Scalable resolvent analysis for three-dimensional flows

Ali Farghadan¹, Eduardo Martini², Aaron Towne^{*1}

¹Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA ²Institut Pprime, CNRS, Université de Poitiers ISAE-ENSMA, Poitiers, France

Abstract.

Resolvent analysis is a powerful tool for studying coherent structures in turbulent flows. However, its application beyond canonical flows with symmetries that can be used to simplify the problem to inherently three-dimensional flows and other large systems has been hindered by the computational cost of computing resolvent modes. In particular, the CPU and memory requirements of state-of-the-art algorithms scale poorly with the problem dimension, *i.e.*, the number of discrete degrees of freedom. In this paper, we present RSVD- Δt , a novel approach that overcomes these limitations by combining randomized singular value decomposition with an optimized time-stepping method for computing the action of the resolvent operator. Critically, the CPU cost and memory requirements of the algorithm scale linearly with the problem dimension, and we develop additional strategies to minimize these costs and control errors. We validate the algorithm using a Ginzburg-Landau test problem and demonstrate its low cost and improved scaling using a three-dimensional discretization of a turbulent jet. Lastly, we use it to study the impact of low-speed streaks on the development of Kelvin-Helmholtz wavepackets in the jet via secondary stability analysis, a problem that would have been intractable using previous algorithms.

1 Introduction

Turbulent flows are characterized by chaotic and disorganized motions, but recurring dominant patterns can play a significant role in laminar to turbulent transition (Schmid & Henningson, 2001) and sustaining turbulence (McKeon, 2017). These 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 (Taira *et al.*, 2017). Popular data-driven methods include proper orthogonal decomposition (POD) (Sirovich, 1987*a*), dynamic mode decomposition (DMD) (Schmid, 2010), and spectral proper orthogonal decomposition (SPOD) (Lumley, 1967; Towne *et al.*, 2018). In particular, SPOD identifies energy-ranked, single-frequency structures that evolve coherently in space and time.

^{*}Email address for correspondence: towne@umich.edu

Resolvent (or input-output) analysis originates from classical control theory (Dunford & Schwartz, 1958; Kato, 2013) and has become arguably the most important operator-theoretic modal decomposition techniques in fluid mechanics (McKeon & Sharma, 2010; Taira et al., 2017; Jovanović, 2021). Resolvent analysis has been applied to a wide variety of flows, including canonical wall-bounded flows (Dawson & McKeon, 2019; Morra et al., 2019), turbulent jets (Jeun et al., 2016; Schmidt et al., 2018; Lesshafft et al., 2019; Pickering et al., 2020), and airfoils (Thomareis & Papadakis, 2018; Yeh et al., 2020). It has been used for diverse tasks including design optimization (Chavarian & Luhar, 2020; Ran et al., 2021), receptivity analysis (Kamal et al., 2023; Cook & Nichols, 2023), and flow control (Yeh & Taira, 2019; Towne et al., 2020; Martini et al., 2020, 2022). Singular value decomposition (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 (Towne *et al.*, 2018; McKeon & Sharma, 2010). Specifically, the resolvent modes associated with the largest singular values provide an approximation of the leading SPOD modes (Towne et al., 2018) and, in some cases, capture the majority of the power spectral density (PSD) of the flow (Symon et al., 2019). 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) (Halko *et al.*, 2011) has been shown to further reduce the cost of resolvent analysis of one- (Moarref *et al.*, 2013) and two-dimensional (Ribeiro *et al.*, 2020) 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 §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 (Sipp & Marquet, 2013; Jeun *et al.*, 2016; Schmidt *et al.*, 2018; Karban *et al.*, 2020). Brynjell-Rahkola *et al.* (2017) used LU decomposition along with a power iteration with a Laplace preconditioner to increase the convergence rate of the resolvent modes. More recently, Ribeiro *et al.* (2020) used LU decomposition along with RSVD, which we call "RSVD-LU" in this study, 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.

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 (Li & Malik, 1996) requires deleterious regularization that makes it ill-suited to compute resolvent modes in most cases (Towne *et al.*. 2019). One exception is very low frequencies, where PSE has been used to compute resolvent modes corresponding to boundary-layer streaks (Sasaki et al., 2022). The one-way Navier–Stokes (OWNS) equations (Towne & Colonius, 2015) 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 (Towne et al., 2022). 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 (Zhu & Towne, 2023). 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.* (2015) and Towne (2016) 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 dynamic mode decomposition (DMD) (Schmid, 2010) to estimate the resolvent modes from data (Herrmann *et al.*, 2021). This approach benefits from the advancements in DMD (Schmid, 2022) and is robust, but to accurately approximate the resolvent modes, many random initial conditions may need to be simulated.

Barthel *et al.* (2022) 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.* (2022) 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.* (2010) used

time stepping along with power iteration to compute resolvent modes of 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.* (2016) 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.* (2021) introduced two additional variations of time-stepping approaches for computing resolvent modes with improved efficiency. The first, referred to as the transitional response method, evaluates the transitional response of the LNS to compact forcing. The second variation, known as the steadystate 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 paper, 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.* (2021). 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.

In the remainder of the paper, we provide a brief review of the formulation and computation of resolvent analysis in §2, discuss the RSVD-LU algorithm in §3, explain the time-stepping method in §4, and introduce our RSVD- Δt algorithm in §5. An overview of the computational complexity of all approaches is given in §6, the sources of errors of our algorithm are detailed in §7, and approaches to optimize the algorithm are developed in §8. Two test cases are defined in §9 to validate, examine and compare the accuracy and performance of RSVD- Δt against other approaches. In §10, we use RSVD- Δt to study the impact of streaks on the Kelvin-Helmholtz wavepackets in a jet. Concluding remarks are made in §11.

2 Resolvent analysis

2.1 Formulation

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

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

where the nonlinear Navier-Stokes operator \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{q} and the fluctuation q'. Substituting (2.2) into (2.1) yields

$$\frac{\partial \boldsymbol{q}'}{\partial t} = \boldsymbol{A}(\bar{\boldsymbol{q}})\boldsymbol{q}' + \boldsymbol{B}\boldsymbol{f}'(\bar{\boldsymbol{q}},\boldsymbol{q}'),$$

$$\boldsymbol{y}' = \boldsymbol{C}\boldsymbol{q}',$$

(2.3)

where $\mathbf{A} \in \mathbb{C}^{N \times N}$ is the linearized Navier-Stokes (LNS) operator, $\mathbf{B} \in \mathbb{C}^{N \times N_f}$ is an input matrix that can be used to restrict the forcing $\mathbf{f}' \in \mathbb{C}^{N_f}$, and $\mathbf{C} \in \mathbb{C}^{N_y \times N}$ is an output matrix that extracts the output of interest $\mathbf{y}' \in \mathbb{C}^{N_y}$ from the state. The forcing \mathbf{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 A is unstable, discounting can be used to obtain a stable system (Jovanovic, 2004; Yeh & Taira, 2019). 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

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

of (2.3) and solving for the output yields

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

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

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

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

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

$$\sigma = \frac{||\hat{y}||_q}{||\hat{f}||_f} = \frac{||\boldsymbol{R}\hat{f}||_q}{||\hat{f}||_f},$$
(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 Appendix A.

