


Scalable resolvent analysis for three-dimensional flows

Ali Farghadan^{a, }, Eduardo Martini^{b, }, Aaron Towne^{a, },^{*}

^a University of Michigan, Department of Mechanical Engineering, Ann Arbor, 48109, MI, USA

^b Institut Pprime, CNRS, Université de Poitiers ISAE-ENSMA, Poitiers, France

ARTICLE INFO

Keywords:

Resolvent analysis

Algorithm

Randomized singular value decomposition

Time stepping

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. We develop additional strategies to minimize these costs and control errors. We validate the algorithm using a Ginzburg-Landau test problem and demonstrate RSVD- Δt 's 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, though chaotic, often exhibit recurring patterns that are essential to their dynamics. For instance, near-wall streaks affect both momentum transfer and energy dissipation [116], while large-scale motions contribute to turbulence by shaping the energy cascade [40], and hairpin vortices play a key role in energy and momentum transport within turbulent boundary layers [2]. In free-shear flows, coherent structures arising from the Kelvin-Helmholtz instability, Orr mechanism, and lift-up mechanism have been observed in free jets, where they are integral to energy transfer and noise generation [44,69]. Coherent structures are also critical to the transition from laminar to turbulent flow [83] and to sustaining turbulence [60]. Popular data-driven methods include proper orthogonal decomposition (POD) [89], dynamic mode decomposition (DMD) [81], and spectral proper orthogonal decomposition (SPOD) [52,107]. In particular, SPOD identifies energy-ranked, single-frequency structures that evolve coherently in space and time.

Resolvent (or input-output) analysis originates from classical control theory [27,49] and has become arguably the most important operator-theoretic modal decomposition technique in fluid mechanics [61,98,46]. Resolvent analysis has been applied to a wide variety of flows, including canonical wall-bounded flows [25,64], turbulent jets [41,86,50,69], and airfoils [100,114]. It has been

* Corresponding author.

E-mail address: towne@umich.edu (A. Towne).

<https://doi.org/10.1016/j.jcp.2024.113695>

Received 7 May 2024; Received in revised form 16 December 2024; Accepted 17 December 2024

used for diverse tasks including design optimization [18,71], receptivity analysis [47,22], and flow control [115,104,56,57]. 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 [61,107]. Specifically, the resolvent modes associated with the largest singular values can provide an approximation of the leading SPOD modes [107] and, in some cases, capture the majority of the power spectral density (PSD) of the flow [97]. 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, and offer information about the mechanisms driving these responses. The singular values, referred to as the gains, quantify the amplification from unit-norm forcing to the corresponding response.

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) [33] has been shown to further reduce the cost of resolvent analysis of one- [62] and two-dimensional [73] 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 [88,41,86,48]. Brynjell-Rahkola *et al.* [15] used LU decomposition along with a power iteration with a Laplace preconditioner to increase the convergence rate of the resolvent modes. Ribeiro *et al.* [73] 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 , typically between $O(N^{1.5})$ to $O(N^2)$, remains a limiting factor, impeding the investigation of three-dimensional flows and other large systems. Recently, Houtman *et al.* [37] used an iterative solver as an alternative to LU decomposition to handle large systems. While this approach is attractive due to its potential to achieve $O(N)$ scaling, the absolute cost of iterative solvers depends heavily on the availability of an effective preconditioner, which is typically problem dependent, especially for the ill-conditioned matrices obtained from the linearized Navier–Stokes equations.

Resolvent modes can be computed at a reduced cost for slowly varying flows, *i.e.*, flows whose mean changes gradually in some spatial direction, by using spatial marching methods to approximate the action of the resolvent operator. Spatial marching methods approximately evolve perturbations in the slowly varying direction. The best-known spatial marching method is the parabolized stability equations (PSE), but the inherent ill-posedness of PSE [51] requires deleterious regularization that makes it ill-suited to compute resolvent modes in most cases [105]. One exception is very low frequencies, where PSE has been used to compute resolvent modes corresponding to boundary-layer streaks [77]. The one-way Navier–Stokes (OWNS) equations [102] 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 [106]. 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 [117]. 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.* [103] and Towne [101] 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) [81] to estimate the resolvent modes from data [35]. This approach benefits from the advancements in DMD [82] and is robust, but to accurately approximate the resolvent modes, many random initial conditions may need to be simulated.

Barthel *et al.* [10] 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. [10] 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. [63] used time stepping along with power iteration to compute resolvent modes for a flat-plate boundary-layer flow. Modes at a particular frequency of interest were computed by forcing the linearized equations exclusively at that frequency and time stepping until a steady-state solution is obtained. Gómez et al. [31] 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. [58] introduced two additional variations of time-stepping approaches for computing resolvent modes with improved efficiency. The first, referred to as the transient response method, evaluates the transitional response of the LNS to temporally compact forcing. The second variation, known as the steady-state response method, computes the steady-state solution of the LNS when it is forced with a set of harmonic frequencies. Both methods allow all frequencies of interest to be simultaneously computed by isolating each frequency in the flow response using a discrete Fourier transform. Additionally, the steady-state method can be easily paired with more advanced SVD algorithms (e.g., Arnoldi, rather than power iteration) to obtain multiple resolvent modes at each frequency.

