relationship between the streamwise wavenumbers of the KH instability k_{kh}, the shock cells k_s, and the guided jet mode k_{gj} is given by

$$k_{gj} = k_{kh} - k_s. \tag{4.10}$$

This equation implies that the downstream-traveling KH mode interacts with shock cells, and the difference between the wavenumbers generates the upstream-traveling guided jet. Based on previous analyses of similar screeching jets, $k_{kh} = 11$ and $k_s = 7$ lead to $k_{gj} = -4$. The positive signs indicate downstream-traveling waves, while the negative sign for the guided jet suggests it is an upstream-traveling wave.

The computation of k_s is based on the mean streamwise velocity, as shown in figure 4.19. Due to the progressive decrease in the size of the shock cells, the wavelength is not a single value but encompasses a range; in this study, k_s is found to be 11 ± 0.5. To determine k_{kh} and k_{gj}, the wavenumber spectra of both streamwise velocity and pressure of the screech mode are plotted in figure 4.20. The guided jet is more prominent in the pressure wavenumber spectrum, occurring at $k_{gj} = -5 \pm 0.5$, while the KH wavenumber exhibits a broader range. The KH mode closer to the core region appears at $k_{kh} = 6$, as observed in the streamwise velocity wavenumber spectrum. The wavenumber range is within $k_{kh} = 7 \pm 1$ in the pressure field. This confirms the relationship between the wavenumber spectra of various modes as expressed in (4.10).

To explore the relationship between nonlinear interactions, the wavenumber spectra of the linear $\hat{\boldsymbol{V}}_{1,l}$ and nonlinear $\hat{\boldsymbol{V}}_{1,nl}$ optimal forcing modes were computed, as shown in figure 4.21. These modes produce responses that cancel each other on KH modes and modify the guided jet wavepacket, despite appearing different in shape. For the linear forcing, the mode is primarily localized at the nozzle exit, close to the shear layer at $r = 0.5$, and extends

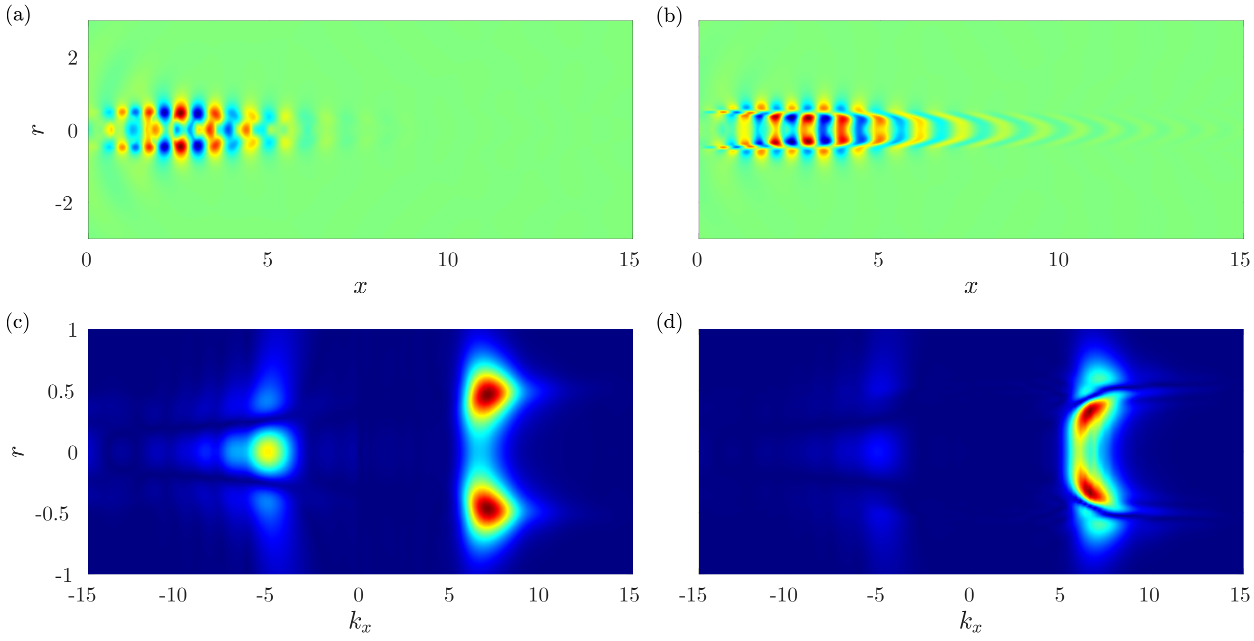


Figure 4.20: Pressure (a) and streamwise velocity (b) components of the screech mode, with corresponding wavenumber spectra: pressure (c) and streamwise velocity (d).

downstream to support the response throughout the entire region. In contrast, the nonlinear case shows the shear layer mode localized away from the nozzle, with the main wavepacket centered in the core region. The wavenumbers are spaced at intervals of approximately 11, indicating that the shock cells and their harmonics interact with the response to create this two-way traveling forcing.

Finally, the optimal harmonic response at $St = 0$ and $St = St_s$ and the associated wavenumber spectra are presented in figure 4.22. The wavenumbers for $St = 0$ are closely aligned with those found at $St = St_s$ for the forcing. The mode at this frequency is stationary and can be interpreted as a modification to the shock cells. The wavepacket comprises several peaks with consistent spacing of approximately 11, starting at $k_{mod} = 15$. At $St = St_s$, $k_{kh} \approx 7.5$, and the relationship is expressed as

$$k_{mod} \in 2k_{kh} + ck_s, c \in \mathbb{N}. \tag{4.11}$$

Collectively, the findings from these three jet configurations contribute to a deeper understanding of the intricate dynamics governing jet flows. The application of RSVD-Δt has proven to be effective and reliable in predicting patterns and extracting coherent structures results from resolvent and harmonic resolvent analyses.

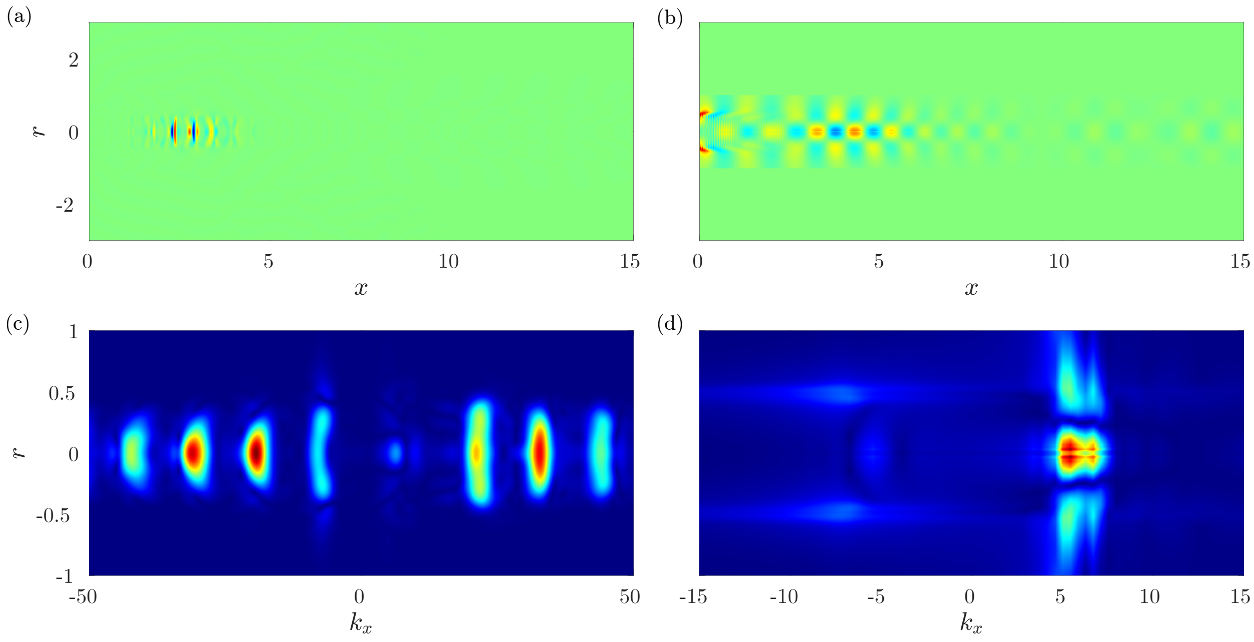


Figure 4.21: Pressure components of the optimal nonlinear forcing (a) and optimal linear forcing (b), along with their wavenumber spectra: nonlinear forcing (c) and linear forcing (d).

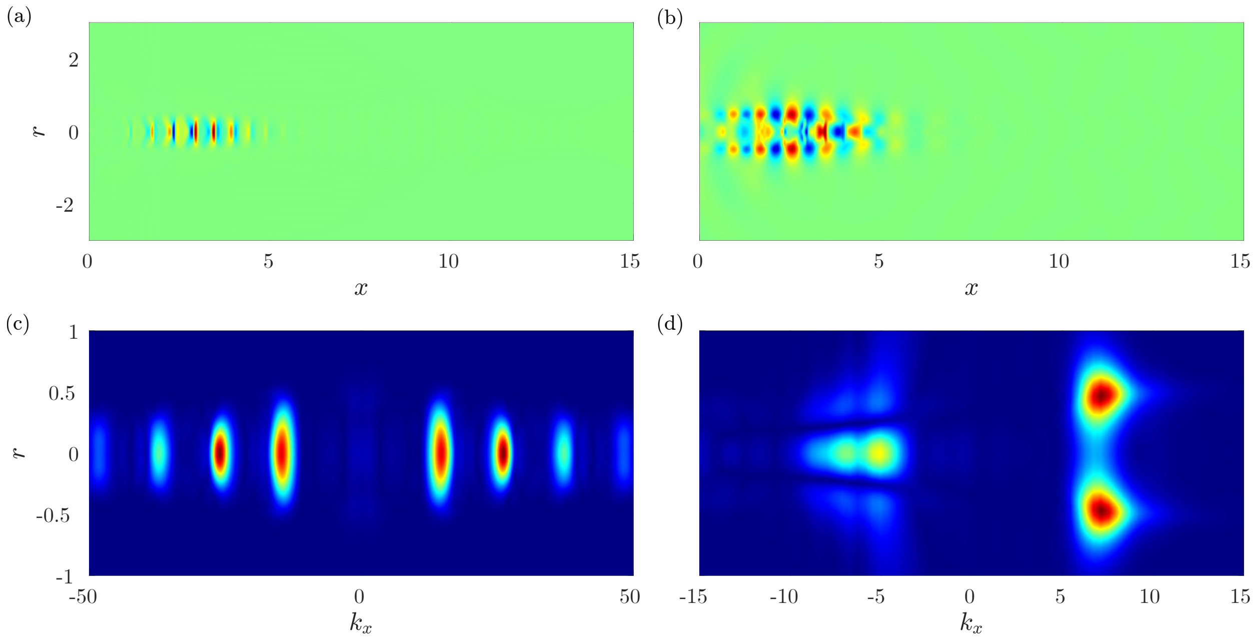


Figure 4.22: Pressure components of the optimal harmonic resolvent response at $St = 0$ (a) and $St = St_s$ (b), with wavenumber spectra at $St = 0$ (c) and $St = St_s$ (d).