Solving the Rayleigh quotient (2.7) is equivalent to computing the SVD of the resolvent operator (Stewart, 1993)

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

where Σ contains the singular values (aka *gains*), and V and U are right and left singular vectors corresponding to input and output vectors (aka 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 (Moarref *et al.*, 2013), 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 (Sipp & Marquet, 2013; Jeun *et al.*, 2016; Schmidt *et al.*, 2018; Thomareis & Papadakis, 2018; Karban *et al.*, 2020). 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 $\mathbf{R}^*\mathbf{R}$, *i.e.*, $\mathbf{R}^*\mathbf{R} = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^*$. By computing the leading eigenmodes of $\mathbf{R}^*\mathbf{R}$, both right singular vectors and square of singular values of the resolvent operator are obtained. Recovering the left singular vectors is done via $\mathbf{U} = \mathbf{R}\mathbf{V}\boldsymbol{\Sigma}^{-1}$. The leading eigenvalues and eigenvectors can be efficiently computed via Arnoldi iteration (Arnoldi, 1951). The cost of the Arnoldi method relies on the desired number of modes and the convergence threshold. Computing the LU decomposition of $(i\omega I - \mathbf{A})$ circumvents computing \mathbf{R} directly. This is a common practice to speed up the process of constructing the orthonormal basis of the Krylov subspace (Theofilis, 2011). However, the $O(N^2)$ scaling remains poor for three-dimensional systems.

The main objective of this paper 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.

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 (Halko *et al.*, 2011) by sampling its image and range. In the following two subsections, we introduce the RSVD algorithm and discuss its application to resolvent analysis.

3.1 RSVD algorithm

\triangleright Create random test matrices
\triangleright Sample the range of R
\triangleright Optional power iteration
\triangleright Algorithm 2
\triangleright Build the orthonormal subspace \boldsymbol{Q}
\triangleright Sample the image of R
\triangleright Obtain $\boldsymbol{\Sigma}, \boldsymbol{V}$
\triangleright Recover U

There exist several variations of the RSVD algorithm; here, we outline the algorithm from Halko

et al. (2011). The first step is to sample the range of \mathbf{R} by forming its sketch (line 3)

$$\boldsymbol{Y} = \boldsymbol{R}\boldsymbol{\Theta},\tag{3.1}$$

where $\boldsymbol{\Theta}$ 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 (Vershynin, 2018), 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} for the sketch is obtained via QR decomposition (line 6), which is then used to sample the image of \boldsymbol{R} (line 7) as

$$\mathbf{S} = \mathbf{Q}^* \mathbf{R}. \tag{3.2}$$

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

$$(\boldsymbol{R}\boldsymbol{R}^*)^q \boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}(\boldsymbol{V}^*\boldsymbol{V})\boldsymbol{\Sigma}\boldsymbol{U}^*)^q \boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^*)^q \boldsymbol{Y} = (\boldsymbol{U}\boldsymbol{\Sigma}^{2q}\boldsymbol{U}^*)\boldsymbol{Y}.$$
(3.3)

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.* (2011).

\triangleright For stabilization purposes
\triangleright Sample the image of R
\triangleright For stabilization purposes
\triangleright Sample the range of R

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.* (2020) addressed this challenge by adopting the approach developed by Jeun *et al.* (2016) for computing resolvent modes using an Arnoldi algorithm.

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

$$(i\omega I - A)Y = \Theta \tag{3.4}$$

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

$$(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^* \boldsymbol{S}^* = \boldsymbol{Q}. \tag{3.5}$$

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

$$(i\omega I - A) = LP, \tag{3.6}$$

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 $(i\omega \boldsymbol{I} - \boldsymbol{A})^*$ since

$$(\mathrm{i}\omega I - \mathbf{A})^* = (\mathbf{L}\mathbf{P})^* = \mathbf{P}^* \mathbf{L}^*. \tag{3.7}$$

Solving these linear systems is indeed significantly less computationally demanding than computing the inverse of $(i\omega I - A)$ to form 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 paper, we will use the term "RSVD-LU" to refer to the modified version of RSVD that is compatible with resolvent analysis (Ribeiro *et al.*, 2020).

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.* (2010) and recently was improved upon by Martini *et al.* (2021), 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 paper.

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)

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{A}\boldsymbol{q} + \boldsymbol{f},\tag{4.1}$$

where

$$\boldsymbol{f}(t) = \hat{\boldsymbol{f}} e^{\mathrm{i}\omega t} \tag{4.2}$$

is the harmonic forcing with frequency $\omega \in \mathbb{R}$ and $\hat{f} \in \mathbb{C}^N$ is an arbitrary vector. The steady-state response of (4.1) is

$$\boldsymbol{q}(t) = \hat{\boldsymbol{q}}_s e^{i\omega t},\tag{4.3}$$

where

$$\hat{\boldsymbol{q}}_s = (\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\hat{\boldsymbol{f}} = \boldsymbol{R}\hat{\boldsymbol{f}}$$
(4.4)



Figure 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.

is the Fourier-domain solution. Therefore, the action of \mathbf{R} can be obtained by computing the steadystate solution q(t) of (4.1) and subsequently taking a Fourier transform to obtain \hat{q}_s . Similarly, the action of \mathbf{R}^* can be obtained by computing the steady-state response $\mathbf{z}(t)$ of the adjoint equation

$$-\frac{d\boldsymbol{z}}{dt} = \boldsymbol{A}^* \boldsymbol{z} + \boldsymbol{f},$$

$$\boldsymbol{f} = \hat{\boldsymbol{f}} e^{\mathrm{i}\omega t},$$

(4.5)

backward in time and taking a Fourier transform to obtain

$$\hat{\boldsymbol{z}}_s = (-\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A}^*)^{-1}\hat{\boldsymbol{f}} = \boldsymbol{R}^*\hat{\boldsymbol{f}}.$$
(4.6)

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

4.2 Direct and adjoint actions for a range of frequencies

This section describes an important contribution from Martini *et al.* (2021) 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 (4.1) typically generates a transient response T_t before obtaining the desired steady-state solution, as shown in figure 1. The length of T_t affects the length of time stepping and the accuracy of the output, as discussed in §7.2.2. The discrete nature of time stepping encourages the usage of discrete Fourier transform (DFT) where $\hat{q}_s(\omega)$ 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.

In order to compute resolvent modes for all frequencies of interest

$$\Omega = \{0, \pm \omega_{min}, \pm 2\omega_{min}, \pm 3\omega_{min}, \dots, \pm \omega_{max}\},\tag{4.7}$$



Figure 2: Flowchart depicting the action of \mathbf{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 in computationally advantageous for large systems.

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

$$\boldsymbol{f} = \sum_{\omega_j \in \Omega} \hat{\boldsymbol{f}}_j e^{\mathrm{i}\omega_j t} \tag{4.8}$$

must include all frequencies in Ω . 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 (Nyquist, 1928). Performing time integration of (4.1) results in computing N_s steady-state snapshots within the T_s -period, where typically $N_s \ge N_{\omega}$, 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 $\mathbf{Q}_{N_s} = \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, ..., \mathbf{q}_{N_s}\}$ (analogous to the pink dots in figure 1), where \mathbf{q}_j represents the j^{th} steady-state snapshot in the time domain. The fast Fourier transform (FFT) can efficiently compute $\hat{\mathbf{Q}}_{N_s} = \{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3, ..., \hat{\mathbf{q}}_{N_s}\}$. However, the maximum resolved frequency within $\hat{\mathbf{Q}}_{N_s}$ 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 $\omega \in \Omega$ without aliasing is to consider N_{ω} equally spaced snapshots in $\mathbf{Q}_{N_{\omega}} = \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, ..., \mathbf{q}_{N_{\omega}}\}$ (analogous to the cyan dots in figure 1). Taking the FFT of $\mathbf{Q}_{N_{\omega}}$ yields $\hat{\mathbf{Q}}_{N_{\omega}} = \{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3, ..., \hat{\mathbf{q}}_{N_{\omega}}\}$, where each member $\hat{\mathbf{q}}_j$ represents the solution to $(i\omega_j \mathbf{I} - \mathbf{A})\hat{\mathbf{q}}_j = \hat{\mathbf{f}}_j$, with $\omega_j \in \Omega$.