Time-stepping methods for computing resolvent modes are potentially powerful because they obtain the action of the resolvent operator without the need for inverses or LU decomposition. Indeed, we will show that time-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 including simultaneous time integration for all frequencies of interest, the use of explicit solvers when implicit solvers require costly LU decomposition or preconditioning, efficient removal of undesired transient responses—especially if they decay slowly—and leveraging streaming calculations.

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 are computed simultaneously using a steady-state response approach as in Martini et al. [58]. 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. An open-source, parallelized implementation of our algorithm is available on GitHub (<https://github.com/AliFarghadan/RSVD-Delta-t>).

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

$$\frac{\partial \mathbf{q}}{\partial t} = \mathcal{N}(\mathbf{q}), \quad (1)$$

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

$$\mathbf{q}(\mathbf{x}, t) = \bar{\mathbf{q}}(\mathbf{x}) + \mathbf{q}'(\mathbf{x}, t) \quad (2)$$

partitions the flow state into the time-averaged mean $\bar{\mathbf{q}}$ and the fluctuation \mathbf{q}' . Substituting (2) into (1) and applying a Taylor expansion of \mathcal{N} around $\bar{\mathbf{q}}$,

$$\mathbf{A}(\bar{\mathbf{q}}) = \left. \frac{\partial \mathcal{N}}{\partial \mathbf{q}} \right|_{\mathbf{q}=\bar{\mathbf{q}}} \in \mathbb{C}^{N \times N}, \quad (3)$$

leads to

$$\begin{aligned}\frac{\partial \mathbf{q}'}{\partial t} &= \mathbf{A}(\bar{\mathbf{q}})\mathbf{q}' + \mathbf{B}\mathbf{f}'(\bar{\mathbf{q}}, \mathbf{q}'), \\ \mathbf{y}' &= \mathbf{C}\mathbf{q}',\end{aligned}\tag{4}$$

where \mathbf{A} 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. Here, the forcing term $\mathbf{f}'(\bar{\mathbf{q}}, \mathbf{q}')$ encapsulates all nonlinear terms from the Taylor expansion of \mathcal{N} around $\bar{\mathbf{q}}$, excluding the linear term represented by $\mathbf{A}(\bar{\mathbf{q}})\mathbf{q}'$. It can also represent exogenous forcing. The matrices \mathbf{B} and \mathbf{C} serve as *masks* in equation (4), providing generality and flexibility.

Resolvent analysis is most natural when \mathbf{A} is stable, i.e., all of its eigenvalues lie in the left-half plane. If \mathbf{A} is unstable, discounting can be used to obtain a stable system [45,115]. For more details on this procedure, see Rolandi et al. [74]. We assume that, if necessary, discounting has already been performed so that \mathbf{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

$$\mathcal{F}(\cdot) = \hat{(\cdot)}(\omega) = \int_{-\infty}^{+\infty} (\cdot) e^{-i\omega t} dt\tag{5}$$

of (4) and solving for the output yields

$$\hat{\mathbf{y}}(\omega) = \mathbf{R}(\omega)\hat{\mathbf{f}}(\omega),\tag{6}$$

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

$$\mathbf{R} = \mathbf{C}(i\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\tag{7}$$

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

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

$$\sigma = \frac{\|\hat{\mathbf{y}}\|_q}{\|\hat{\mathbf{f}}\|_f} = \frac{\|\mathbf{R}\hat{\mathbf{f}}\|_q}{\|\hat{\mathbf{f}}\|_f},\tag{8}$$

where $\|\mathbf{x}\|_f^2 = \langle \mathbf{x}, \mathbf{x} \rangle_f = \mathbf{x}^* \mathbf{W}_f \mathbf{x}$ computes the f -norm of any vector \mathbf{x} and $(\cdot)^*$ denotes the conjugate transpose. \mathbf{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 (8) is equivalent to computing the SVD of the resolvent operator [92]

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*,\tag{9}$$

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

2.2. Computation

Computing resolvent modes by following the definitions from the previous § involves two computationally intensive steps: (i) forming the resolvent operator by computing the inverse in (7) and (ii) computing the full singular value decomposition in (9). Both of these steps nominally require $O(N^3)$ operations. This is workable for one-dimensional problems, e.g., a channel flow [62], 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 [88,41,86,100,48]. This approach utilizes a mathematical equivalence to compute the resolvent modes faster than the natural approach. The right singular vectors of the resolvent operator are defined as the eigenfunctions of $\mathbf{R}^* \mathbf{R}$, i.e., $\mathbf{R}^* \mathbf{R} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^*$. 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 $\mathbf{U} = \mathbf{R} \mathbf{V} \mathbf{\Sigma}^{-1}$. The leading eigenvalues and eigenvectors can be efficiently computed via Arnoldi iteration [6]. The cost of the Arnoldi method relies on the desired number of modes and the convergence threshold. The Arnoldi algorithm requires the repeated computation of $\mathbf{R}^* \mathbf{R} \mathbf{v}$ for a given vector \mathbf{v} . Computing the LU decomposition of $(i\omega\mathbf{I} - \mathbf{A})$ circumvents computing \mathbf{R} directly. This is because solving $(i\omega\mathbf{I} - \mathbf{A})\mathbf{v} = \mathbf{b}$ using LU decomposition is equivalent to finding $\mathbf{v} = (i\omega\mathbf{I} - \mathbf{A})^{-1} \mathbf{b} = \mathbf{R} \mathbf{b}$ without explicit inversion. This is a common practice to speed up the process of constructing the orthonormal basis of the Krylov subspace [99]. However, the $O(N^2)$ scaling of the LU decomposition remains burdensome 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 [33] 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