CHAPTER 5

RSVD-Δt User Manual

As detailed in previous chapters, RSVD-Δt was developed to tackle the computational challenges associated with computing resolvent and harmonic resolvent modes for large-scale fluid flows, particularly when existing algorithms lack scalability. This cutting-edge tool combines the strengths of RSVD with an optimized time-stepping approach, delivering reliable resolvent modes with remarkable efficiency. Hence, RSVD-Δt enables researchers to apply resolvent and harmonic resolvent analyses to more complex and larger systems than ever before. Although our algorithm has been thoroughly explained in [45, 44, 43], the implementation of code within open-source libraries, with extensive parallelization, can be time-consuming. To facilitate this process, we share the open-source software package.

This chapter provides a comprehensive user manual for RSVD-Δt, detailing its features and operational procedures. By the end of this chapter, users will be equipped to install and utilize RSVD-Δt in their research, thereby advancing the study of large-scale flow dynamics. The open-source package, which includes source codes for both resolvent and harmonic resolvent analyses as well as a brief user manual with a tutorial, is available on GitHub under separate branches.

5.1 Key features of the RSVD-Δt implementation

RSVD-Δt offers several key features that make it a robust and efficient tool for resolvent and harmonic resolvent analyses in large-scale systems:

- **Linear CPU time scalability**: Computational complexity scales linearly with the number of discrete degrees of freedom, $O(N)$, which significantly reduces computational overhead for large systems compared to existing algorithms.
- **Linear memory usage scalability**: Memory usage also scales linearly with the number of discrete degrees of freedom, $O(N)$, and further minimizes memory usage through streaming Fourier sums, facilitating the study of high-dimensional systems.

- **Error control strategies**: The tool incorporates strategies such as power iteration and transient removal to manage errors and ensure the reliability of computed resolvent and harmonic resolvent modes.
- **User-friendly interface**: RSVD-Δt features an input list for specifying directories and parameters, as outlined in this chapter. Computations are managed by an executable that leverages parallel processing via PETSc and SLEPc libraries, allowing researchers to focus on analysis rather than computational complexities.

5.2 Getting started

To use RSVD-Δt, follow these steps:

1. **Install dependencies**:
   - OpenMPI (or another MPI package such as MPICH)
   - A C++ compiler
   - PETSc and SLEPc packages (the open-source package is developed using PETSc 3.19.4 and SLEPc 3.19)
2. **Download and extract** the RSVD-Δt package.
3. **Navigate** to the directory containing the extracted files.
4. **Build the executable** using the provided makefile.
5. **Run the executable** with the appropriate input parameters to compute resolvent modes.

The installation process for both resolvent and harmonic resolvent analyses follows the same procedure. However, the source codes and input variables vary due to the fundamental differences between the two types of analyses. In the following sections, we will guide you through these steps in detail.

5.3 Installing PETSc and SLEPc

Proper installation of the PETSc and SLEPc libraries is crucial for setting up our package. For detailed instructions, please refer to the official installation guides:

- PETSc official installation guide
- SLEPc official installation guide

5.3.1 PETSc

Starting with PETSc, once the package is downloaded and unzipped on your local machine or cluster, navigate to the unzipped PETSc directory. Ensure that all your dependencies (including MPI and C++ compiler) are accessible:

- On an HPC cluster, ensure that dependencies are accessible by using the appropriate commands in your Linux environment:

  ```
  module load <your C++ compiler>
  module load <your MPI package>
  ```

  For example, on the Great Lakes cluster at the University of Michigan, we use:

  ```
  module load gcc/10.3.0 openmpi/4.1.6
  ```

- On your laptop, when configuring PETSc, either navigate to your local MPI directory with:

  ```
  --with-mpi-dir=/usr/local/mpich
  ```

  or allow PETSc to download it with:

  ```
  --download-openmpi or --download-mpich
  ```

- If using a local MPI directory, ensure that BLAS/LAPACK are also configured with:

  ```
  --with-blaslapack-dir=/usr/local/blaslapack
  ```

- **IMPORTANT NOTE:** if MPI is not properly accessible or downloaded, PETSc will not use the MPI package, limiting the installation to a single core.

For additional details on configuring PETSc, please refer to the configuring PETSc page. Now, you can configure PETSc.

After navigating to your PETSc directory, you must configure your PETSc package. A recommended configuration for PETSc is as follows:

```
./configure --with-debugging=0 --with-scalar-type=complex
--with-64-bit-indices PETSC_ARCH=complex-opt
```

This configuration ensures that computations use complex values, as resolvent modes are inherently complex numbers. It also enables 64-bit integer numbers, allowing matrices to have sizes up to 2^{64}–1. While this increases memory usage by a factor of two compared to 32-bit integers, it is necessary for matrices larger than $2^{32}–1 = 2,147,483,647$, beyond which matrices would become unusable. In addition, performance is enhanced by disabling debugging, which is only needed when errors arise and their source cannot be pinpointed. Since our package has been thoroughly tested, debugging should not be necessary. The name of `PETSC_ARCH` is arbitrary, but `complex-opt` is used here as a default convention. You may include additional options as needed. For example:

```
./configure --with-debugging=0 --with-scalar-type=complex
--with-64-bit-indices PETSC_ARCH=complex-opt --download-openmpi
```

which downloads OpenMPI and installs PETSc on top of it. Here are a few notes on the installation process:

1. Follow PETSc's instructions to ensure a successful installation. The process usually takes less than 10 minutes.

2. The order of configuration options does not affect the installation.

3. You can have multiple PETSc installations with different `PETSC_ARCH` names. For instance, you might have one installation for real values with debugging enabled:

   ```
   ./configure --with-scalar-type=complex --with-debugging=1
   PETSC_ARCH=real-debug
   ```

4. While newer versions of PETSc and SLEPc generally retain the same core principles, syntax updates in the source code may be necessary.

5. The `PETSC_ARCH=<PETSc-arch-name>` can be customized as needed.

6. If additional packages are desired, include `--download-<package>` in the configuration command. For instance, `--download-mumps` will include MUMPS in your architecture. Ensure that the makefile is updated accordingly.

Before proceeding with the SLEPc installation, ensure that you have defined `PETSC_DIR` and `PETSC_ARCH` as follows:

```
export PETSC_DIR=/path/to/petsc
export PETSC_ARCH=<PETSc-arch-name>
```

5.3.2 SLEPc

SLEPc, the Scalable Library for Eigenvalue Problem Computations, is a powerful tool designed for solving large-scale eigenvalue and SVD problems. Built on PETSc, SLEPc provides scalable and efficient algorithms, making it ideal for high-performance scientific computing.

In our implementation, SLEPc is crucial for performing QR decomposition and SVD. Its advanced capabilities enable efficient handling of these operations, even with large matrices, ensuring our code operates effectively and accurately.

After navigating to your SLEPc directory and setting up the variables `PETSC_DIR` and `PETSC_ARCH`, configure the SLEPc package using the `./configure` command. After configuration, wait for the installation to complete, follow the provided instructions, and ensure that all tests pass successfully. Note that you need to use the same C++ compiler and MPI module for proper installation.

5.3.3 Setting up environment variables for PETSc and SLEPc

After you have completed the installation of PETSc and SLEPc, you need to define the environment variables `PETSC_DIR` and `SLEPC_DIR` to point to the installation directories of these packages. This is necessary for the proper functioning of your applications that rely on PETSc and SLEPc.

5.3.3.1 Steps to define environment variables

- **Identify the installation paths**: Determine the directories where PETSc and SLEPc are installed. For example, if PETSc is installed in `/path/to/PETSC` and SLEPc is installed in `/path/to/SLEPC`, you will use these paths in the following steps.
- **Export the environment variables:** Open a terminal and export the environment variables by running the following commands:

  ```
  export PETSC_DIR=/path/to/PETSC
  export SLEPC_DIR=/path/to/SLEPC
  ```

- **Persist the environment variables:** To avoid setting these variables every time you open a new terminal, add the export commands to your shell configuration file. If you are using a Linux environment with Bash, you can add these lines to your `/.bashrc` file:

```
echo 'export PETSC_DIR=/path/to/PETSC' >> ~/.bashrc
echo 'export SLEPC_DIR=/path/to/SLEPC' >> ~/.bashrc
```

After adding these lines, apply the changes by running:

```
source ~/.bashrc
```

For other environments (*e.g.*, Windows or macOS), you can add the equivalent commands to your respective shell configuration files (*e.g.*, `.bash_profile`, `.zshrc`).

By setting these environment variables, you ensure that the paths to PETSc and SLEPc are correctly defined, allowing you to compile and run your applications without needing to manually set the paths each time.

This step completes the installation and provides access to the required libraries needed to build our executable. Next, we will explain how to create the executable and provide an example of a job file.

5.4 Makefile usage

The `makefile` is used to build the executable from the source code. First, navigate to the directory of the RSVD-Δt package, where you will find the `makefile`, `inputs.yaml`, and a `SourceCodes` folder. The `inputs.yaml` file serves as the interface for defining all parameters before computing the modes. Note that the input list for harmonic resolvent analysis differs from that for resolvent analysis. The `SourceCodes` directory contains all the functions implemented for our algorithm. These files do not need to be modified unless you wish to contribute or add new features. Do not alter the relative path between the `makefile` and the `SourceCodes` folder, as this will prevent the executable from being created.

You can now use the following commands:
`make` to build the executable and `make clean` to remove the executable

If your `PETSC_ARCH` name differs from `complex-opt`, specify it in the `make` command as follows:

```
make PETSC_ARCH=<PETSc-arch-name>
```

Note: The default `make` command assumes `PETSC_ARCH=complex-opt`. If using a different name, you must include it in the command.