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

$$T_s = dt \times N_s = \Delta t \times N_\omega. \tag{4.9}$$

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

$$\Delta t = \frac{T_s}{N_\omega} = \frac{2\pi/\omega_{min}}{2\left\lceil\frac{\omega_{max}}{\omega_{min}}\right\rceil} \tag{4.10}$$

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

Figure 2 demonstrates the equivalence between computing the action of \mathbf{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.

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 **R** and **R**^{*} circumvents the need for LU decomposition, improving the scaling of the algorithm (see §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

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

Algorithm 3: k, q, Ω are common parameters with RSVD. $(\cdot)_{\Omega}$ means the function is separately applied to each $\omega \in \Omega$, and TSS is an abbreviation for time-stepping schemes (*e.g.*, backward Euler)

As in the standard RSVD algorithm, the first step is to create random forcing matrices to sketch \mathbf{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 $\hat{\mathbf{Y}}$ of the resolvent operator \mathbf{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 $\hat{\mathbf{Q}}$ 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 $\hat{\mathbf{Q}}$ matrices in the time domain to sample the image of the resolvent operator \mathbf{R} for all $\omega \in \Omega$. Finally, the estimates of the k leading right singular vectors \mathbf{V} and gains $\boldsymbol{\Sigma}$ are obtained via an economy SVD of the $N \times k$ matrix $\hat{\mathbf{S}}$ (line 8), and left singular vectors \mathbf{U} are recovered in line 9.

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 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 **A** is constructed, the goal is to solve the linear system given by

$$(i\omega I - A)x = b \tag{6.1}$$

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

Table 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.

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

6.1 CPU cost

Direct solvers find the solution of (6.1) to machine precision. A common approach is to find the LU decomposition of $(i\omega I - A)$ 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 (Duff et al., 2017) and is often the dominant cost of solving a linear system (Marquet & Larsson, 2015). 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 (Amestov et al., 2019). 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 (Anderson et al., 1999), MUMPS (Amestov et al., 2001), PARDISO (Schenk et al., 2001), and Hypre (Falgout & Yang, 2002), 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 (Saad, 2003b), which can cause slow convergence and numerical stability issues for iterative solvers (Skeel, 1979). The LNS operator \mathbf{A} is typically a sparse but ill-conditioned matrix. When ω is small, $(i\omega \mathbf{I} - \mathbf{A})$ inherits the ill-conditioning of \mathbf{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 (Trefethen & Bau III, 1997).

In principle, iterative solvers are attractive when solving (6.1) 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 $(i\omega I - A)$, as explained above. One potential solution is the common practice of using a preconditioner (Saad, 2003*a*). 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 (Axelsson, 1985; Benzi, 2002; Pearson & Pestana, 2020). Despite numerous developments in this area, effective preconditioners do not exist for all matrices, including many LNS operators. Accordingly, direct methods/LU decompositions are almost always used to solve (6.1) when computing resolvent modes (Moarref *et al.*, 2013; Jeun *et al.*, 2016; Schmidt *et al.*, 2018; Ribeiro *et al.*, 2020).

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 (4.1)) has a long history and is a mature field (Hairer *et al.*, 1993; Wanner & Hairer, 1996; Trefethen & Bau III, 1997). 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

$$\mathcal{A}\boldsymbol{x} = \boldsymbol{b} \tag{6.2}$$

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 $\boldsymbol{\mathcal{A}} = c_1 \boldsymbol{I} + c_2 \boldsymbol{\mathcal{A}}$ for multi-step methods, where constants are determined based on integration scheme and time step, *e.g.*, $\boldsymbol{\mathcal{A}} = \boldsymbol{I} - dt \boldsymbol{\mathcal{A}}$ for backward Euler. A superficial comparison between (6.2) and (6.1) indicates that implicit time steppers suffer from the same issues elaborated above. However, the key difference is that $\boldsymbol{\mathcal{A}}$ is multiplied by the (small) time step dt, so the ill-conditioning of $\boldsymbol{\mathcal{A}}$ is largely overwhelmed by the ideal conditioning of the identity matrix \boldsymbol{I} . This improved conditioning makes possible the application of iterative solvers.

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 (6.1), 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 (Süli & Mayers, 2003). 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 \mathbf{A} with vectors during the time-stepping process, which scales like O(N). Generating forcing input and transforming responses to Fourier space are also O(N) operations (see §8.1.1). 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 §7.2.1. By fixing the time step and time-stepping scheme while varying N, it is evident that explicit integrators scale linearly with dimension. Implicit integrators, on the other hand, require at least one LU decomposition of \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).

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 (Davis *et al.*, 2016). Generally, direct solvers are more robust than iterative solvers but can consume significant memory due to the fill-in process of factorization (Marquet & Larsson, 2015). 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 (Towne *et al.*, 2022). 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 §8.1.1. On the other hand, implicit integration schemes, in addition to the Fourier space matrices, require memory for solving (6.2), 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.

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} (de Pando *et al.*, 2012; Martini *et al.*, 2021). 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.

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 **R**. By effectively addressing these sources of error, the RSVD- Δt method can be optimized for improved efficiency.

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.* (2011), 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.

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.

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 (Hairer *et al.*, 1993; Wanner & Hairer, 1996). 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

$$LTE = C \frac{d^p f(t)}{dt^p} O(dt^p), \tag{7.1}$$

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 $f(t) = \hat{f}e^{i\omega t}$. Substituting the forcing term into (7.1), we observe that

$$LTE \propto O((\omega dt)^p). \tag{7.2}$$

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} .

7.2.2 Transient error

The second source of time-stepping error arises from the unwanted transient response. The solution of (4.1) can be written as a sum of its transient and steady-state components,

$$\boldsymbol{q}(t) = \boldsymbol{q}_t(t) + \boldsymbol{q}_s(t), \tag{7.3}$$

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

$$\hat{\boldsymbol{q}}(\omega) = \hat{\boldsymbol{q}}_t(\omega) + \hat{\boldsymbol{q}}_s(\omega). \tag{7.4}$$

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 (Schmid, 2007) 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,

$$\epsilon = \frac{||\boldsymbol{q}_t||}{||\boldsymbol{q}_s||},\tag{7.5}$$

where the l^2 -norms can be replaced with $|| \cdot ||_q$ for non-identity weight matrices. In cases where we solve (4.1) with a zero initial condition (which is often the case), *i.e.*, $q(0) = q_t(0) + q_s(0) = 0$, the transient error is initially one,

$$\epsilon(0) = \frac{||\boldsymbol{q}_t(0)||}{||\boldsymbol{q}_s(0)||} = 1.$$
(7.6)

In the long term, the transient error approaches zero,

$$\lim_{t \to \infty} \epsilon(t) = \lim_{t \to \infty} \frac{||\boldsymbol{q}_t(t)||}{||\boldsymbol{q}_s(t)||} = 0,$$
(7.7)

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.