Algorithm 1 RSVD-LU.

```

1: Inputs:  $\mathbf{R}, k, q$ 
2:  $\Theta \leftarrow \text{randn}(N, k)$ 
3:  $\mathbf{Y} \leftarrow \mathbf{R}\Theta$ 
4: if  $q > 0$  then
5:    $\mathbf{Y} \leftarrow \text{PI}(\mathbf{R}, \mathbf{Y}, q)$ 
6:  $\mathbf{Q} \leftarrow \text{qr}(\mathbf{Y})$ 
7:  $\mathbf{S} \leftarrow \mathbf{Q}^* \mathbf{R}$ 
8:  $(\tilde{\mathbf{U}}, \Sigma, \mathbf{V}) \leftarrow \text{svd}(\mathbf{S})$ 
9:  $\mathbf{U} \leftarrow \mathbf{Q}\tilde{\mathbf{U}}$ 
10: Outputs:  $\mathbf{U}, \Sigma, \mathbf{V}$ 

```

▷ Create random test matrices
 ▷ Sample the range of \mathbf{R}
 ▷ Optional power iteration
 ▷ Algorithm 2
 ▷ Build the orthonormal subspace \mathbf{Q}
 ▷ Sample the image of \mathbf{R}
 ▷ Obtain Σ, \mathbf{V}
 ▷ Recover \mathbf{U}

Algorithm 1. Inputs: resolvent operator \mathbf{R} , number of modes k , and number of power iterations q . Outputs: k response modes \mathbf{U} , k forcing modes \mathbf{V} and k gains Σ .

There exist several variations of the RSVD algorithm; here, we outline the algorithm from Halko et al. [33]. The first step is to sample the range of \mathbf{R} by forming its sketch (line 3)

$$\mathbf{Y} = \mathbf{R}\Theta, \quad (10)$$

where $\Theta \in \mathbb{C}^{N \times k}$ is a dense random test matrix (line 2) with $k \ll N$ columns that determines the number of leading modes to be approximated. Increasing the number of test vectors slightly beyond the desired number of modes enhances the accuracy of the leading modes. A feature of high-dimensional random vectors is that they form an orthonormal set with high probability [109], such that, on average, Θ projects uniformly onto all of the right singular vectors of \mathbf{R} . Therefore, the sketch preserves the leading left singular vectors of \mathbf{R} . An orthonormal basis $\mathbf{Q} \in \mathbb{C}^{N \times k}$ for the sketch is obtained via QR decomposition (line 6), which is then used to sample the image of \mathbf{R} (line 7) as

$$\mathbf{S} = \mathbf{Q}^* \mathbf{R}. \quad (11)$$

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

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

$$(\mathbf{R}\mathbf{R}^*)^q \mathbf{Y} = (\mathbf{U}\Sigma(\mathbf{V}^*\mathbf{V})\Sigma\mathbf{U}^*)^q \mathbf{Y} = (\mathbf{U}\Sigma^{2q}\mathbf{U}^*)^q \mathbf{Y} = (\mathbf{U}\Sigma^{2q}\mathbf{U}^*)\mathbf{Y}. \quad (12)$$

Raising the singular values to a high power artificially accelerates the decay rate of the singular values of \mathbf{R} , improving the effectiveness of the RSVD algorithm. The QR factorizations improve numerical stability, as discussed by Halko et al. [33].

Algorithm 2 Power iteration.

```

1: Inputs:  $\mathbf{R}, \mathbf{Y}, q$ 
2: for  $i = 1 : q$  do
3:    $\mathbf{Q} \leftarrow \text{qr}(\mathbf{Y})$ 
4:    $\mathbf{Y} \leftarrow \mathbf{R}^* \mathbf{Q}$ 
5:    $\mathbf{Q} \leftarrow \text{qr}(\mathbf{Y})$ 
6:    $\mathbf{Y} \leftarrow \mathbf{R}\mathbf{Q}$ 
7: Output:  $\mathbf{Y}$ 

```

▷ For stabilization purposes
 ▷ Sample the image of \mathbf{R}
 ▷ For stabilization purposes
 ▷ Sample the range of \mathbf{R}

Algorithm 2. Inputs: resolvent operator \mathbf{R} , k response modes from the first direct action \mathbf{Y} , and number of power iterations q . Outputs: k response modes \mathbf{Y} .

3.2. RSVD for resolvent analysis

The algorithm outlined in the previous section assumes direct access to the matrix \mathbf{R} . In the context of resolvent analysis, \mathbf{R} is defined in terms of an inverse, which should be avoided. Ribeiro et al. [73] addressed this challenge by adopting the approach developed by Jeun et al. [41] 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, $\mathbf{Y} = \mathbf{R}\boldsymbol{\Theta}$ (line 3 of Algorithm 1) can be obtained by solving the linear system

$$(i\omega\mathbf{I} - \mathbf{A})\mathbf{Y} = \boldsymbol{\Theta} \quad (13)$$

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

$$(i\omega\mathbf{I} - \mathbf{A})^*\mathbf{S}^* = \mathbf{Q}. \quad (14)$$

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

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

$$(i\omega\mathbf{I} - \mathbf{A}) = \mathbf{L}\mathbf{P}, \quad (15)$$

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

$$(i\omega\mathbf{I} - \mathbf{A})^* = (\mathbf{L}\mathbf{P})^* = \mathbf{P}^*\mathbf{L}^*. \quad (16)$$

Solving these linear systems is indeed significantly less computationally demanding than computing the inverse of $(i\omega\mathbf{I} - \mathbf{A})$ to form \mathbf{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 [73].