This step creates the executable, allowing you to run the RSVD-Δt algorithm on your local machine or HPC cluster. You will now find the `RSVDt` executable ready to use in the same directory.

5.5 Example jobfile

If you are running on your local machine, first navigate to the directory containing the executable, then you can run the job using the following command:

```
mpiexec RSVDt -inputs variables.yaml
```

To illustrate the usage of the RSVD-Δt algorithm on a HPC cluster, we provide an example jobfile in 5.1.

Listing 5.1: Example jobfile script on a cluster

```
#!/bin/bash
#SBATCH --job-name=<name>
#SBATCH --nodes=<count>
#SBATCH --ntasks-per-node=<count>
#SBATCH --mem=<memory>
#SBATCH --time=<dd-hh:mm:ss>

module load gcc/<version> openmpi/<version>

cd /path/to/executable/
make PETSC\_ARCH=complex-opt

mpiexec RSVDt -inputs variables.yaml
```

Note that you may need to use `srun` or `mpirun` instead of `mpiexec` depending on your installation and cluster configurations. Moreover, `make PETSC_ARCH=complex-opt` is required only once; it compiles the source files to create the executable or does nothing if the executable is already compiled. Finally, you might encounter slight differences in defining the number of nodes, tasks, CPUs, memory, etc., based on your cluster specifications. This jobfile serves as a sample case.

The installation of PETSc and SLEPc packages is required only once. To perform resolvent analysis, download the package from the `resolvent-analysis` branch on our GitHub page and compile it to generate the executable. For harmonic resolvent analysis, use the package available in the `harmonic-resolvent-analysis` branch on the same GitHub page. Please note that the source codes and input variables differ between these two types of analyses.

In the following sections, we provide detailed explanations of the parameters for each type of analysis. We also thoroughly describe the procedure using a Ginzburg-Landau system, which we used to validate our algorithm.

5.6 RSVD-Δt tutorial for resolvent analysis

In this tutorial, we will walk through the process of using the RSVD-Δt algorithm on a Ginzburg-Landau test case. We showcase both the transient run (the pre-processing step) and the main algorithm (for computing resolvent modes). We also describe all related input variables and the logistics of I/O directories. The linearized operator is provided in `Tutorial` folder under `resolvent-analysis` branch on our GitHub page. This operator is the one we used in 2.10.1.

5.6.1 List of input variables

Here is a list of variables used for this test case. Please note that you will need to modify these variables to suit your own projects.

Listing 5.2: Example input variables for RSVD-Δt

```
# Root directory (string)
# This directory must exist before the simulation
RootDir:            /path/to/root/directory/

# Results directory (string)
# The resolvent modes/gains will be saved in RootDir/ResultsDir/
   ResolventModes_<int>
# This directory must exist before the simulation
ResultsDir:         /path/to/results/

# The linearized operator (string)
# The operator directory is defined as RootDir/OperatorDir
# This directory must exist before the simulation, OperatorDir can be same
    as ResultsDir
OperatorDir:        /path/to/A # expected a matrix saved in binary format

# Number of test vectors (integer)
k:                  10
```

```

# Number of power iterations (integer)
q:                    1

# Number of frequencies to resolve (integer - even number)
Nw:                   42

# Base frequency (real)
# The resolvent modes will be computed at freuqency range (-Nw/2-1:1:Nw/2)
    *w
w:                    0.05

# Convert frequencies to angular frequencies (boolean)
# if true: w <-- 2*pi*w, otherwise, w <-- w
TwoPI:                false

# Transient length (real)
TransientLength:      10

# Time step (real)
dt:                   0.003

# Transient removal flag (boolean)
TransientRemoval:     true

# Display options (integer 0 <= Display <= 2)
Display:              2

# Discounting flag (boolean)
# Applies discounting for unstable linear systems, A <-- A - beta I
DiscFlg:              false

# beta value when "DiscFlg = True", otherwise beta is ignored (real > 0)
beta:                 0.1

# Seeding random number to replicate data if needed (integer)
RandSeed:             14

# Saving resolvent modes options (integer 1 <= SaveResultsOpt <= 2)
SaveResultsOpt:       1

# Input forcing flag (boolean)
# The input forcing directory must exist before the simulation if true,
    otherwise ignored
InputForcingFlg:    false
# Input forcing directory when "InputForcing = True" (string)
# This directory is defined as RootDir/InputForcingDir
InputForcingDir:    /path/to/F_hat # expected a matrix saved in binary
    format, size must match N x (k x Nw)

# Inverse input weight flag (boolean)
# The inverse input weight directory must exist before the simulation if
    true, otherwise ignored
InvInputWeightFlg:  false
# Inverse input weight directory when "InvInputWeightFlg = True" (string)
# This directory is defined as RootDir/InvInputWeightDir
InvInputWeightDir:  /path/to/W_f_sqrt_inv # expected a matrix saved in
    binary format

# Output weight flag (boolean)
# The input weight directory must exist before the simulation if true,
    otherwise ignored
OutputWeightFlg:    false
# Output weight directory when when "OutputWeightFlg = True" (string)
# This directory is defined as RootDir/OutputWeightDir
OutputWeightDir:    /path/to/W_q_sqrt # expected a matrix saved in binary
    format

# Inverse output weight flag (boolean)
# The inverse input weight directory must exist before the simulation if
    true, otherwise ignored
InvOutputWeightFlg: false
# Inverse output weight directory when "InvOutputWeightFlg = True" (string
    )
# This directory is defined as RootDir/InvOutputWeightDir
InvOutputWeightDir: /path/to/W_q_sqrt_inv # expected a matrix saved in
    binary format

# Input matrix flag (boolean)
# The input matrix directory must exist before the simulation if true,
    otherwise ignored
InputMatrixFlg:     false
# Input matrix directory when when "InputMatrixFlg = True" (string)
# This directory is defined as RootDir/InputMatrixDir
InputMatrixDir:     /path/to/B # expected a matrix saved in binary format

```

```
# Output matrix flag (boolean)
# The output matrix directory must exist before the simulation if true,
    otherwise ignored
OutputMatrixFlg:    false
# Output matrix directory when when "OutputMatrixFlg = True" (string)
# This directory is defined as RootDir/OutputMatrixDir
OutputMatrixDir:    /path/to/C # expected a matrix saved in binary format

# Runs the transient simulation if true, otherwise RSVD-delta-t will run (
    boolean)
TransRun:           false

# All variables below this are relevant when TransRun is true

# Number of periods to integrate (integer)
TransPeriods:       3

# Saves the transient outputs if true (boolean)
# The snapshots will be saved in RootDir/ResultsDir/TransientSnapshots_<
    int>
TransSave:          false

# Saves the snapshots every "TransSaveMod" number (integer)
TransSaveMod:       500

# Estimates the transient error using our strategy if true (boolean)
TransRemovalEst:    false

# Divergence value, the transient simulation will stop when reached (real)
TransDivVal:        1e3

# Convergence value, the transient simulation will stop when reached (real
    )
TransConVal:        1e-6

# Initial condition flag (boolean)
# The initial condition directory must exist before the simulation if true
    , otherwise ignored
TransICFlg:         false
# Initial condition directory when "TransICFlg = True" (string)
# This directory is defined as RootDir/TransICDir
TransICDir:         /path/to/ICvec # expected a vector saved in binary
    format
```

Important notes:

- For boolean flags, the following values are equivalent:
  - `False`, `false`, and `0` all represent `false`
  - `True`, `true`, and `1` all represent `true`
- Mathematical expressions in inputs are not allowed, *e.g.*, 2×2, $\sqrt{3}$, π will generate errors.
- For integer values, ensure only integers are provided. Decimal values such as 2.5, 1., or 3.14 will cause errors.
- When using discounting, ensure `beta` is positive.
- Error messages will be displayed, and the simulation will terminate if an error occurs.

5.6.2 Running the RSVD-Δt algorithm executable

5.6.2.1 Prerequisite modules

Make sure you have the following prerequisite modules loaded/installed:

- OpenMPI (or another MPI package such as MPICH)
- A C++ compiler
- PETSc and SLEPc packages

Note that the same versions of C++ compiler and MPI module are used to compile PETSc and SLEPc as those used to build the executable. Please refer to 5.3 for instructions on installing PETSc and SLEPc.

The process of running the RSVD-Δt algorithm can be divided into two main parts: the transient simulation and the RSVD-Δt algorithm. Before defining the variables specific to each part, we will first cover the variables that are common to both.

5.6.2.2 Common variables between transient and RSVD-Δt

- `RootDir`: Specifies the root directory path for the simulation.
- `ResultsDir`: Defines the path to the results directory where output files will be saved. This directory must exist within `RootDir`. If it does not, the system will create the directory at the specified path. Ensure you have write access to the root directory, or an error message will be displayed.

- `OperatorDir`: Specifies the directory path for the linearized operator matrix. If the operator is located in the `RootDir`, you only need to provide the operator name (*e.g.*, `A_GL`). Otherwise, specify the relative path to the operator from `RootDir` (*e.g.*, `matrices/A_GL`).
- `RandSeed`: Indicates the seeding number for random number generation. Using the same number of cores and `RandSeed` value allows for repeatable results in simulations.
- `DiscFlg` is a boolean flag indicating whether to use a discounting strategy.
- `beta` specifies the beta value when `DiscFlg = true`. It is ignored if `DiscFlg = false`.

5.6.2.3 Transient simulation

Before running the actual RSVD-Δt algorithm, it is often necessary to perform a transient simulation to ensure the stability of the system and visualize the transient decay rate.

To run the transient simulation, set the `TransRun` flag to `true` in the input file (`variables.yaml`).

5.6.2.4 Transient variables

- `TransPeriods` determines the length of transient simulation. We define the period length as $T = 2\pi/\omega_{min}$, where ω_{min} (defined as `w`) is the base frequency.
- `TransRemovalEst`, if `true`, applies the transient removal strategy we developed for slowly decaying systems (see 2.8.2 for details). It estimates the updated transient residual at the end of each period.
- `TransDivVal` and `TransConVal` are divergence and convergence values, respectively, which stop the simulation if the transient norm reaches either of them.