Figure 3: 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.

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

$$\frac{d\boldsymbol{q}_h}{dt} - \boldsymbol{A}\boldsymbol{q}_h = 0, \tag{7.8}$$

initialized with a random state (Eriksson & Rizzi, 1985; Edwards *et al.*, 1994). A random perturbation represents a worst-case scenario, as it excites all the slow modes of \mathbf{A} . By monitoring the norm of q_h over time, we can estimate the slowest decay rate, which corresponds to the real part of the least-damped eigenvalue of \mathbf{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 §8.2, we will present an efficient method to achieve a smaller transient error within a shorter time frame.

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 §6, are crucial in facilitating affordable resolvent analysis of complex three-dimensional flows.

8.1 Minimizing memory requirements

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

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 §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 3(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, 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 (2019). This procedure is shown schematically in figure 3(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,

$$\boldsymbol{f}_p = \sum_{s=1}^{N_\omega} \boldsymbol{Z}'_{ps} \hat{\boldsymbol{f}}_s, \tag{8.1}$$

where $\mathbf{Z}'_{ps} = exp(-2\pi i/N_s)^{(p-1)(s-1)}$. The integer p $(1 \le p \le N_s)$ specifies the phase of the periodic forcing at the current time step. Here, $\hat{f}_s \in \mathbb{C}^{N \times k \times N_\omega}$ denotes Fourier modes that are accessible from memory. The sum is taken over every $\omega \in \Omega$, and it outputs the p^{th} time domain snapshot $f_p \in \mathbb{C}^{N \times k}$. 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 q_l within the steady-state response contributes to each each Fourier mode according to the partial sum

$$[\hat{\boldsymbol{q}}_s]_r = [\hat{\boldsymbol{q}}_s]_{r-1} + \boldsymbol{Z}_{ls} \boldsymbol{q}_r = \sum_{l=1}^r \boldsymbol{Z}_{ls} \boldsymbol{q}_l, \qquad (8.2)$$

where $\mathbf{Z}_{ls} = exp(-2\pi i/N_{\omega})^{(l-1)(s-1)}, 1 \leq (l,s) \leq N_{\omega}$. Here, $[\hat{\mathbf{q}}_s]_r$ represents the sum of contributions up to \mathbf{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 $\mathbf{Z}' \in \mathbb{C}^{N_{\omega} \times N_s}$ and the DFT matrix $\mathbf{Z} \in \mathbb{C}^{N_{\omega} \times N_{\omega}}$ is their sizes: \mathbf{Z} is used to generate N_s temporal snapshots of the forcing from N_{ω}

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

Table 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.

Fourier modes, while \mathbf{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. 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.

8.1.2 Optimal cost for real-valued matrices

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

$$\boldsymbol{R}_{-\omega} = (-\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1} = (\mathrm{i}\omega\overline{\boldsymbol{I}} - \overline{\boldsymbol{A}})^{-1} = \overline{(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}} = \overline{\boldsymbol{R}}_{\omega} = \overline{\boldsymbol{U}}\boldsymbol{\Sigma}\overline{\boldsymbol{V}}^*, \tag{8.3}$$

where (\cdot) denotes the complex conjugate and $\mathbf{A} = \overline{\mathbf{A}}$ when \mathbf{A} is real-valued. Equation (8.3) 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 §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.

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.

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 (4.1), as discussed in §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 (4.1), $q_1 = q(t_1)$ and $q_2 = q(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts, as in (7.3), as

$$\begin{aligned}
 q_1 &= q_{s,1} + q_{t,1}, \\
 q_2 &= q_{s,2} + q_{t,2},
 \end{aligned}$$
(8.4)

where $q_{s,1}, q_{s,2}, q_{t,1}$, and $q_{t,2}$ are four unknowns. Applying a prescribed forcing in (4.1) at a single frequency ω yields

$$\boldsymbol{q}_{s,2} = \boldsymbol{q}_{s,1} e^{\mathrm{i}\omega\Delta t}.$$
(8.5)

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

$$\boldsymbol{q}_{t,2} = e^{\boldsymbol{A}\Delta t} \boldsymbol{q}_{t,1}. \tag{8.6}$$

Simplifying (8.4), (8.5), and (8.6) for $q_{t,1}$, we obtain

$$(\mathbf{I} - e^{-i\omega\Delta t}e^{\mathbf{A}\Delta t})\mathbf{q}_{t,1} = \mathbf{b},$$
(8.7)

where $\mathbf{b} = \mathbf{q}_1 - \mathbf{q}_2 e^{-i\omega\Delta t}$ is known from the time-stepping solution. Equation (8.7) holds for any two points in time with arbitrary separation Δt . The exact steady-state solution with no transient error is obtained by solving (8.7) for $\mathbf{q}_{t,1}$ and using (8.4) to obtain $\mathbf{q}_{s,1} = \mathbf{q}_1 - \mathbf{q}_{t,1}$.

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 (8.7) for all $\omega \in \Omega$.

Solving (8.7) can be computationally expensive, particularly for large systems, even if we assume that computing $e^{\mathbf{A}\Delta t}$ is feasible. To address this issue, one possible approach is to choose a small Δt and expand the exponential term as $e^{\mathbf{A}\Delta t} = \sum_{j} \frac{(\mathbf{A}\Delta t)^{j}}{j!}$. However, this leads to solving a similar linear system to (6.1), 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 (8.7).

Consider a low-dimensional representation of the transient response as

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}\boldsymbol{\beta}_1, \tag{8.8}$$

where $\phi \in \mathbb{C}^{N \times r}$, with $r \ll N$, is an orthonormal test basis spanning the transient response and $\beta_1 \in \mathbb{C}^r$ represents the coefficients describing the transient in this basis. By substituting (8.8) into (8.7), the linear system

$$(\mathbf{I} - e^{-i\omega\Delta t}e^{\mathbf{A}\Delta t})\boldsymbol{\phi}\boldsymbol{\beta}_1 = \mathbf{b}$$
(8.9)

is overdetermined. Petrov-Galerkin projection with trial basis $\psi \in \mathbb{C}^{N \times r}$ is employed to close (8.9), giving

$$\boldsymbol{\psi}^*(\boldsymbol{I} - e^{-i\omega\Delta t}e^{\boldsymbol{A}\Delta t})\boldsymbol{\phi}\boldsymbol{\beta}_1 = \boldsymbol{\psi}^*\boldsymbol{b}.$$
(8.10)