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. [63] and recently was improved upon by Martini et al. [58], 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. A key difference between this paper and the work of Martini et al. [58] is that while their time-stepping approaches are integrated into Arnoldi’s method to compute the leading resolvent modes, we will incorporate it into the RSVD algorithm. RSVD has demonstrated superior speed compared to previous methods, including Arnoldi and standard SVD-based techniques [73]. Additionally, we introduce new approaches to minimize the CPU and memory costs for any time-stepping method for computing resolvent modes.

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 \mathbf{R} on a vector (or matrix) \mathbf{b} is defined as follows; Given \mathbf{b} , our objective is to compute $\mathbf{x} = \mathbf{R}\mathbf{b}$, which is equivalent to solving the linear system $\mathbf{R}^{-1}\mathbf{x} = \mathbf{b}$ for \mathbf{x} .

Starting with a harmonically forced ordinary differential equation (ODE)

$$\frac{d\mathbf{q}}{dt} = \mathbf{A}\mathbf{q} + \mathbf{f}, \quad (17)$$

where

$$\mathbf{f}(t) = \hat{\mathbf{f}}e^{i\omega t} \quad (18)$$

is the harmonic forcing with frequency $\omega \in \mathbb{R}$ and $\hat{\mathbf{f}} \in \mathbb{C}^N$ is an arbitrary vector. The steady-state response of (17) is

$$\mathbf{q}(t) = \hat{\mathbf{q}}_s e^{i\omega t}, \quad (19)$$

where

$$\hat{\mathbf{q}}_s = (i\omega\mathbf{I} - \mathbf{A})^{-1}\hat{\mathbf{f}} = \mathbf{R}\hat{\mathbf{f}} \quad (20)$$

is the Fourier-domain solution. Therefore, the action of \mathbf{R} can be obtained by computing the steady-state solution $\mathbf{q}(t)$ of (17) and subsequently taking a Fourier transform to obtain $\hat{\mathbf{q}}_s$. Similarly, the action of \mathbf{R}^* can be obtained by computing the steady-state response $\mathbf{z}(t)$ of the adjoint equation

$$\begin{aligned} -\frac{d\mathbf{z}}{dt} &= \mathbf{A}^*\mathbf{z} + \mathbf{f}, \\ \mathbf{f} &= \hat{\mathbf{f}}e^{i\omega t}, \end{aligned} \quad (21)$$

backward in time and taking a Fourier transform to obtain

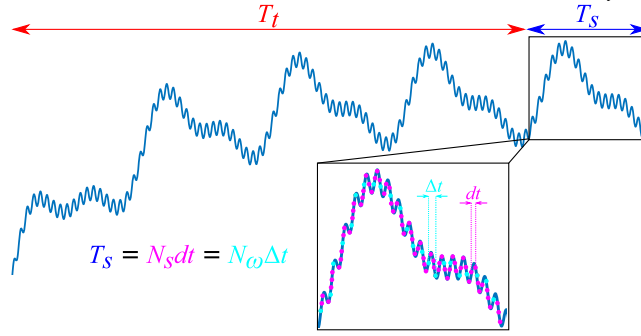


Fig. 1. Schematic of the response waveform. The solution contains a transient portion of length T_t before the steady-state solution of period T_s is achieved. The numerical solution contains N_s time steps of size dt within one period of the steady-state solution, but only N_ω points with Δt spacing are required to decompose the N_ω frequencies of interest without aliasing. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\hat{z}_s = (-i\omega\mathbf{I} - \mathbf{A}^*)^{-1} \hat{f} = \mathbf{R}^* \hat{f}. \quad (22)$$

We use a discrete adjoint, *i.e.*, we take the complex conjugate of the LNS operator, \mathbf{A}^* , as the adjoint operator. This is consistent with the common approach of computing resolvent modes from the SVD of \mathbf{R} without explicitly defining an adjoint. We note, however, that using a continuous formulation of the adjoint equations or a tailored set of differentiation operators and boundary conditions such as the summation-by-parts, simultaneous approximation term (SBP-SAT) method [16] may be necessary in certain cases to prevent numerical artifacts [12,110].

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. [58] 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 (17) typically generates a transient response T_t before obtaining the desired steady-state solution, as shown in Fig. 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 period 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

$$\Omega = \{0, \pm\omega_{min}, \pm 2\omega_{min}, \pm 3\omega_{min}, \dots, \pm\omega_{max}\}, \quad (23)$$

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

$$\mathbf{f} = \sum_{\omega_j \in \Omega} \hat{f}_j e^{i\omega_j t} \quad (24)$$

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 [66]. Performing time integration of (17) results in computing N_s steady-state snapshots within the T_s -period, where typically $N_s \geq N_\omega$, as the time step (dt) is chosen to ensure sufficient integration accuracy. Ultimately, by choosing N_ω steady-state snapshots, we can determine the Fourier coefficients by taking a DFT.