5.6.2.5 Saving transient results

- A folder is created in the results directory with the prefix `TransientSnapshots_<int>`, where `<int>` is an integer starting from 0. If `TransientSnapshots_i` already exists, the code increments the integer until a unique folder name is found, ensuring that results from different simulations are not overwritten.
- If `TransSave` is `true`, snapshots are saved as `q_transient_<int>` every `TransSaveMod` time steps in the results directory. In addition, the norm of the snapshots

is saved in a vector `q_transient_norms`, and the last snapshot is saved as `q_transient_last_snapshot`.

- If `TransSave` is `false`, only `q_transient_norms` and `q_transient_last_snapshot` are saved.
- If `TransRemovalEst` is `true`, the simulation saves the initial and updated transient norms to `Initial_transient_norm_period_<int>` and `Updated_transient_norm_period_<int>`, respectively, at the end of each period. For instance, `Initial_transient_norm_period_1` and `Updated_transient_norm_period_1` contain the norm of snapshots across the frequency range at the end of the first period.
- Note that the order of frequencies is as follows: column 1 corresponds to frequency 0, column 2 to frequency ω, and so on up to frequency $\frac{N_\omega}{2} \times \omega$. After this, the frequencies continue from $-\left(\frac{N_\omega}{2}+1\right) \times \omega$ up to the last column, which represents the $-\omega$ frequency. This ordering is similar to MATLAB's frequency arrangement.

5.6.2.6 Default values

If a transient variable is not specified or is commented out (using `#` before it), a warning message will be displayed, and the default value will be used instead. The default values are as following:

1. `TransRun`: `false`
2. `TransPeriods`: `1`
3. `TransSave`: `false`
4. `TransRemovalEst`: `false`
5. `TransDivVal`: `1e3`
6. `TransConVal`: `1e-6`
7. `RandSeed`: `1373`

For consistency and to avoid confusion, all variables starting with `Trans` are used exclusively in the transient part of the analysis.

5.6.2.7 RSVD-Δt algorithm

Setting `TransRun = false` runs the RSVD-Δt algorithm by default.

5.6.2.8 RSVD-Δt variables

- `k` indicates the number of test vectors.
- `q` is the number of power iterations.
- `Nw` represents the number of frequencies to resolve.
- `w` specifies the base frequency.
- `TwoPI` indicates whether to convert frequencies to angular frequencies. If true, the output frequencies are converted to angular frequencies by multiplying with 2π.
- `TransientLength` defines the length of the transient simulation.
- `dt` is the time step for transient simulations.
- `TransientRemoval` determines whether the transient removal strategy is used.
- `Display` controls the amount of information printed during the computation. It ranges from 0 (no output) to 2 (verbose output).
  - `Display = 0`: Minimal output, with no information displayed.
  - `Display = 1`: Standard output, displaying:
    * Problem information
    * Elapsed time for each test vector
    * Total elapsed time
    * Estimated remaining time
  - `Display = 2`: Detailed output, including everything from `Display = 1`, plus progress percentage of the first test vector
- `SaveResultsOpt` is an integer option that controls the saving results format, with two possible values: 1 and 2.
  - `SaveResultsOpt = 1`: Saves resolvent modes as `k` matrices of size `N` $\times$ `Nw`.
  - `SaveResultsOpt = 2`: Saves resolvent modes as `Nw` matrices of size `N` $\times$ `k`.

5.6.2.9 Saving resolvent modes

- We create a folder in the results directory with a fixed prefix `ResolventModes_<int>`, where `<int>` is an integer starting from 0. If `ResolventModes_i` exists, the code increments the integer until the folder name is unique, ensuring that results from different simulations are not overwritten.
- Once the computation is complete, two saving formats are available:
  - Option 1 when `SaveResultsOpt = 1`:
    - `k` response modes (each of size `N` × `Nw`) are saved as `U_hat_k<int>_allFreqs`, where `<int>` represents the integer index of the mode.
    - Similarly, forcing modes are saved as `V_hat_k<int>_allFreqs`.
    - For instance, `U_hat_k1_allFreqs` and `V_hat_k1_allFreqs` contain the optimal response and forcing modes, respectively, across all frequencies of interest.
    - Note that the order of columns corresponds to the optimality of the test vectors: column 1 contains the optimal mode, column 2 contains the first suboptimal mode, and so on.
  - Option 2 when `SaveResultsOpt = 2`:
    - `Nw` response modes (each of size `N` × `k`) are saved as `U_hat_Freq<int>_allK`, where `<int>` represents the integer index of the frequency.
    - Similarly, forcing modes are saved as `V_hat_Freq<int>_allK`.
    - For instance, `U_hat_Freq1_allK` and `V_hat_Freq1_allK` contain the response and forcing modes, respectively, associated with the first frequency.
    - Note that the order of frequencies starts with column 1 (frequency 0), column 2 (frequency ω), up to frequency $\frac{N_\omega}{2}\times\omega$, and then from $-\left(\frac{N_\omega}{2}+1\right)\times\omega$ up to the last column that contains the $-\omega$ frequency (similar to MATLAB ordering).
- Finally, gains are saved as a single matrix `S_hat` of size `k` × `Nw` in either case.

Important note: Not all variables have default values. If a variable is not specified, you will receive a warning or error message.

5.6.2.10 Default values for some variables

The following default values are used if a variable is not specified in the input list:

1. `k`: `3`

2. `q`: `0`
3. `TwoPI`: `false`
4. `TransientRemoval`: `false`
5. `Display`: `2`
6. `DiscFlg`: `false`
7. `Randseed`: `1373`

5.6.2.11 Required variables

The following variables must be specified as they have no default values:

- `RootDir`
- `ResultsDir`
- `OperatorDir`
- `TransientLength`
- `w`
- `Nw`
- `dt`
- `beta` (only when `DiscFlg` is `true`)

5.6.3 Transfer data between MATLAB and PETSc/SLEPc binary format

You may frequently need to transfer data between PETSc/SLEPc and MATLAB for post-processing results. This section will guide you through the process of converting data formats to ensure compatibility and seamless integration between MATLAB and PETSc/SLEPc.

5.6.3.1 Converting data from MATLAB to PETSc/SLEPc

MATLAB can save data in various formats, but for use with PETSc/SLEPc, we need to ensure the data is saved in a binary format compatible with these libraries. PETSc provides MATLAB functions to facilitate this conversion.

1. **Save data in MATLAB:**

   First, ensure you add the PETSc MATLAB interface to your MATLAB path:

   ```
   addpath('/path/to/PETSc/share/petsc/matlab/');
   ```

   Now, you may save your matrix (.mat file) to binary format using PETSc's binary write function. We have provided `A_GL` in both binary and .mat formats which you can test.

   ```
   PetscBinaryWrite('/path/to/your/matrix/A', ...
   A_GL, 'complex', true, 'indices', 'int64');
   ```

   `A_GL` is the variable name in MATLAB (.mat file). `A` is a sample name and can be replaced by any `<NAME_OF_YOUR_MATRIX>` for the binary saved file. Depending on the PETSc architecture you have compiled, `'complex', true` and `'indices', 'int64'` may vary. Please refer to the `PetscBinaryWrite` function for more information.

2. **Load data in PETSc:**

   In your PETSc/SLEPc environment, you can now load the binary file:

   ```
   Mat A;
   PetscViewer viewer;

   PetscViewerBinaryOpen(PETSC_COMM_WORLD, "/path/to/your/matrix/A",
   FILE_MODE_READ, &viewer);
   MatCreate(PETSC_COMM_WORLD, &A);
   MatSetType(A, MATMPIAIJ);
   MatLoad(A, viewer);
   PetscViewerDestroy(&viewer);
   ```

We assumed a sparse matrix is loaded on multiple cores. In case a single core is used, you can use `MATSEQAIJ`. If your matrix is dense, you may use `MATMPIDENSE`. Please refer to PETSc matrices page for more information.

5.6.3.2 Converting data from PETSc/SLEPc to MATLAB

To transfer data from PETSc/SLEPc to MATLAB, follow these steps:

1. **Save data in PETSc/SLEPc:**

   Save the matrix data in PETSc's binary format:

   ```
   PetscViewer viewer;

   PetscViewerBinaryOpen(PETSC_COMM_WORLD, "/path/to/your/matrix/A",
    FILE_MODE_WRITE, &viewer);
   MatView(A, viewer);
   PetscViewerDestroy(&viewer);
   ```

2. **Load data in MATLAB:**

   Read the PETSc binary file in MATLAB:

   ```
   addpath('/path/to/PETSc/share/petsc/matlab/');
   A = PetscBinaryRead('/path/to/your/matrix/A', ...
   'complex', true, 'indices', 'int64');
   ```

You do not need to be concerned with coding in the PETSc environment. Your primary task is to save your operator in binary format (from `.mat` to `.bin`) and to read your data from binary format into MATLAB (from `.bin` to `.mat`). For completeness, we have provided explanations for both ways.

In this tutorial, we covered the setup and execution of the RSVD-Δt algorithm for computing resolvent modes of the Ginzburg-Landau system. We discussed input variables, the process of running the algorithm, and how to save results. For your specific problems, you may need to adjust the input variables accordingly.

CHAPTER 6

Conclusion

This chapter presents a summary of the key findings and outlines directions for future research.

6.1 Summary

This thesis introduces RSVD-Δt, a novel algorithm designed for efficient computation of global resolvent modes in high-dimensional systems, particularly in the context of three-dimensional flows. By leveraging a time-stepping approach, RSVD-Δt eliminates the reliance on LU decomposition that often hampers the scalability of current state-of-the-art algorithms. As a result, RSVD-Δt not only enhances scalability but also extends the applicability of resolvent analysis to three-dimensional systems, overcoming previous computational limitations.

Scalability is of utmost importance for algorithms dealing with high-dimensional flows, and RSVD-Δt excels in this regard. In contrast, the decomposition of $(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})$ into lower and upper matrices poses a significant computational challenge for the RSVD-LU algorithm, limiting its scalability with $O(N^2)$ scaling for 3D problems. The CPU demand of RSVD-Δt, on the other hand, exhibits linear proportionality to the state dimension.

In addition to CPU considerations, memory requirements play a crucial role in computing resolvent modes for large systems. The LU decomposition of $(\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})$ is the primary contributor to peak memory usage in the RSVD-LU and other common algorithms. In contrast, the RSVD-Δt algorithm primarily utilizes RAM to store input and output matrices in Fourier space, resulting in linear growth of memory consumption with dimension. To minimize the required memory, we utilize streaming calculations, which maintains low memory requirements with minimal computational impact. If memory limitations persist, the set of desired frequencies can be split into d groups to further reduce the required memory by a factor of d.