Solving (8.10) for β and inserting the solution into (8.8) yields

$$\boldsymbol{q}_{t,1} = \boldsymbol{\phi}(\boldsymbol{\psi}^* \boldsymbol{\phi} - e^{-\mathrm{i}\omega\Delta t} \tilde{\boldsymbol{M}})^{-1} \boldsymbol{\psi}^* \boldsymbol{b}, \qquad (8.11)$$

where

$$\tilde{\boldsymbol{M}} = \boldsymbol{\psi}^* e^{\boldsymbol{A} \Delta t} \boldsymbol{\phi} \in \mathbb{C}^{r \times r}$$
(8.12)

is a reduced matrix that maps the coefficients. The advantage of this strategy is that it allows for the computation of the inverse of $(\boldsymbol{\psi}^*\boldsymbol{\phi} - e^{-i\omega\Delta t}\tilde{\boldsymbol{M}})$ 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.*, ϕ is also the trial 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 ϕ 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 test bases, we note that one can construct a new ϕ for each $\omega \in \Omega$, however, the bases that we define later are universal for all frequencies. Hence, the reduced matrix \tilde{M} is constructed once for all frequencies. Subsequently, (8.11) 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 \mathbf{A} as the chosen test basis. By excluding the least-damped eigenvectors, we effectively increase the decay rate of the transient response. Let λ_1 denote the least-damped eigenvectors, we effectively increase the decay rate of the transient response. Let λ_1 denote the least-damped eigenvalue of \mathbf{A} , with \mathbf{V}_1 representing the corresponding eigenvector. We define $\phi = \mathbf{V}_1$, thereby removing the transient component projected onto \mathbf{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 eigenvectors, $\phi = \operatorname{orth}\{\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_{r-1}\}$, leading to a decay rate governed by the r^{th} least-damped eigenvalue of \mathbf{A} . For this particular test 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 test 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.*, Krylovbased methods (Eriksson & Rizzi, 1985; Edwards *et al.*, 1994). Overall, the least-damped modes of \boldsymbol{A} are most helpful for systems that suffer from only a few slowly decaying modes.

Another powerful test basis is formed by stacking the snapshots into a matrix during the integration of the LNS equations, resulting in $\phi = \operatorname{orth}\{q_1, q_2, q_3, ..., q_r\}$ (an orthogonalization of the matrix of snapshots). Specifically, ϕ can be constructed as the union of $\hat{\boldsymbol{Q}}$ and $\hat{\boldsymbol{Q}}^{\Delta t}$ as a reliable test basis. Performing QR decomposition on this matrix is essential to ensure orthogonality. As the LNS equations are allowed to run for a longer duration, ϕ becomes an increasingly effective test 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 test bases. For instance, by considering the matrix of least-damped eigenvectors of \mathbf{A} in ϕ_1 and the on-the-fly snapshots in ϕ_2 , a combined test basis $\phi = \phi_1 \cup \phi_2$ can be constructed and orthogonalized. The combination of test bases, with ϕ_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 (7.8) using a random initial condition with unit norm. By employing (8.4), (8.5), and (8.6), and assuming $q_s = 0$, we can apply either Petrov-Galerkin or Galerkin projection to calculate the updated transient norms. This approach is feasible when ϕ does not depend on real-time simulation, such as when it represents the matrix of least-damped eigenvectors. However, if ϕ consists of snapshots, we must generate synthetic snapshots. To accomplish this, we set ${m q}_{s,0}=-{m q}_{t,0}$ to ensure the initial snapshot $q_0 = q_{s,0} + q_{t,0}$ equals zero. Subsequent snapshots are obtained by superimposing the transient responses (from the homogeneous simulation) onto steady-state responses generated as $q_{s,j} = e^{i\omega j\Delta t} q_{s,1}$, where Δt is the time-distance between snapshots. Using this technique, we can construct ϕ for varying periods and assess the efficacy of the transient removal strategy. The updated transient error, similar to (7.5), 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 test bases in $\S9$.

9 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.

9.1 Complex Ginzburg-Landau equation

The complex Ginzburg-Landau equation was initially derived for analytical studies of Poiseuille flow (Stewartson & Stuart, 1971) and has subsequently been used more generally as a convenient



Figure 4: Relative error between gains computed using the RSVD-LU and RSVD- Δt algorithms for the Ginzburg-Landau problem: (a) $T_t = 5000$ and {TSS, dt} = {BDF4, 0.1} (purple), {BDF4, 0.01} (red), (BDF6, 0.01) (green), and {BDF6, 0.001} (blue) varies; (b) {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 $\omega \in [0.6, 4]$.

model of a flow susceptible to non-modal amplification (Hunt & Crighton, 1991; Bagheri *et al.*, 2009; Chen & Rowley, 2011; Cavalieri *et al.*, 2019). 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

$$\mathbf{A} = -\nu \frac{\partial}{\partial x} + \gamma \frac{\partial^2}{\partial x^2} + \mu(x),$$

$$\mu(x) = (\mu_0 - c_{\mu}^2) + \frac{\mu_2}{2} x^2,$$

$$\mathbf{B} = \mathbf{C} = \mathbf{I}.$$
(9.1)

Following Bagheri *et al.* (2009), we set $\gamma = 1 - i$, $\nu = 2 + 0.2i$, $\mu_0 = 0.38$, $c_{\mu} = 0.2$, 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 (Bagheri *et al.*, 2009). 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 (Bagheri *et al.*, 2009), and the weight matrix \boldsymbol{W} is set to the identity on account of the uniform grid.

9.1.1 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 §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.

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



Figure 5: 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 ϕ of V_1 , $\{V_1, V_2\}$, 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)).

and the relative errors for various cases are shown in figure 4. 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 4(a), the transient length is held fixed at $T_t = 5000$ and the gains are obtained using {BDF6, $dt = 10^{-2}$ }, {BDF4, $dt = 10^{-2}$ }, and {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 §7.2.

Figure 4(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 {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 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 §9.2, the peak transient for the jet flows occurs near zero frequency.

9.1.2 Efficient transient removal

In this section, we demonstrate the transient removal strategy proposed in §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 5(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 6: 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.

Figure 5(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 (4.1) 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 5(b). By defining ϕ_1 as V_1 and utilizing Galerkin projection, we remove the fraction of the transient decaying at the rate of $e^{\lambda_1 t}$, resulting in a noticeable change in the decay slope. Including the two least-damped modes with $\phi_2 = \{V_1, V_2\}$ further steepens the decay rate, aligning closely with the corresponding least-damped eigenvalues shown in figure 5(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 {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 ϕ_1 , and (iii) Galerkin projection with ϕ_2 . The error is measured as the relative difference in gain between the RSVD-LU and RSVD- Δt algorithms, as depicted in figure 5(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.

9.1.3 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,



Figure 7: The mean streamwise velocity of the axisymmetric jet, 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.

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 5(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 6. 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.

9.2 Round turbulent jet

Second, 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 (Brès *et al.*, 2017, 2018), 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.* (2018).

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 7. 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,



Figure 8: Three leading gains of the axisymmetric jet for four azimuthal wavenumbers.

where $St = \frac{\omega D_j}{2\pi U_j}$ is the non-dimensional form of frequency. The range of $St \in [0, 1]$ is wide enough to include the most important physical phenomena captured by resolvent analysis (Schmidt *et al.*, 2018). The effective *Re* is reduced to 1000 to account for un-modeled Reynolds stresses (Pickering *et al.*, 2021) and the effect of *Re* is thoroughly investigated and reported in Schmidt *et al.* (2017).