To elaborate on the previous point, assume a set of snapshots $\mathbf{Q}_{N_s} = \{q_1, q_2, q_3, \dots, q_{N_s}\}$ (analogous to the pink dots in Fig. 1), where q_j represents the j^{th} steady-state snapshot in the time domain. The fast Fourier transform (FFT) can efficiently compute $\hat{Q}_{N_s} = \{\hat{q}_1, \hat{q}_2, \hat{q}_3, \dots, \hat{q}_{N_s}\}$. However, the maximum resolved frequency within \hat{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 $\mathbf{Q}_{N_\omega} = \{q_1, q_2, q_3, \dots, q_{N_\omega}\}$ (analogous to the cyan dots in Fig. 1). Taking the FFT of \mathbf{Q}_{N_ω} yields $\hat{Q}_{N_\omega} = \{\hat{q}_1, \hat{q}_2, \hat{q}_3, \dots, \hat{q}_{N_\omega}\}$, where each member \hat{q}_j represents the solution to $(i\omega_j\mathbf{I} - \mathbf{A})\hat{q}_j = \hat{f}_j$, with $\omega_j \in \Omega$.

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

$$T_s = dt \times N_s = \Delta t \times N_\omega. \quad (25)$$

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

$$\Delta t = \frac{T_s}{N_\omega} = \frac{2\pi/\omega_{min}}{2 \lceil \frac{\omega_{max}}{\omega_{min}} \rceil} \quad (26)$$

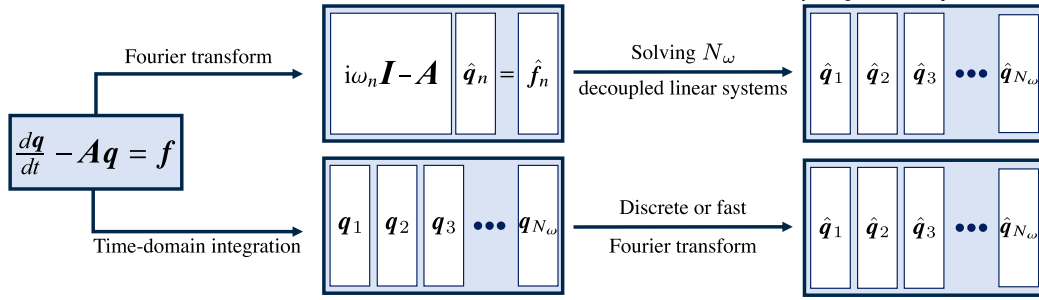


Fig. 2. Flowchart depicting the action of \mathbf{R} on N_ω inputs for the RSVD-LU (upper route) and the RSVD- Δt (bottom route) algorithms. Both routes produce the same result, but the bottom route is computationally advantageous for large systems.

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

Fig. 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 \mathbf{R} and \mathbf{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 .

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

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

As in the standard RSVD algorithm (Algorithm 1), 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{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{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{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 vector V and gains Σ are obtained via an economy SVD of the $N \times k$ matrix \hat{S} (line 8), and left singular vectors U are recovered in line 9.

In the context of resolvent analysis, the RSVD algorithm can be slightly modified to enhance the accuracy of the response modes. One approach, as described by Ribeiro et al. [73], involves performing an additional direct action after the final SVD in the standard RSVD algorithm to compute more accurate response modes. The cost of this additional step is equivalent to performing a direct action.

On the other hand, the costliest steps in executing the RSVD- Δt algorithm are typically the direct and adjoint actions, while the costs of the QR and SVD steps are almost negligible (see Table 3). This suggests that we can perform an SVD instead of QR in line 6 and use the left singular vectors as the response modes instead of recovering them later in line 9. The modes we obtain this way are more

Table 1

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

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

accurate than the recovered versions. Therefore, if one does not wish to perform an additional action, we recommend performing SVD in line 6 and ignoring 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 \mathbf{A} is constructed, the goal is to solve the linear system given by

$$(i\omega\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b} \quad (27)$$

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

6.1. CPU cost

Direct solvers find the solution of (27) to machine precision. A common approach is to find the LU decomposition of $(i\omega\mathbf{I} - \mathbf{A})$ 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 [26] and is often the dominant cost of solving a linear system [55]. 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 [3]. 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 [5], MUMPS [4], PARDISO [78], and Hypre [30], 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 [76], which can cause slow convergence and numerical stability issues for iterative solvers [91]. 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 [108].

In principle, iterative solvers are attractive when solving (27) 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 $(i\omega\mathbf{I} - \mathbf{A})$, as explained above. One potential solution is the common practice of using a preconditioner [75]. 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 [7,11,68]. Despite numerous advancements in this field, not all matrices have effective preconditioners. Some recent studies [37] are exploring the use of iterative methods to solve (27) within the context of resolvent analysis, but direct methods, especially LU decomposition, have long been the dominant choice [62,41,86,73].

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 (17)) has a long history and is a mature field [32,113,108]. 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

$$\mathcal{A}\mathbf{x} = \mathbf{b} \quad (28)$$

to be solved at every iteration. Here, $\mathbf{b} \in \mathbb{C}^{N \times k}$ is a function of the solution at previous time and the exogenous forcing (if present), and $\mathcal{A} \in \mathbb{C}^{N \times N}$ is the temporal discretized operator, which is a function of the linear operator \mathbf{A} . For example, \mathcal{A} can be written

as a first-order polynomial of the form $\mathcal{A} = c_1 \mathbf{I} + c_2 \mathbf{A}$ for multi-step methods, where constants are determined based on integration scheme and time step, e.g., $\mathcal{A} = \mathbf{I} - dt\mathbf{A}$ for backward Euler. A superficial comparison between (28) and (27) indicates that implicit time steppers suffer from the same issues elaborated above. However, the key difference is that \mathbf{A} is multiplied by the (small) time step dt , so the ill-conditioning of \mathbf{A} is largely overwhelmed by the ideal conditioning of the identity matrix \mathbf{I} . This improved conditioning makes possible the application of iterative solvers. Implicit integrators 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 $O(N)$.