The RSVD-Δt algorithm contains three sources of error, each of which can be controlled by carefully selecting method parameters. The first arises from the RSVD approximation inherited from the RSVD algorithm. This error can be significantly reduced by employing power iteration and utilizing more test vectors than the desired number. The second source of error stems from the time integration method employed to compute the action of $\boldsymbol{R}$ and $\boldsymbol{R}^*$. Time-stepping errors encompass the transient response and truncation error. Truncation error arises from the numerical integration of the LNS equations and can be managed through careful selection of the time-stepping scheme and time step. The transient response emerges when the initial condition is not synchronized with the applied forcing, decaying over time but potentially requiring many periods to become sufficiently small. To expedite the removal of transients, a novel strategy is introduced involving the decomposition of snapshots into transient and steady-state components, with subsequent solving of equations for the transient. This computation is facilitated through Petrov-Galerkin and Galerkin projections. To ensure optimal performance, it is important to maintain a balance between truncation and transient errors. Focusing too much on reducing one source significantly while neglecting the other can lead to a waste of CPU time without an impact on the outcome. Also, keeping both errors smaller than the RSVD approximation error will not improve the accuracy of RSVD-Δt with respect to SVD-based (true) results. By effectively eliminating both truncation and transient errors up to machine precision, RSVD-Δt has been validated against the RSVD-LU algorithm using the complex Ginzburg-Landau equation.

The RSVD-Δt algorithm is particularly valuable for analyzing three-dimensional flows, where other algorithms become impractical. The superior scalability of the RSVD-Δt algorithm leads to an increasingly pronounced disparity in computational complexity compared to the RSVD-LU algorithm as the value of N grows larger. As an example, we consider a moderately large state dimension of $N \approx 39$ million. Using the RSVD-LU algorithm for this problem would require an estimated 75 million CPU-hours and 130 TB of RAM. In contrast, the RSVD-Δt algorithm required just 18,000 CPU-hours and 0.75 TB of RAM, a reduction of three and two orders of magnitude, respectively. In general, the benefits of the RSVD-Δt algorithm are most pronounced for three dimensional flows and other large systems, while little advantage is gained for simple one- and two-dimensional flows.

Our algorithm also has several implementation advantages. Our time-stepping approach enables matrix-free implementation, eliminating the explicit formation of the LNS matrix $\boldsymbol{A}$, instead directly utilizing built-in linear direct and adjoint capabilities available within many existing codes. All operations within the RSVD-Δt algorithm are amenable to efficient parallelization; we have optimized out implementation of the algorithm for parallel computing using the PETSc [8] and SLEPc [58] environments, facilitating full utilization of the computational power offered by modern high-performance clusters. Moreover, our code is designed to leverage GPUs, enabling the delegation of compute-intensive tasks to the GPU architecture for quicker and more efficient calculations. Finally, the efficiency and accuracy of the RSVD-Δt algorithm could be further enhanced by incorporating strategies developed for the RSVD-LU algorithm. Notably, techniques proposed by Ribeiro *et al.* [118] and House *et al.* [62] can be integrated into our approach to use physical insight to select the initial test vectors instead of relying on entirely random ones.

This thesis presents an extension of the RSVD-Δt algorithm that was originally developed for resolvent analysis of steady base flows to handle harmonic resolvent analysis of periodic flows. Specifically, we demonstrate that the time-stepping technique employed within RSVD-Δt can effectively replace the actions of $\boldsymbol{H}$ and its complex conjugate $\boldsymbol{H}^*$ in Fourier space. In terms of CPU cost, the RSVD-Δt algorithm exhibits a scaling of $O(N)$ for both resolvent and harmonic resolvent analyses. This offers a significant advantage, considering that the LU decomposition of $\boldsymbol{T}$ scales as $O((N_\omega N)^c)$ with $c \geq 1.5$. Regarding memory usage, our algorithm only stores relevant matrices in Fourier space and exhibits $O(NN_\omega)$ scaling; in contrast, the memory peak of LU decomposition of $\boldsymbol{T}$ empirically scales with $O((N_\omega N)^{1.5})$ or worse. One difference between applying RSVD-Δt to resolvent and harmonic resolvent analyses lies in the necessity to update the LNS operators during the time-stepping process for the latter. These operators are efficiently generated on the fly from their N_b Fourier components.

The error sources in our algorithm align with those in resolvent analysis, extensively investigated in Farghadan *et al.* [44]. A unique contribution of this thesis is the introduction of a novel transient removal strategy for harmonic resolvent analysis. While reminiscent of our approach for resolvent operators, the challenge in harmonic resolvent analysis lies in the intertwining of all retained frequencies. Our strategy takes advantage of the differing evolution of the steady-state and transient components of the response and Petrov-Galrkin or Galerkin projections.

A validation of RSVD-Δt against RSVD-LU is conducted using a periodic Ginzburg-Landau system. Additionally, we verify the governing role of the Floquet exponent in determining the decay rate of the least-damped mode of the system. Extending the application of our algorithm, we analyze a two-dimensional flow passing an airfoil, providing insights into the CPU and memory complexities. Despite dealing with a mid-sized flow scenario, a substantial memory gap persists between RSVD-LU and RSVD-Δt algorithms. The computed forcing and response modes corresponding to the first suboptimal gain closely resemble those obtained by Padovan *et al.* [106]. The adaptation of the transient removal strategy for these test cases significantly enhanced the performance of RSVD-Δt, resulting in a 10-fold speed-up for the Ginzburg-Landau problem and a 60-fold speed-up for the airfoil to reach a 1% relative error. The speed-up is even greater for lower error tolerances.

Moreover, our algorithm exhibits versatility, accommodating non-identity weight, input, and output matrices. This capability extends to computing modes for the modified harmonic resolvent operator, and in particular, the analysis of cross-frequency amplification mechanisms via $\boldsymbol{H}_{\omega_2,\omega_1}$. Our algorithm tackles the computation of subharmonic resolvent modes if desired. Implementation of the RSVD-Δt algorithm within Petsc [8] and Slepc [58] environments leverage parallel computations for enhanced efficiency. Our time-stepping approach allows for a matrix-free application. This can be implemented using any code equipped with linear direct and adjoint capabilities, bypassing the explicit formation of $\boldsymbol{T}$ [108, 92]. Finally, the time stepping within our algorithm can also be used along with other SVD algorithms, *e.g.*, Arnoldi and power iteration, if desired.

In turbulent flows, where the state dimension N can become very large due to higher resolution requirements, CPU cost and memory requirements can swiftly impose constraints on the applicability of RSVD-LU. This limitation applies even to steady problems and is further exacerbated for harmonic resolvent analysis due to the inflated frequency-domain operators necessitated by triadic frequency coupling. Our algorithm circumvents this issue by strategically operating in the time domain, avoiding the need for limiting operations like LU decomposition, leading to linear cost scaling. Because of this key distinction, the RSVD-Δt could enable harmonic resolvent analysis of previous intractable turbulent flows.

In this thesis, we have investigated three distinct jet configurations—jet with streaks, twin jets, and screeching jets—each exhibiting unique flow characteristics and underlying mechanisms. Through a secondary stability analysis in which the steady streaks are added to the axisymmetric mean flow, we showed the significant impact of the streaks on the KH wavepackets. This included a substantial increase in gains within the range $St \in [0, 0.5]$, a change in the most amplified azimuthal wavenumber, and coupling of multiple azimuthal wavenumbers is some of the modes. Given the recently demonstrated presence of streaks in real jets, these finds warrant further investigation in the future.

The twin jet configuration, involving two single jets with a defined separation, was examined under four different boundary conditions. It was found that when the nozzle spacing is $s/D = 2$ the gains in the lower frequency range shift significantly at the SA boundary condition, while under other boundary conditions, the changes are minimal.

For screeching jets, we conducted both resolvent and harmonic resolvent analyses. Initially, the eigenspectra of the LNS operator revealed the presence of eigenmodes at non-screech frequencies with similar characteristics, an interesting topic for further analysis. The harmonic resolvent modes showed much smaller amplification compared to what was predicted by resolvent analysis, primarily due to nonlinear interactions between the screech frequency and its harmonics. This study highlights the importance of various nonlinear mechanisms in generating response modes at different frequencies, elucidating underlying mechanisms previously unclear. Specifically, the nonlinear interactions between $\hat{\boldsymbol{A}}_1$ and $\hat{\boldsymbol{q}}_1$ observed in experiments are now better understood.

6.2 Future work

While this thesis has made considerable progress, several areas remain for future exploration. Some potential directions for future work include:

- **Extension of the RSVD-Δt algorithm:** The RSVD-Δt algorithm, designed for efficient computation of resolvent modes using a time-stepping approach, is versatile and adaptable. Future work could explore extending this algorithm to compute "space-time" resolvent modes, as proposed by Lopez-Doriga *et al.* [86], and "wavelet-based" resolvent analysis, as proposed by Ballouz *et al.* [9]. Such an extension would involve modifications to the current framework to handle the added complexity of new variants of resolvent analysis, potentially leading to new insights into the spatiotemporal dynamics of complex flows.

- **Application to to additional inherently three-dimensional flows:** The development of the RSVD-Δt algorithm was primarily motivated by the need to analyze three-dimensional flows, which are computationally demanding using traditional methods. Future research should focus on applying the algorithm to a variety of three-dimensional flow configurations, including but not limited to boundary layers, wake flows, and various jet flows, including twin jets, elliptical jets, rectangular jets, among others. This work could uncover new physical phenomena that were previously computationally inaccessible.

- **Secondary stability analysis:** The thesis demonstrated the impact of streaks on KH wavepackets in a turbulent jet. The secondary analysis of the screech mode and its effects on screeching phenomena were also analyzed. Future studies could further investigate secondary stability analyses in different flow configurations. This line of research could lead to a deeper understanding of the interplay between various instability mechanisms in turbulent flows, investigates the transition to turbulence, and contribute to the development of advanced flow control strategies.

- **Exploration of nonlinear interactions:** While the current work has focused on linear and weakly nonlinear analyses, future research could extend the RSVD-Δt algorithm to explore fully nonlinear interactions in turbulent flows. This would involve coupling the algorithm with nonlinear solvers and could provide a more comprehensive understanding of turbulence dynamics, particularly in flows where nonlinear effects play a dominant role [119].