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

$$\boldsymbol{q}'(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, t) = \sum_{\omega} \hat{\boldsymbol{q}}_{\omega}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) e^{\mathrm{i}\omega t}$$
(9.2)

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

$$\boldsymbol{q}'(\boldsymbol{x},\boldsymbol{r},\boldsymbol{\theta},t) = \sum_{m,\omega} \hat{\boldsymbol{q}}_{m,\omega}(\boldsymbol{x},\boldsymbol{r}) e^{\mathrm{i}\boldsymbol{m}\boldsymbol{\theta}} e^{\mathrm{i}\omega t}$$
(9.3)

is characterized by the pair (m, ω) , where m denotes azimuthal wavenumber. The domain of interest for resolvent analysis is $x \times r \in [0, 20] \times [0, 4]$ surrounded by a sponge region which is spatially discretized using fourth-order summation by parts finite differences (Mattsson & Nordström, 2004) 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 $0 \le St \le 1$. The remaining parameters are kept the same as in the threedimensional discretization of the jet.

Figure 8 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.* (2017) and Pickering *et al.* (2021). The optimal mode when $m = 0, St \ge 0.2$ corresponds to Kelvin-Helmholtz (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.* (2018) and Pickering *et al.* (2020) 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



Figure 9: 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.

gains and corresponding modes of the axisymmetric jet are used as a baseline for comparison to the three-dimensional jet.

9.2.1 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 4th order Runge–Kutta (RK4) integrator with dt = 0.00625. The steady-state interval is $T_s = 20$. Figure 9 shows the expected transient error in the time and frequency domains. The transient initially grows in time before slowly decaying in figure 9(a). The resulting error in the frequency domain obtained from selecting each colored segment for computing resolvent modes is shown in figure 9(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 10 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 (Sirovich, 1987b). Since higher wavenumbers (m > 3) have lower gains (Pickering *et al.*, 2021), 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



Figure 10: 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).

the grid for the compact structures that appear at these frequencies.

Figure 11 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 threedimensional 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.* (2021), symmetries such as the azimuthal homogeneity of the jet produce pairs of 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.

9.2.2 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. 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 §9.2.1. 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 (6.1). The reported costs encompasses the entire RSVD- Δt algorithm, including one extra period of time-stepping duration to account for the transient removal strategy, as explained in §8.2. The RSVD- Δt algorithm is implemented using PETSc (Balay *et al.*, 2019), while the LU decomposition in the RSVD-LU algorithm utilizes PETSc in conjunction with the MUMPS (Amestoy *et al.*, 2001) 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 12(a) as a function of the state



Figure 11: 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.



Figure 12: 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.

dimension N. The RSVD-LU algorithm scales poorly, in fact exceeding the theoretical scaling of $O(N^2)$ for three-dimensional flows (refer to §6) due to poor performance at low frequencies that has also been noted in other studies (Pickering *et al.*, 2020). In contrast, the RSVD- Δt algorithm achieves (near) linear scaling, $O(N^{1.1})$, confirming its scalability to large problems.

Similar observations can be made about the memory requirements of the two algorithms, shown in figure 12(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 §6.

The range of N in figure 12 was selected to make the scaling study tractable for the RSVD-LU algorithm, but the corresponding grids are under-resolved. Table 3 compares the costs of RSVD-LU and RSVD- Δt for a more realistic state dimension $N \approx 39$ million (5 state variables × a [400×140²] grid), which was used for the three-dimensional calculations in §9.2.2, 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 12. On the other hand, for RSVD- Δt , the CPU time and memory usage are directly taken from our simulation, which employed 300 parallel cores. 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 12. This implies that for q = 1, the CPU time includes a single LU decomposition and three times solving the LU-decomposed system.

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. 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 3 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

Algorithm		CPU time (hours)		Memory (GB)
	Total	Action of \pmb{R} and \pmb{R}^*	SVD/QR	
RSVD-LU RSVD- Δt	$\begin{array}{c} 7.53\times10^7 \\ 1.83\times10^4 \end{array}$	7.53×10^{7} 1.83×10^{4}	$0.762 \\ 0.762$	$\begin{array}{c} 1.33\times10^5\\ 7.36\times10^2\end{array}$

Table 3: 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 **R** and **R**^{*} use time stepping for RSVD- Δt and a direct solver for the RSVD-LU algorithm.

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 §8.1.3, while no such trade-off exists for the RSVD-LU algorithm.

10 Application: jet with streaks

Finally, we apply the RSVD- Δt algorithm to study the impact of streaks on other coherent structures within a turbulent jet. This is a fully three-dimensional problem for which results obtained using other algorithms are not available.

Streaks – elongated regions of low-velocity fluid – have historically been observed and studied in turbulent channel flows (see McKeon (2017) and Jiménez (2018) 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 (Jimenez-Gonzalez & Brancher, 2017). Nogueira *et al.* (2019) and Pickering *et al.* (2020) showed that streaks emerge as the dominant structures in the SPOD and resolvent spectra of jets at very low 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 (Marant & Cossu, 2018) and Tollmien–Schlichting waves in the Blasius boundary layer (Cossu & Brandt, 2002). Similar findings on a high-speed turbulent jet by Wang *et al.* (2021) 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 Kelvin-Helmholtz wavepackets described by the leading resolvent modes via a secondary stability analysis.

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 (Pickering *et al.*, 2020). 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.* (2021).

The updated mean flow with the streaks added has 6-fold rotational symmetry and, following Sinha *et al.* (2016), can be written as

$$\bar{\boldsymbol{q}}(x,r,\theta) = \sum_{j=-\infty}^{\infty} \hat{\boldsymbol{q}}_{lj}(x,r)e^{ilj\theta}.$$
(10.1)

They proved that after plugging the Fourier ansatz of the resulting mean flow into the LNS equations, given an azimuthal wavenumber m, the associated axisymmetric mode $\hat{q}_{m,\omega}$ can only couple with $\hat{q}_{m-lj,\omega}$ 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 $q_{\omega}^{L} = \{\hat{q}_{L-lj,\omega}\}_{l=-\infty}^{l=\infty}$, 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 $L = \pm 1, \pm 2$ 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. (2021) 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. (2019).

Once the axisymmetric streaks are computed, the three-dimensional streaks are obtained by revolving them around the x-axis with phase $e^{il\theta}$ (see figure 7). 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.* (2021), 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 paper 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 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 §9.2. The time-stepping scheme is RK4 with dt = 0.00625. Transient errors are held below 1% for St > 0 via 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 13(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



Figure 13: 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).

the leading gain is maximum, are shown in figure 13. 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 13(f). This plot shows that the optimal mode is comprised of L = 3 with a larger weight and L + l = 3 + 6 = 9with 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. Utilizing the RSVD-LU algorithm for the same grid with state dimension $N \approx 39$ million, the anticipated CPU time surpasses 75 million hours, as discussed in §9.2.2. 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.

11 Conclusions

This paper 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 $(i\omega I - A)$ 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 $(i\omega I - A)$ 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 errors, 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.

Lastly, we leveraged the novel capabilities of the RSVD- Δt algorithm to investigate the influence of streaks within the turbulent jet on the KH wavepackets. 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.

Our algorithm also has several implementation advantages. Our time-stepping approach enables matrix-free implementation, eliminating the explicit formation of the LNS matrix \mathbf{A} , instead directly utilizing built-in linear direct and adjoint capabilities available within many existing codes. All operations within our the RSVD- Δt algorithm are amenable to efficient parallelization; we have optimized out implementation of the algorithm for parallel computing using the PETSc (Balay *et al.*, 2019) and SLEPc (Hernandez *et al.*, 2005) 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.* (2020) and House *et al.* (2022) can be integrated into our approach to use physical insight to select the initial test vectors instead of relying on entirely random ones.