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 (27), 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 [96]. 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 $O(N)$. The number of times these operations must be performed over a fixed time interval is determined by the time step dt . If the time step is set by a CFL-like stability condition, then it scales like $O(N^{-1/2})$ and $O(N^{-1/3})$ for two- and three-dimensional problems, respectively, and the total number of time steps N_t in a fixed time interval scales as the inverse of these values. The overall CPU cost scales like $O(NN_t)$, leading to $O(N^{3/2})$ and $O(N^{4/3})$ scaling for two- and three-dimensional problems, respectively. In practice, however, the time step is chosen to control the error associated with the highest frequency of interest rather than being determined by a CFL condition, as discussed in §7.2.1, and is thus independent of N . Accordingly, we observe linear scaling when using explicit integrators in practice.

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 [24]. Generally, direct solvers are more robust than iterative solvers but can consume significant memory due to the fill-in process of factorization [55]. 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 [106]. 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 (28), 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 \mathbf{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 \mathbf{A} [67,58]. 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 \mathbf{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. [33], and it naturally depends on the gain separation, defined as the ratio σ_i/σ_{i+1} , where σ_i is the i^{th} singular value. 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 \mathbf{R} and \mathbf{R}^* using time stepping, two types of errors are introduced in addition to the RSVD approximation: truncation and transient errors.

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 [32,113]. 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

$$LTE = C \frac{d^p f(t)}{dt^p} O(dt^p), \tag{29}$$

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 (29), we observe that

$$LTE \propto O((\omega dt)^p). \tag{30}$$

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 (17) can be written as a sum of its transient and steady-state components,

$$\mathbf{q}(t) = \mathbf{q}_t(t) + \mathbf{q}_s(t), \tag{31}$$

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

$$\hat{\mathbf{q}}(\omega) = \hat{\mathbf{q}}_t(\omega) + \hat{\mathbf{q}}_s(\omega). \tag{32}$$

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 [80] but eventually decays at the rate of the least-damped eigenvalue of \mathbf{A} .

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

$$\epsilon = \frac{\|\mathbf{q}_t\|}{\|\mathbf{q}_s\|}, \tag{33}$$

where the l^2 -norms can be replaced with $\|\cdot\|_q$ for non-identity weight matrices. In cases where we solve (17) with a zero initial condition (which is often the case), *i.e.*, $\mathbf{q}(0) = \mathbf{q}_t(0) + \mathbf{q}_s(0) = 0$, the transient error is initially one,

$$\epsilon(0) = \frac{\|\mathbf{q}_t(0)\|}{\|\mathbf{q}_s(0)\|} = 1. \tag{34}$$

In the long term, the transient error approaches zero,

$$\lim_{t \rightarrow \infty} \epsilon(t) = \lim_{t \rightarrow \infty} \frac{\|\mathbf{q}_t(t)\|}{\|\mathbf{q}_s(t)\|} = 0, \tag{35}$$

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

The eigenspectrum of the linearized system \mathbf{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 \mathbf{A} is challenging, especially for large systems. Even obtaining a small number of eigenvalues using the Krylov-Schur method can be cumbersome. Therefore, a practical approach to understanding the long-term behavior of a system is to simulate the homogeneous ODE

$$\frac{d\mathbf{q}_h}{dt} - \mathbf{A}\mathbf{q}_h = 0, \tag{36}$$

initialized with a random state [29,28]. A random perturbation represents a worst-case scenario, as it excites all the slow modes of \mathbf{A} . By monitoring the norm of \mathbf{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.

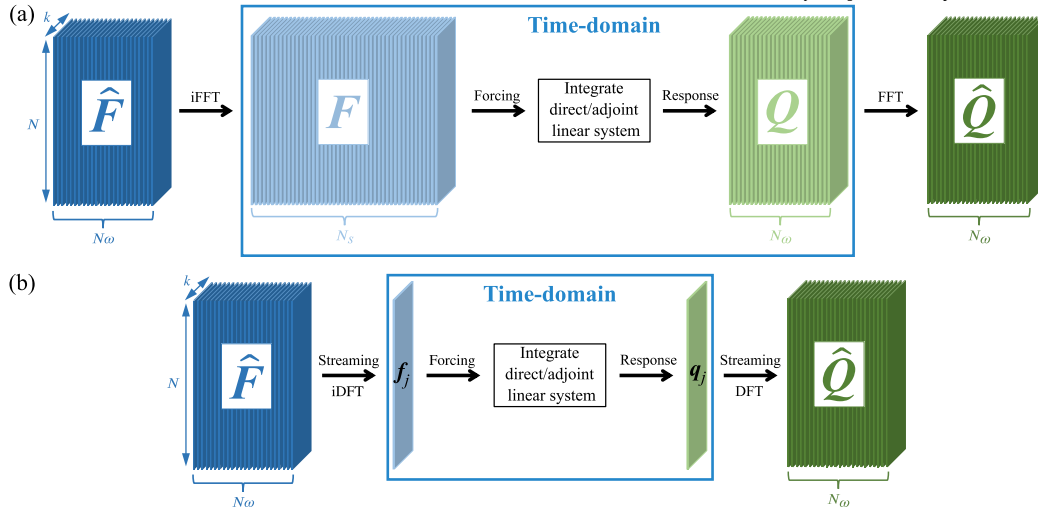


Fig. 3. Schematic of the action of \mathbf{R} with (a) FFT/iFFT and (b) streaming DFT/iDFT methods to transform between the Fourier and time domains.