- **Optimization and parallelization:** Although the RSVD-Δt algorithm has been optimized for parallel computing, further improvements could be made by exploring advanced parallelization techniques, such as domain decomposition or GPU acceleration. Any improvements in either RSVD or time stepping of ODEs can be directly applicable to further optimizing our algorithm. Moreover, optimizing the algorithm for specific architectures, such as exascale computing platforms, could significantly enhance its performance, making it suitable for the most challenging computational problems in fluid dynamics.

The advancements presented in this thesis lay a strong foundation for future research and development in resolvent analysis and turbulence modeling. By exploring the opportunities outlined above, future work can continue to push the boundaries of our understanding and control of turbulent flows.

BIBLIOGRAPHY

[1] Ahn, M., Lee, D. & Mihaescu, M. 2021 A numerical study on near-field pressure fluctuations of symmetrical and anti-symmetrical flapping modes of twin-jet using a high-resolution shock-capturing scheme. *Aerospace Science and Technology* **119**, 107147.

[2] Amestoy, P. R., Buttari, A., l'Excellent, J. Y. & Mary, T. A. 2019 Bridging the gap between flat and hierarchical low-rank matrix formats: The multilevel block low-rank format. *SIAM Journal on Scientific Computing* **41** (3), A1414–A1442.

[3] Amestoy, P. R, Duff, I. S., L'Excellent, J. Y. & Koster, J. 2001 A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications* **23** (1), 15–41.

[4] Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. & others 1999 *LAPACK users' guide*. SIAM.

[5] Arnoldi, W. E. 1951 The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics* **9** (1), 17–29.

[6] Axelsson, O. 1985 A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT Numerical Mathematics* **25** (1), 165–187.

[7] Bagheri, S., Henningson, D. S., Hoepffner, J. & Schmid, P. J. 2009 Input-output analysis and control design applied to a linear model of spatially developing flows. *Applied Mechanics Reviews* **62** (2).

[8] Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. & others 2019 *PETSc users manual*. Argonne National Laboratory.

[9] Ballouz, E., Lopez-Doriga, B., Dawson, S. & Bae, H. J. 2024 Wavelet-based resolvent analysis of non-stationary flows. *arXiv preprint arXiv:2404.06600* .

[10] Barkley, D. 2016 Theoretical perspective on the route to turbulence in a pipe. *Journal of Fluid Mechanics* **803**, P1.

[11] Barthel, B., Gomez, S. & McKeon, B. J. 2022 Variational formulation of resolvent analysis. *Physical Review Fluids* **7** (1), 013905.

[12] Benzi, M. 2002 Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics* **182** (2), 418–477.

[13] Brès, G. A., Ham, F. E., Nichols, J. W. & Lele, S. K. 2017 Unstructured large-eddy simulations of supersonic jets. *AIAA journal* **55** (4), 1164–1184.

[14] Brès, G. A., Jordan, P., Jaunet, V., Le Rallic, M., Cavalieri, A. V. G., Towne, A., Lele, S. K., Colonius, T. & Schmidt, O. T. 2018 Importance of the nozzle-exit boundary-layer state in subsonic turbulent jets. *Journal of Fluid Mechanics* **851**, 83–124.

[15] Brunton, S. L. 2021 Applying machine learning to study fluid mechanics. *Acta Mechanica Sinica* **37** (12), 1718–1726.

[16] Brunton, S. L. & Kutz, J. N. 2022 *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press.

[17] Brunton, S. L. & Noack, B. R. 2015 Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews* **67** (5), 050801.

[18] Brynjell-Rahkola, M., Tuckerman, L. S., Schlatter, P. & Henningson, D. S. 2017 Computing optimal forcing using Laplace preconditioning. *Communications in Computational Physics* **22** (5), 1508–1532.

[19] Cavalieri, A. V. G., Jordan, P., Colonius, T. & Gervais, Y. 2012 Axisymmetric superdirectivity in subsonic jets. *Journal of Fluid Mechanics* **704**, 388–420.

[20] Cavalieri, A. V. G., Jordan, P. & Lesshafft, L. 2019 Wave-packet models for jet dynamics and sound radiation. *Applied Mechanics Reviews* **71** (2).

[21] Chavarin, A. & Luhar, M. 2020 Resolvent analysis for turbulent channel flow with riblets. *AIAA Journal* **58** (2), 589–599.

[22] Chen, K. K. & Rowley, C. W. 2011 H2 optimal actuator and sensor placement in the linearised complex Ginzburg–Landau system. *Journal of Fluid Mechanics* **681**, 241–260.

[23] Chomaz, J. M., Huerre, P. & Redekopp, L. G. 1988 Bifurcations to local and global modes in spatially developing flows. *Physical Review Letters* **60** (1), 25.

[24] Chu, B.-T. 1965 On the energy transfer to small disturbances in fluid flow (part i). *Acta Mechanica* **1** (3), 215–234.

[25] Cook, D. A. & Nichols, J. W. 2023 Three-dimensional receptivity of hypersonic sharp and blunt cones to free-stream planar waves using hierarchical input-output analysis. *arXiv preprint arXiv:2306.03248* .

[26] Cossu, C. & Brandt, L. 2002 Stabilization of Tollmien–Schlichting waves by finite amplitude optimal streaks in the Blasius boundary layer. *Physics of Fluids* **14** (8), L57–L60.

[27] Crow, S. C. & Champagne, F. H. 1971 Orderly structure in jet turbulence. *Journal of Fluid Mechanics* **48** (3), 547–591.

[28] Davis, T. A., Rajamanickam, S. & Sid-Lakhdar, W. M. 2016 A survey of direct methods for sparse linear systems. *Acta Numerica* **25**, 383–566.

[29] Dawson, S. T. M. & McKeon, B. J. 2019 On the shape of resolvent modes in wall-bounded turbulence. *Journal of Fluid Mechanics* **877**, 682–716.

[30] Drazin, P. G. 2002 *Introduction to hydrodynamic stability*, , vol. 32. Cambridge university press.

[31] Drazin, P. G. & Reid, W. H. 2004 *Hydrodynamic stability*. Cambridge university press.

[32] Duff, I. S., Erisman, A. M. & Reid, J. K. 2017 *Direct methods for sparse matrices*. Oxford University Press.

[33] Dunford, N. & Schwartz, J. T. 1958 *Linear operators part I: general theory*. John Wiley & Sons.

[34] Edgington-Mitchell, D. 2019 Aeroacoustic resonance and self-excitation in screeching and impinging supersonic jets–a review. *International Journal of Aeroacoustics* **18** (2-3), 118–188.

[35] Edgington-Mitchell, D., Li, X., Liu, N., He, F., Wong, T. Y., Mackenzie, J. & Nogueira, P. 2022 A unifying theory of jet screech. *Journal of Fluid Mechanics* **945**, A8.

[36] Edgington-Mitchell, D., Wang, T., Nogueira, P., Schmidt, O., Jaunet, V., Duke, D., Jordan, P. & Towne, A. 2021 Waves in screeching jets. *Journal of Fluid Mechanics* **913**, A7.

[37] Edwards, W. S., Tuckerman, L. S., Friesner, R. A. & Sorensen, D. C. 1994 Krylov methods for the incompressible Navier-Stokes equations. *Journal of computational physics* **110** (1), 82–102.

[38] Erichson, N. B., Voronin, S., Brunton, S. L. & Kutz, J. N. 2019 Randomized matrix decompositions using R. *Journal of Statistical Software* **89** (1), 1–48.

[39] Eriksson, L. E. & Rizzi, A. 1985 Computer-aided analysis of the convergence to steady state of discrete approximations to the Euler equations. *Journal of Computational Physics* **57** (1), 90–128.

[40] Esfahani, A., Webb, N. J. & Samimy, M. 2021 Coupling modes in supersonic twin rectangular jets. *AIAA Paper #2021-1292* .

[41] Falgout, R. D. & Yang, U. M. 2002 hypre: A library of high performance preconditioners. In *International Conference on Computational Science*, pp. 632–641.

[42] Farghadan, A. & Arzani, A. 2019 The combined effect of wall shear stress topology and magnitude on cardiovascular mass transport. *International Journal of Heat and Mass Transfer* **131**, 252–260.

[43] Farghadan, A., Jung, J., Bhagwat, R. & Towne, A. 2024 Efficient harmonic resolvent analysis via time stepping. *Theoretical and Computational Fluid Dynamics* pp. 1–23.

[44] Farghadan, A., Martini, E. & Towne, A. 2023 Scalable resolvent analysis for three-dimensional flows. *arXiv preprint arXiv:2309.04617* .

[45] Farghadan, A., Towne, A., Martini, E. & Cavalieri, A. 2021 A randomized time-domain algorithm for efficiently computing resolvent modes. *AIAA Paper #2021-2896* .

[46] Franceschini, L., Sipp, D., Marquet, O., Moulin, J. & Dandois, J. 2022 Identification and reconstruction of high-frequency fluctuations evolving on a low-frequency periodic limit cycle: application to turbulent cylinder flow. *Journal of Fluid Mechanics* **942**, A28.

[47] Gao, J., Xu, X. & Li, X. 2018 Numerical simulation of supersonic twin-jet noise with high-order finite difference scheme. *AIAA Journal* **56** (1), 290–300.

[48] Giannenas, A. E., Laizet, S. & Rigas, G. 2022 Harmonic forcing of a laminar bluff body wake with rear pitching flaps. *Journal of Fluid Mechanics* **945**, A5.

[49] Gojon, R., Bogey, C. & Mihaescu, M. 2018 Oscillation modes in screeching jets. *AIAA Journal* **56** (7), 2918–2924.

[50] Gómez, F., Sharma, A. S. & Blackburn, H. M. 2016 Estimation of unsteady aerodynamic forces using pointwise velocity data. *Journal of Fluid Mechanics* **804**, R4.

[51] Goparaju, K. & Gaitonde, D. V. 2018 Dynamics of closely spaced supersonic jets. *Journal of Propulsion and Power* **34** (2), 327–339.

[52] Hairer, E., Nørsett, S.P. & Wanner, G. 1993 *Solving ordinary differential equations I: nonstiff problems*. Springer Berlin Heidelberg.

[53] Halko, N., Martinsson, P. & Tropp, J. A. 2011 Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review* **53** (2), 217–288.

[54] Heidt, L. & Colonius, T. 2023 Spectral proper orthogonal decomposition of harmonically forced turbulent flows. *arXiv preprint arXiv:2305.05628* .

[55] Heisenberg, W. 1985 *Über stabilität und turbulenz von flüssigkeitsströmen*. Springer.