Acknowledgements

We would like to express our gratitude to André Cavalieri for his invaluable feedback, insights, and contributions. We also acknowledge the University of Michigan's Great Lakes cluster for providing the essential computational resources that enabled us to conduct all computations for this research. A.F. and A.T. gratefully acknowledge funding for this work from the Michigan Institute for Computational Discovery and Engineering (MICDE) and AFOSR award number FA9550-20-1-0214.

Appendix A RSVD- Δt for the weighted resolvent operator

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 (see §2). In this appendix, 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 (Towne *et al.*, 2018)

$$\tilde{\boldsymbol{R}} = \boldsymbol{W}_{q}^{1/2} \boldsymbol{C} (\mathrm{i}\omega \boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{B} \boldsymbol{W}_{f}^{-1/2}, \qquad (A.1a)$$

$$\tilde{\boldsymbol{R}} = \tilde{\boldsymbol{U}} \boldsymbol{\Sigma} \tilde{\boldsymbol{V}}^*, \tag{A.1b}$$

and further

$$U = W_q^{-1/2} U,$$

$$V = W_f^{-1/2} \tilde{V},$$
(A.2)

where Σ contains the gains, and V and U are forcing and response modes, respectively. The resolvent operator is recovered as

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\boldsymbol{W}_f. \tag{A.3}$$

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

$$\hat{\boldsymbol{y}} = \tilde{\boldsymbol{R}}\hat{\boldsymbol{f}} = \boldsymbol{W}_q^{1/2}\boldsymbol{C}(\mathrm{i}\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{W}_f^{-1/2}\hat{\boldsymbol{f}}$$
(A.4)

for all $\omega \in \Omega$ using time stepping. The process begins by computing the product between $\hat{f}_W = W_f^{-1/2} \hat{f}$ in Fourier space, followed by $\hat{f}_{W,B} = B\hat{f}_W$. 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 $\hat{F}_{W,B}$. Subsequently, the action of $(i\omega I - A)^{-1}$ is computed on $\hat{F}_{W,B}$ using time stepping to yield \hat{Y} . The resulting output undergoes $\hat{y}_C = C\hat{y}$ and $\hat{y}_{C,W} = W_q^{1/2}\hat{y}_C$, which are repeated for all frequencies to obtain $\hat{Y}_{C,W}$. Figure 14 visually illustrates the order of calculations for R in the top row and \tilde{R} in the bottom row. An analogous process is utilized to compute the action of \tilde{R}^* .

Appendix B Removing the least-damped modes using eigenvalues only

The transient removal strategies described in $\S8.2$ require a basis for the transient, either in the form of eigenvectors for the least-damped eigenvalues or data. In this appendix, we outline an

$$\hat{\boldsymbol{F}} \xrightarrow{\text{Input}} \text{Time stepping} \xrightarrow{\text{Output}} \hat{\boldsymbol{Y}}$$

$$\hat{\boldsymbol{F}} \xrightarrow{\text{Weighted}} \hat{\boldsymbol{F}}_{W} \xrightarrow{\text{Filtered}} \hat{\boldsymbol{F}}_{W,B} \xrightarrow{\text{Input}} \text{Time stepping} \xrightarrow{\text{Output}} \hat{\boldsymbol{Y}} \xrightarrow{\text{Filtered}} \hat{\boldsymbol{Y}}_{C} \xrightarrow{\text{Weighted}} \hat{\boldsymbol{Y}}_{C,W}$$

Figure 14: The schematic of computing the action of \boldsymbol{R} on top and the action of \boldsymbol{R} on the bottom row.

alternative procedure to expedite the decay of transients that that uses knowledge of the leastdamped eigenvalues themselves. Considering two solutions of (4.1), $q_1 = q(t_1)$ and $q_2 = q(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts as

$$q_1 = q_{s,1} + q_{t,1},$$

 $q_2 = q_{s,2} + q_{t,2},$
(B.1)

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

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

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

$$\boldsymbol{q}_{\lambda_1,2} = \boldsymbol{q}_{\lambda_1,1} e^{\lambda_1 \Delta t},\tag{B.3}$$

where $q_{\lambda_1,j}$ is essentially the projection of the transient response onto the least-damped eigenmode of **A** at $t = t_j$. The steady-state evolution at a prescribed forcing at a single frequency ω follows (8.5). Therefore, in case of $||q_{rest,j}|| = 0$, the system of equations is deterministic and $q_{t,1}$ can be found as

$$\boldsymbol{q}_{t,1} = \frac{\boldsymbol{b}}{c},\tag{B.4}$$

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

$$\boldsymbol{q}_{t,1} = \frac{\boldsymbol{b}}{c} - \frac{(1-c)\boldsymbol{q}_{rest,1} - \boldsymbol{q}_{rest,2}e^{-\mathrm{i}\omega\Delta t}}{c}.$$
(B.5)

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 $\frac{b}{c}$, 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 $\frac{b}{c}$ 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

$$\boldsymbol{q}_{l} = \boldsymbol{q}_{s,l} + \sum_{j=1}^{d} \boldsymbol{q}_{\lambda_{j},l} + \boldsymbol{q}_{rest,l}, \qquad (B.6)$$

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.

References

- 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.
- 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.
- 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.
- 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.
- AXELSSON, O. 1985 A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT Numerical Mathematics* **25** (1), 165–187.
- 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).
- 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.
- BARTHEL, B., GOMEZ, S. & MCKEON, B. J. 2022 Variational formulation of resolvent analysis. *Physical Review Fluids* 7 (1), 013905.
- BENZI, M. 2002 Preconditioning techniques for large linear systems: a survey. Journal of Computational Physics 182 (2), 418–477.
- 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.
- 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 boundarylayer state in subsonic turbulent jets. *Journal of Fluid Mechanics* 851, 83–124.
- 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.
- CAVALIERI, A. V. G., JORDAN, P. & LESSHAFFT, L. 2019 Wave-packet models for jet dynamics and sound radiation. *Applied Mechanics Reviews* **71** (2).
- CHAVARIN, A. & LUHAR, M. 2020 Resolvent analysis for turbulent channel flow with riblets. *AIAA Journal* 58 (2), 589–599.
- CHEN, 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.
- 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.
- 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.

- DAVIS, T. A., RAJAMANICKAM, S. & SID-LAKHDAR, W. M. 2016 A survey of direct methods for sparse linear systems. Acta Numerica 25, 383–566.
- 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.
- DUFF, I. S., ERISMAN, A. M. & REID, J. K. 2017 Direct methods for sparse matrices. Oxford University Press.
- DUNFORD, N. & SCHWARTZ, J. T. 1958 Linear operators part I: general theory. John Wiley & Sons.
- 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.
- 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.
- FALGOUT, R. D. & YANG, U. M. 2002 hypre: A library of high performance preconditioners. In International Conference on Computational Science, pp. 632–641.
- GÓMEZ, F., SHARMA, A. S. & BLACKBURN, H. M. 2016 Estimation of unsteady aerodynamic forces using pointwise velocity data. *Journal of Fluid Mechanics* 804, R4.
- HAIRER, E., NØRSETT, S.P. & WANNER, G. 1993 Solving ordinary differential equations I: nonstiff problems. Springer Berlin Heidelberg.
- 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.
- 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.
- HERRMANN, B., BADDOO, P. J., SEMAAN, R., BRUNTON, S. L. & MCKEON, B. J. 2021 Datadriven resolvent analysis. *Journal of Fluid Mechanics* **918**, A10.
- HOUSE, D., SKENE, C., RIBEIRO, J. H. M., YEH, C.-A. & TAIRA, K. 2022 Sketch-based resolvent analysis. AIAA Paper #2022-3335.
- 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.
- 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.
- JIMÉNEZ, J. 2018 Coherent structures in wall-bounded turbulence. *Journal of Fluid Mechanics* 842, P1.
- JIMENEZ-GONZALEZ, J. I. & BRANCHER, P. 2017 Transient energy growth of optimal streaks in parallel round jets. *Physics of Fluids* **29** (11), 114101.
- JOVANOVIC, M. R. 2004 Modeling, analysis, and control of spatially distributed systems. University of California, Santa Barbara.
- 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.
- KAMAL, O., LAKEBRINK, M. T. & COLONIUS, T. 2023 Global receptivity analysis: physically realizable input-output analysis. *Journal of Fluid Mechanics* **956**, R5.