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: streaming Fourier sums and shortcuts for real-valued matrices.

8.1.1. Streaming Fourier sums

A straightforward implementation of computing the action of \mathbf{R} (or \mathbf{R}^*) via time stepping entails (i) transferring the forcing from Fourier space to the time domain, $\hat{\mathbf{F}} \xrightarrow{\text{iFFT}} \mathbf{F}$, (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, $\mathbf{Q} \xrightarrow{\text{FFT}} \hat{\mathbf{Q}}$. A schematic of these steps is displayed in Fig. 3(a).

The first step requires zero-padding $\hat{\mathbf{F}} \in \mathbb{C}^{N \times k \times N_\omega}$ since $\mathbf{F} \in \mathbb{C}^{N \times k \times N_s}$ 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. $\hat{\mathbf{F}}$ is automatically discarded before proceeding to the second step. In step (ii), $\mathbf{f}_j \in \mathbf{F}$ is used to force the linear system at each time step until the transient ends, and the steady-state responses are stored in \mathbf{Q} . After integration, \mathbf{F} is no longer needed and is removed. Lastly, obtaining $\hat{\mathbf{Q}}$ from \mathbf{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 \mathbf{F} and \mathbf{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 and Towne [84]. This procedure is shown schematically in Fig. 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 $\hat{\mathbf{F}}$ and $\hat{\mathbf{Q}}$. 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,

$$\mathbf{f}_p = \sum_{s=1}^{N_\omega} \mathbf{Z}'_{ps} \hat{\mathbf{f}}_s, \quad (37)$$

where $\mathbf{Z}'_{ps} = \exp(-2\pi i / N_s)^{(p-1)(s-1)}$. The integer p ($1 \leq p \leq N_s$) specifies the phase of the periodic forcing at the current time step. Here, $\hat{\mathbf{f}}_s \in \mathbb{C}^{N \times k \times N_\omega}$ 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 $\mathbf{f}_p \in \mathbb{C}^{N \times k}$. This process continues in a loop of size N_s until the transient is passed and steady-state data is computed.

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.

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

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 \mathbf{q}_l within the steady-state response contributes to each Fourier mode according to the partial sum

$$[\hat{\mathbf{q}}_s]_r = [\hat{\mathbf{q}}_s]_{r-1} + \mathbf{Z}_{ls} \mathbf{q}_r = \sum_{l=1}^r \mathbf{Z}_{ls} \mathbf{q}_l, \tag{38}$$

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 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. Indeed, this is usually case for LNS operator, except when considering a non-zero wavenumber in a Fourier-transformed homogeneous direction or when using complex-valued non-reflecting boundary conditions [21,38]. When \mathbf{A} is real-valued, memory requirements can be significantly reduced. Having $\mathbf{R}_\omega = (i\omega \mathbf{I} - \mathbf{A})^{-1} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^*$, the resolvent operator corresponding to $-\omega$ can be written as

$$\mathbf{R}_{-\omega} = (-i\omega \mathbf{I} - \mathbf{A})^{-1} = \overline{(i\omega \mathbf{I} - \mathbf{A})}^{-1} = \overline{(i\omega \mathbf{I} - \mathbf{A})}^{-1} = \overline{\mathbf{R}}_\omega = \overline{\mathbf{U}} \mathbf{\Sigma} \overline{\mathbf{V}}^*, \tag{39}$$

where $\overline{(\cdot)}$ denotes the complex conjugate and $\mathbf{A} = \overline{\mathbf{A}}$ when \mathbf{A} is real-valued. Equation (39) 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 $\Omega_+ = \{0, +\omega_{\min}, +2\omega_{\min}, \dots, +\omega_{\max}\}$, 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 ($\hat{\mathbf{F}}$ and $\hat{\mathbf{Q}}$ 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 (17), 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 (17), $\mathbf{q}_1 = \mathbf{q}(t_1)$ and $\mathbf{q}_2 = \mathbf{q}(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts, as in (31), as

$$\mathbf{q}_1 = \mathbf{q}_{s,1} + \mathbf{q}_{t,1}, \quad (40)$$

$$\mathbf{q}_2 = \mathbf{q}_{s,2} + \mathbf{q}_{t,2},$$

where $\mathbf{q}_{s,1}$, $\mathbf{q}_{s,2}$, $\mathbf{q}_{t,1}$, and $\mathbf{q}_{t,2}$ are four unknowns. Applying a prescribed forcing in (17) at a single frequency ω yields

$$\mathbf{q}_{s,2} = \mathbf{q}_{s,1} e^{i\omega\Delta t}. \quad (41)$$

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

$$\mathbf{q}_{t,2} = e^{A\Delta t} \mathbf{q}_{t,1}. \quad (42)$$

Simplifying (40), (41), and (42) for $\mathbf{q}_{t,1}$, we obtain

$$(\mathbf{I} - e^{-i\omega\Delta t} e^{A\Delta t}) \mathbf{q}_{t,1} = \mathbf{b}, \quad (43)$$