[56] Henderson, B. 2010 Fifty years of fluidic injection for jet noise reduction. *International Journal of Aeroacoustics* **9** (1-2), 91–122.

[57] Herbert, T. 1984 Analysis of the subharmonic route to transition in boundary layers. In *22nd Aerospace Sciences Meeting*, p. 9.

[58] Hernandez, V., Roman, J. E. & Vidal, V. 2005 Slepc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)* **31** (3), 351–362.

[59] Herrmann, B., Baddoo, P. J., Semaan, R., Brunton, S. L. & McKeon, B. J. 2021 Data-driven resolvent analysis. *Journal of Fluid Mechanics* **918**, A10.

[60] Hileman, J. I., Thurow, B. S., Caraballo, E. J. & Samimy, M. 2005 Large-scale structure evolution and sound emission in high-speed jets: real-time visualization with simultaneous acoustic measurements. *Journal of Fluid Mechanics* **544**, 277–307.

[61] Holmes, P 2012 *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press.

[62] House, D., Skene, C., Ribeiro, J. H. M., Yeh, C.-A. & Taira, K. 2022 Sketch-based resolvent analysis. *AIAA Paper #2022-3335* .

[63] Houtman, J., Timme, S. & Sharma, A. 2023 Resolvent analysis of a finite wing in transonic flow. *Flow* **3**, E14.

[64] Hoyas, S. & Jiménez, J. 2006 Scaling of the velocity fluctuations in turbulent channels up to reτ= 2003. *Physics of Fluids* **18** (1).

[65] Hunt, R. E. & Crighton, D. G. 1991 Instability of flows in spatially developing media. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* **435** (1893), 109–128.

[66] Jeun, J., Nichols, J. W. & Jovanović, M. R. 2016 Input-output analysis of high-speed axisymmetric isothermal jet noise. *Physics of Fluids* **28** (4), 047101.

[67] Jiménez, J. 2018 Coherent structures in wall-bounded turbulence. *Journal of Fluid Mechanics* **842**, P1.

[68] Jimenez-Gonzalez, J. I. & Brancher, P. 2017 Transient energy growth of optimal streaks in parallel round jets. *Physics of Fluids* **29** (11), 114101.

[69] Jin, B., Symon, S. & Illingworth, S. J. 2021 Energy transfer mechanisms and resolvent analysis in the cylinder wake. *Physical Review Fluids* **6** (2), 024702.

[70] Joseph, Daniel D 2013 *Stability of fluid motions I*, , vol. 27. Springer Science & Business Media.

[71] Jovanovic, M. R. 2004 *Modeling, analysis, and control of spatially distributed systems*. University of California, Santa Barbara.

[72] Jovanović, M. R. 2021 From bypass transition to flow control and data-driven turbulence modeling: An input–output viewpoint. *Annual Review of Fluid Mechanics* **53** (1), 010719–060244.

[73] Jung, J., Bhagwat, R. & Towne, A. 2023 Resolvent-based estimation of laminar flow around an airfoil. *AIAA Paper #2023-0077* .

[74] Kamal, O., Lakebrink, M. T. & Colonius, T. 2023 Global receptivity analysis: physically realizable input–output analysis. *Journal of Fluid Mechanics* **956**, R5.

[75] Karban, U., Bugeat, B., Martini, E., Towne, A., Cavalieri, A. V. G., Lesshafft, L., Agarwal, A., Jordan, P. & Colonius, T. 2020 Ambiguity in mean-flow-based linear analysis. *Journal of Fluid Mechanics* **900**, R5.

[76] Kato, T. 2013 *Perturbation theory for linear operators*. Springer Science & Business Media.

[77] Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P. & Hoyer, S. 2021 Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences* **118** (21), e2101784118.

[78] Kumar, C. & Prakash, A. 2022 Effect of mass injection on secondary instability of hypersonic boundary layer over a blunt cone. *Physics of Fluids* **34** (6).

[79] Laban, A., Aleyasin, S., Tachie, M. F. & Koupriyanov, M. 2019 Experimental investigation of nozzle spacing effects on characteristics of round twin free jets. *Journal of Fluids Engineering* **141** (7), 071201.

[80] Lall, S., Marsden, J. E. & Glavaški, S. 2002 A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* **12** (6), 519–535.

[81] Leclercq, C. & Sipp, D. 2023 Mean resolvent operator of a statistically steady flow. *Journal of Fluid Mechanics* **968**, A13.

[82] LeGresley, P. & Alonso, J. 2000 Airfoil design optimization using reduced order models based on proper orthogonal decomposition. In *Fluids 2000 Conference and Exhibit*, p. 2545.

[83] Lesshafft, L., Semeraro, O., Jaunet, V., Cavalieri, A. V. G. & Jordan, P. 2019 Resolvent-based modeling of coherent wave packets in a turbulent jet. *Physical Review Fluids* **4** (6), 063901.

[84] Li, F. & Malik, M. R. 1996 On the nature of PSE approximation. *Theoretical and Computational Fluid Dynamics* **8** (4), 253–273.

[85] Lin, C.-T., Tsai, M.-L. & Tsai, H.-C. 2023 Flow control of a plunging cylinder based on resolvent analysis. *Journal of Fluid Mechanics* **967**, A41.

[86] Lopez-Doriga, B., Ballouz, E., Bae, H. J. & Dawson, S. 2024 Sparse space-time resolvent analysis for statistically-stationary and time-varying flows. *arXiv preprint arXiv:2404.06331* .

[87] Lumley, J. L. 1967 The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation* pp. 166–178.

[88] Marant, M. & Cossu, C. 2018 Influence of optimally amplified streamwise streaks on the Kelvin–Helmholtz instability. *Journal of Fluid Mechanics* **838**, 478–500.

[89] Marquet, O. & Larsson, M. 2015 Global wake instabilities of low aspect-ratio flat-plates. *European Journal of Mechanics-B/Fluids* **49**, 400–412.

[90] Martini, E., Cavalieri, A. V. G., Jordan, P., Towne, A. & Lesshafft, L. 2020 Resolvent-based optimal estimation of transitional and turbulent flows. *Journal of Fluid Mechanics* **900**, A2.

[91] Martini, E., Jung, J., Cavalieri, A. V. G., Jordan, P. & Towne, A. 2022 Resolvent-based tools for optimal estimation and control via the Wiener–Hopf formalism. *Journal of Fluid Mechanics* **938**, E2.

[92] Martini, E., Rodríguez, D., Towne, A. & Cavalieri, A. V. G. 2021 Efficient computation of global resolvent modes. *Journal of Fluid Mechanics* **919**, A3.

[93] Mattsson, K. & Nordström, J. 2004 Summation by parts operators for finite difference approximations of second derivatives. *Journal of Computational Physics* **199** (2), 503–540.

[94] McKeon, B. J. 2017 The engine behind (wall) turbulence: perspectives on scale interactions. *Journal of Fluid Mechanics* **817**, P1.

[95] McKeon, B. J. & Sharma, A. S. 2010 A critical-layer framework for turbulent pipe flow. *Journal of Fluid Mechanics* **658**, 336–382.

[96] Moarref, R., Sharma, A. S., Tropp, J. A. & McKeon, B. J. 2013 Model-based scaling of the streamwise energy density in high-Reynolds-number turbulent channels. *Journal of Fluid Mechanics* **734**, 275–316.

[97] Moin, P. & Mahesh, K. 1998 Direct numerical simulation: a tool in turbulence research. *Annual Review of Fluid Mechanics* **30** (1), 539–578.

[98] Monokrousos, A., Åkervik, E., Brandt, L. & Henningson, D. S. 2010 Global three-dimensional optimal disturbances in the Blasius boundary-layer flow using time-steppers. *Journal of Fluid Mechanics* **650**, 181–214.

[99] Morra, P., Semeraro, O., Henningson, D. S. & Cossu, C. 2019 On the relevance of Reynolds stresses in resolvent analyses of turbulent wall-bounded flows. *Journal of Fluid Mechanics* **867**, 969–984.

[100] Morris, P. J. 1990 Instability waves in twin supersonic jets. *Journal of Fluid Mechanics* **220**, 293–307.

[101] Naseri O., Ramin, Tachie, M. F. & Wang, B. 2019 Effect of nozzle spacing on turbulent interaction of low-aspect-ratio twin rectangular jets. *Flow, Turbulence and Combustion* **103**, 323–344.

[102] Nogueira, P. A. S., Cavalieri, A. V. G., Jordan, P. & Jaunet, V. 2019 Large-scale streaky structures in turbulent jets. *Journal of Fluid Mechanics* **873**, 211–237.

[103] Nogueira, P. A. S. & Edgington-Mitchell, D. M. 2021 Investigation of supersonic twin-jet coupling using spatial linear stability analysis. *Journal of Fluid Mechanics* **918**, A38.

[104] Nyquist, H. 1928 Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers* **47** (2), 617–644.

[105] Orr, W. M. F. 1907 The stability or instability of the steady motions of a perfect liquid and of a viscous liquid. part ii: A viscous liquid. In *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*, , vol. 27, pp. 69–138. JSTOR.

[106] Padovan, A., Otto, S. E. & Rowley, C. W. 2020 Analysis of amplification mechanisms and cross-frequency interactions in nonlinear flows via the harmonic resolvent. *Journal of Fluid Mechanics* **900**.

[107] Padovan, A. & Rowley, C. W. 2022 Analysis of the dynamics of subharmonic flow structures via the harmonic resolvent: Application to vortex pairing in an axisymmetric jet. *Physical Review Fluids* **7** (7), 073903.

[108] de Pando, M. F., Sipp, D. & Schmid, P. J. 2012 Efficient evaluation of the direct and adjoint linearized dynamics from compressible flow solvers. *Journal of Computational Physics* **231** (23), 7739–7755.

[109] Pearson, J. W. & Pestana, J. 2020 Preconditioners for krylov subspace methods: An overview. *GAMM-Mitteilungen* **43** (4), e202000015.

[110] Peherstorfer, B. & Willcox, K. 2015 Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering* **291**, 21–41.

[111] Pickering, E., Rigas, G., Nogueira, P. A. S., Cavalieri, A. V. G., Schmidt, O. T. & Colonius, T. 2020 Lift-up, Kelvin–Helmholtz and Orr mechanisms in turbulent jets. *Journal of Fluid Mechanics* **896**, A2.