- 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.
- KATO, T. 2013 Perturbation theory for linear operators. Springer Science & Business Media.
- 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.
- LI, F. & MALIK, M. R. 1996 On the nature of PSE approximation. Theoretical and Computational Fluid Dynamics 8 (4), 253–273.
- LUMLEY, J. L. 1967 The structure of inhomogeneous turbulent flows. Atmospheric Turbulence and Radio Wave Propagation pp. 166–178.
- MARANT, M. & COSSU, C. 2018 Influence of optimally amplified streamwise streaks on the Kelvin– Helmholtz instability. *Journal of Fluid Mechanics* 838, 478–500.
- MARQUET, O. & LARSSON, M. 2015 Global wake instabilities of low aspect-ratio flat-plates. European Journal of Mechanics-B/Fluids 49, 400–412.
- MARTINI, E., CAVALIERI, A. V. G., JORDAN, P., TOWNE, A. & LESSHAFFT, L. 2020 Resolventbased optimal estimation of transitional and turbulent flows. *Journal of Fluid Mechanics* **900**, A2.
- MARTINI, E., JUNG, J., CAVALIERI, A. V. G., JORDAN, P. & TOWNE, A. 2022 Resolventbased tools for optimal estimation and control via the Wiener-Hopf formalism. *Journal of Fluid Mechanics* **938**, E2.
- MARTINI, E., RODRÍGUEZ, D., TOWNE, A. & CAVALIERI, A. V. G. 2021 Efficient computation of global resolvent modes. *Journal of Fluid Mechanics* **919**, A3.
- MATTSSON, K. & NORDSTRÖM, J. 2004 Summation by parts operators for finite difference approximations of second derivatives. *Journal of Computational Physics* **199** (2), 503–540.
- MCKEON, B. J. 2017 The engine behind (wall) turbulence: perspectives on scale interactions. Journal of Fluid Mechanics 817, P1.
- MCKEON, B. J. & SHARMA, A. S. 2010 A critical-layer framework for turbulent pipe flow. *Journal* of Fluid Mechanics 658, 336–382.
- 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.
- MONOKROUSOS, A., ÅKERVIK, E., BRANDT, L. & HENNINGSON, D. S. 2010 Global threedimensional optimal disturbances in the Blasius boundary-layer flow using time-steppers. *Journal* of Fluid Mechanics 650, 181–214.
- 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.
- 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.
- NYQUIST, H. 1928 Certain topics in telegraph transmission theory. Transactions of the American Institute of Electrical Engineers 47 (2), 617–644.
- 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.

- PEARSON, J. W. & PESTANA, J. 2020 Preconditioners for krylov subspace methods: An overview. GAMM-Mitteilungen 43 (4), e202000015.
- 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.
- 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.
- RAN, W., ZARE, A. & JOVANOVIĆ, M. R. 2021 Model-based design of riblets for turbulent drag reduction. *Journal of Fluid Mechanics* **906**, A7.
- RIBEIRO, J. H. M., YEH, C.-A. & TAIRA, K. 2020 Randomized resolvent analysis. *Physical Review Fluids* 5 (3), 033902.
- SAAD, Y. 2003a Finding exact and approximate block structures for ilu preconditioning. SIAM Journal on Scientific Computing 24 (4), 1107–1123.
- SAAD, Y. 2003b Iterative methods for sparse linear systems. SIAM.
- 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.
- 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.
- SCHMID, P. J. 2007 Nonmodal stability theory. Annual Review of Fluid Mechanics **39** (1), 129–162.
- SCHMID, P. J. 2010 Dynamic mode decomposition of numerical and experimental data. Journal of Fluid Mechanics 656, 5–28.
- SCHMID, P. J. 2022 Dynamic mode decomposition and its variants. Annual Review of Fluid Mechanics 54, 225–254.
- SCHMID, P. J. & HENNINGSON, D. S. 2001 Stability and transition in shear flows. Springer, New York.
- SCHMIDT, O. T. & TOWNE, A. 2019 An efficient streaming algorithm for spectral proper orthogonal decomposition. *Computer Physics Communications* 237, 98–109.
- 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.
- 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.
- SINHA, A., GUDMUNDSSON, K., XIA, H. & COLONIUS, T. 2016 Parabolized stability analysis of jets from serrated nozzles. *Journal of Fluid Mechanics* **789**, 36–63.
- 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.
- SIROVICH, L. 1987a Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of Applied Mathematics* **45** (3), 561–571.
- SIROVICH, L. 1987b Turbulence and the dynamics of coherent structures. ii. symmetries and transformations. *Quarterly of Applied Mathematics* **45** (3), 573–582.

- SKEEL, R. D. 1979 Scaling for numerical stability in Gaussian elimination. Journal of the ACM (JACM) 26 (3), 494–526.
- STEWART, G. W. 1993 On the early history of the singular value decomposition. *SIAM review* **35** (4), 551–566.
- 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.
- SÜLI, E. & MAYERS, D. F. 2003 An introduction to numerical analysis. Cambridge university press.
- 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.
- 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.
- THEOFILIS, V. 2011 Global linear instability. Annual Review of Fluid Mechanics 43, 319–352.
- 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.
- 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.
- TOWNE, A. & COLONIUS, T. 2015 One-way spatial integration of hyperbolic equations. *Journal* of Computational Physics **300**, 844–861.
- 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*.
- TOWNE, A., LOZANO-DURÁN, A. & YANG, X. 2020 Resolvent-based estimation of space-time flow statistics. *Journal of Fluid Mechanics* 883, A17.
- TOWNE, A., RIGAS, G. & COLONIUS, T. 2019 A critical assessment of the parabolized stability equations. *Theoretical and Computational Fluid Dynamics* **33**, 359–382.
- 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.
- 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.
- TREFETHEN, L. N. & BAU III, D. 1997 Numerical linear algebra. Siam.
- VERSHYNIN, R. 2018 High-dimensional probability: An introduction with applications in data science. Cambridge University Press.
- 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.
- WANNER, G. & HAIRER, E. 1996 Solving ordinary differential equations II: stiff and differentialalgebraic problems. Springer Berlin Heidelberg.
- 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.
- YEH, C.-A. & TAIRA, K. 2019 Resolvent-analysis-based design of airfoil separation control. Journal of Fluid Mechanics 867, 572–610.

ZHU, M. & TOWNE, A. 2023 Recursive one-way Navier-Stokes equations with PSE-like cost. *Journal of Computational Physics* 473, 111744.