[112] Pickering, E., Rigas, G., Schmidt, O. T., Sipp, D. & Colonius, T. 2021 Optimal eddy viscosity for resolvent-based models of coherent structures in turbulent jets. *Journal of Fluid Mechanics* **917**, A29.

[113] Pickering, E., Towne, A., Jordan, P. & Colonius, T. 2021 Resolvent-based modeling of turbulent jet noise. *The Journal of the Acoustical Society of America* **150** (4), 2421–2433.

[114] Powell, A. 1953 On edge tones and associated phenomena. *Acta Acustica United with Acustica* **3** (4), 233–243.

[115] Raman, G. 1999 Supersonic jet screech: half-century from powell to the present. *Journal of Sound and Vibration* **225** (3), 543–571.

[116] Ran, W., Zare, A. & Jovanović, M. R. 2021 Model-based design of riblets for turbulent drag reduction. *Journal of Fluid Mechanics* **906**, A7.

[117] Reynolds, O. 1883 Xxix. an experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philosophical Transactions of the Royal society of London* (174), 935–982.

[118] Ribeiro, J. H. M., Yeh, C.-A. & Taira, K. 2020 Randomized resolvent analysis. *Physical Review Fluids* **5** (3), 033902.

[119] Rigas, G., Sipp, D. & Colonius, T. 2021 Nonlinear input/output analysis: application to boundary layer transition. *Journal of Fluid Mechanics* **911**, A15.

[120] Rodríguez, D. 2021 Wavepacket models for supersonic twin-jets. *AIAA Paper #2021-2121* .

[121] Rodríguez, D., Jotkar, M. R. & Gennaro, E. M. 2018 Wavepacket models for subsonic twin jets using 3d parabolized stability equations. *Comptes Rendus Mécanique* **346** (10), 890–902.

[122] Rodríguez, D., Stavropoulos, M. N., Nogueira, P. A. S., Edgington-Mitchell, Daniel M. & Jordan, P. 2023 On the preferred flapping motion of round twin jets. *Journal of Fluid Mechanics* **977**, A4.

[123] Rolandi, L. V., Ribeiro, J. H. M., Yeh, C. & Taira, K. 2024 An invitation to resolvent analysis. *arXiv preprint arXiv:2404.11789* .

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

[125] Saad, Y. 2003 Finding exact and approximate block structures for ilu preconditioning. *SIAM Journal on Scientific Computing* **24** (4), 1107–1123.

[126] Saad, Y. 2003 *Iterative methods for sparse linear systems*. SIAM.

[127] Sagaut, P. 2005 *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media.

[128] Sasaki, K., Cavalieri, A. V. G., Hanifi, A. & Henningson, D. S. 2022 Parabolic resolvent modes for streaky structures in transitional and turbulent boundary layers. *Physical Review Fluids* **7** (10), 104611.

[129] Schenk, O., Gärtner, K., Fichtner, W. & Stricker, A. 2001 Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems* **18** (1), 69–78.

[130] Schmid, P. J. 2000 Linear stability theory and bypass transition in shear flows. *Physics of Plasmas* **7** (5), 1788–1794.

[131] Schmid, P. J. 2007 Nonmodal stability theory. *Annual Review of Fluid Mechanics* **39** (1), 129–162.

[132] Schmid, P. J. 2010 Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics* **656**, 5–28.

[133] Schmid, P. J. 2022 Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics* **54**, 225–254.

[134] Schmid, P. J. & Henningson, D. S. 2001 *Stability and transition in shear flows*. Springer, New York.

[135] Schmidt, O. T. & Towne, A. 2019 An efficient streaming algorithm for spectral proper orthogonal decomposition. *Computer Physics Communications* **237**, 98–109.

[136] Schmidt, O. T., Towne, A., Colonius, T., Cavalieri, A. V. G., Jordan, P. & Brès, G. A. 2017 Wavepackets and trapped acoustic modes in a turbulent jet: coherent structure eduction and global stability. *Journal of Fluid Mechanics* **825**, 1153–1181.

[137] Schmidt, O. T., Towne, A., Rigas, G., Colonius, T. & Brès, G. A. 2018 Spectral analysis of jet turbulence. *Journal of Fluid Mechanics* **855**, 953–982.

[138] Sinha, A., Gudmundsson, K., Xia, H. & Colonius, T. 2016 Parabolized stability analysis of jets from serrated nozzles. *Journal of Fluid Mechanics* **789**, 36–63.

[139] Sipp, D. & Marquet, O. 2013 Characterization of noise amplifiers with global singular modes: the case of the leading-edge flat-plate boundary layer. *Theoretical and Computational Fluid Dynamics* **27** (5), 617–635.

[140] Sirovich, L. 1987 Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of Applied Mathematics* **45** (3), 561–571.

[141] Sirovich, L. 1987 Turbulence and the dynamics of coherent structures. ii. symmetries and transformations. *Quarterly of Applied Mathematics* **45** (3), 573–582.

[142] Skeel, R. D. 1979 Scaling for numerical stability in Gaussian elimination. *Journal of the ACM (JACM)* **26** (3), 494–526.

[143] Sommerfeld, A. 1909 *Ein beitrag zur hydrodynamischen erklaerung der turbulenten fluessigkeitsbewegungen*.

[144] Stavropoulos, M. N., Martini, E., Edgington-Mitchell, D. M., Weightman, J., Jordan, P. & Nogueira, P. A. S. 2024 On the behaviour of the upstream-travelling waves in merging twin-jet systems. *Journal of Fluid Mechanics* **983**, A17.

[145] Stewart, G. W. 1993 On the early history of the singular value decomposition. *SIAM review* **35** (4), 551–566.

[146] Stewartson, K. & Stuart, J. T. 1971 A non-linear instability theory for a wave system in plane Poiseuille flow. *Journal of Fluid Mechanics* **48** (3), 529–545.

[147] Süli, E. & Mayers, D. F. 2003 *An introduction to numerical analysis*. Cambridge university press.

[148] Symon, S., Sipp, D. & McKeon, B. J. 2019 A tale of two airfoils: resolvent-based modelling of an oscillator versus an amplifier from an experimental mean. *Journal of Fluid Mechanics* **881**, 51–83.

[149] Taira, K., Brunton, S. L., Dawson, S. T. M., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V. & Ukeiley, L. S. 2017 Modal analysis of fluid flows: An overview. *AIAA Journal* **55** (12), 4013–4041.

[150] Tam, C. K. & Ahuja, K. 1990 Theoretical model of discrete tone generation by impinging jets. *Journal of Fluid Mechanics* **214**, 67–87.

[151] Tam, C. K. & Tanna, H. K. 1982 Shock associated noise of supersonic jets from convergent-divergent nozzles. *Journal of Sound and Vibration* **81** (3), 337–358.

[152] Theofilis, V. 2011 Global linear instability. *Annual Review of Fluid Mechanics* **43**, 319–352.

[153] Thomareis, N. & Papadakis, G. 2018 Resolvent analysis of separated and attached flows around an airfoil at transitional Reynolds number. *Physical Review Fluids* **3** (7), 073901.

[154] Towne, A. 2016 Advancements in jet turbulence and noise modeling: accurate one-way solutions and empirical evaluation of the nonlinear forcing of wavepackets. PhD thesis, California Institute of Technology.

[155] Towne, A. & Colonius, T. 2015 One-way spatial integration of hyperbolic equations. *Journal of Computational Physics* **300**, 844–861.

[156] Towne, A., Colonius, T., Jordan, P., Cavalieri, A. V. & Bres, G. A. 2015 Stochastic and nonlinear forcing of wavepackets in a Mach 0.9 jet. *AIAA Paper #2015-2217* .

[157] Towne, A., Lozano-Durán, A. & Yang, X. 2020 Resolvent-based estimation of space–time flow statistics. *Journal of Fluid Mechanics* **883**, A17.

[158] Towne, A., Rigas, G. & Colonius, T. 2019 A critical assessment of the parabolized stability equations. *Theoretical and Computational Fluid Dynamics* **33**, 359–382.

[159] Towne, A., Rigas, G., Kamal, O., Pickering, E. & Colonius, T. 2022 Efficient global resolvent analysis via the one-way Navier-Stokes equations. *Journal of Fluid Mechanics* **948**, A9.

[160] Towne, A., Schmidt, O. T. & Colonius, T. 2018 Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics* **847**, 821–867.

[161] Trefethen, L. N. & Bau III, D. 1997 *Numerical linear algebra*. Siam.

[162] Trefethen, L. N., Trefethen, A. E., Reddy, S. C. & Driscoll, T. A. 1993 Hydrodynamic stability without eigenvalues. *Science* **261** (5121), 578–584.

[163] Troy, C. D. & Koseff, J. R. 2005 The instability and breaking of long internal waves. *Journal of Fluid Mechanics* **543**, 107–136.

[164] Vershynin, R. 2018 *High-dimensional probability: An introduction with applications in data science*. Cambridge University Press.

[165] Wang, C., Lesshafft, L., Cavalieri, A. V. G. & Jordan, P. 2021 The effect of streaks on the instability of jets. *Journal of Fluid Mechanics* **910**, A14.

[166] Wanner, G. & Hairer, E. 1996 *Solving ordinary differential equations II: stiff and differential-algebraic problems*. Springer Berlin Heidelberg.

[167] Wereley, N. M. 1990 Analysis and control of linear periodically time varying systems. PhD thesis, Massachusetts Institute of Technology.

[168] Willcox, K. & Peraire, J. 2002 Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal* **40** (11), 2323–2330.

[169] Wu, W., Meneveau, C., Mittal, R., Padovan, A., Rowley, C. W. & Cattafesta, L. 2022 Response of a turbulent separation bubble to zero-net-mass-flux jet perturbations. *Physical Review Fluids* **7** (8), 084601.

[170] Yeh, C.-A., Benton, S. I., Taira, K. & Garmann, D. J. 2020 Resolvent analysis of an airfoil laminar separation bubble at Re = 500 000. *Physical Review Fluids* **5** (8), 083906.

[171] Yeh, C.-A. & Taira, K. 2019 Resolvent-analysis-based design of airfoil separation control. *Journal of Fluid Mechanics* **867**, 572–610.

[172] Yeung, B., Schmidt, O. T. & Brès, G. A. 2022 Three-dimensional spectral pod of supersonic twin-rectangular jet flow. *AIAA Paper #2022-3345* .

[173] Zhu, M. & Towne, A. 2023 Recursive one-way Navier-Stokes equations with PSE-like cost. *Journal of Computational Physics* **473**, 111744.