where $\mathbf{b} = \mathbf{q}_1 - \mathbf{q}_2 e^{-i\omega\Delta t}$ is known from the time-stepping solution. Equation (43) holds for any two points in time with arbitrary separation Δt . The exact steady-state solution with no transient error is obtained by solving (43) for $\mathbf{q}_{t,1}$ and using (40) 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 $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \dots, \mathbf{q}_{N_\omega}\}$ to construct $\hat{\mathbf{Q}} \in \mathbb{C}^{N \times N_\omega}$, where the snapshots are equidistant with a time interval of Δt . Additionally, we define $\mathbf{Q}^{\Delta t} = \{\mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \dots, \mathbf{q}_{N_\omega+1}\}$ as a shifted matrix, resulting in $\hat{\mathbf{Q}}^{\Delta t} \in \mathbb{C}^{N \times N_\omega}$. Here, $\hat{\mathbf{q}}_j \in \hat{\mathbf{Q}}$ represents \mathbf{q}_1 in the above equations, while $\hat{\mathbf{q}}_j^{\Delta t} \in \hat{\mathbf{Q}}^{\Delta t}$ represents \mathbf{q}_2 , both oscillating at the same frequency. Therefore, a single time stepping is sufficient to obtain (43) for all $\omega \in \Omega$.

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

Solving (43) can be computationally expensive, particularly for large systems, even if we assume that computing $e^{A\Delta t}$ is feasible. To address this issue, one possible approach is to choose a small Δt and expand the exponential term as $e^{A\Delta t} = \sum_j \frac{(A\Delta t)^j}{j!}$. However, this leads to solving a similar linear system to (27), 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 (43).

Consider a low-dimensional representation of the transient response as

$$\mathbf{q}_{t,1} = \boldsymbol{\phi} \boldsymbol{\beta}_1, \quad (44)$$

where $\boldsymbol{\phi} \in \mathbb{C}^{N \times r}$, with $r \ll N$, is an orthonormal trial basis spanning the transient response and $\boldsymbol{\beta}_1 \in \mathbb{C}^r$ represents the coefficients describing the transient in this basis. By substituting (44) into (43), the linear system

$$(\mathbf{I} - e^{-i\omega\Delta t} e^{A\Delta t}) \boldsymbol{\phi} \boldsymbol{\beta}_1 = \mathbf{b} \quad (45)$$

is overdetermined. Petrov-Galerkin projection with test basis $\boldsymbol{\psi} \in \mathbb{C}^{N \times r}$ is employed to close (45), giving

$$\boldsymbol{\psi}^* (\mathbf{I} - e^{-i\omega\Delta t} e^{A\Delta t}) \boldsymbol{\phi} \boldsymbol{\beta}_1 = \boldsymbol{\psi}^* \mathbf{b}. \quad (46)$$

Solving (46) for $\boldsymbol{\beta}_1$ and inserting the solution into (44) yields

$$\mathbf{q}_{t,1} = \boldsymbol{\phi} (\boldsymbol{\psi}^* \boldsymbol{\phi} - e^{-i\omega\Delta t} \tilde{\mathbf{M}})^{-1} \boldsymbol{\psi}^* \mathbf{b}, \quad (47)$$

where

$$\tilde{\mathbf{M}} = \boldsymbol{\psi}^* e^{A\Delta t} \boldsymbol{\phi} \in \mathbb{C}^{r \times r} \quad (48)$$

is a reduced matrix that maps the coefficients. The advantage of this strategy is that it allows for the computation of the inverse of $(\boldsymbol{\psi}^* \boldsymbol{\phi} - e^{-i\omega\Delta t} \tilde{\mathbf{M}})$ due to its reduced dimension. Obtaining $\tilde{\mathbf{M}}$ is also an efficient process, involving two steps: (i) integrating the

columns of ϕ over Δt , and (ii) projecting $e^{A\Delta t}\phi$ onto the columns of ψ . The construction cost of \tilde{M} for each $\omega \in \Omega$ is primarily determined by the first step. Specifically, when the number of columns in ϕ is $r = N_\omega$ and $\Delta t = T_s/N_\omega$, the total cost of constructing \tilde{M} for all $\omega \in \Omega$ is equivalent to integrating the LNS equations for an additional T_s duration. While it is possible to use a Taylor expansion of $e^{A\Delta t}$ to compute $e^{A\Delta t}\phi$, the number of terms required for the expansion can become excessive when $\Delta t > dt$. Therefore, we opt for time integration as a more practical alternative.

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 test function. Using this strategy with either Galerkin or Petrov-Galerkin projections, the accuracy of the solution relies on the ability of the column space of 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 trial 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, (47) 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 A as the chosen trial basis. By excluding the least-damped eigenvectors, we effectively increase the decay rate of the transient response. Let λ_1 denote the least-damped eigenvalue of A , with V_1 representing the corresponding eigenvector. We define $\phi = V_1$, thereby removing the transient component projected onto V_1 . As a result, the norm of the updated transient, obtained by subtracting this projection, follows the decay rate associated with the second least-damped eigenvalue of A . Similarly, the trial basis ϕ can encompass the first $r - 1$ least-damped eigenvectors, $\phi = \text{orth}\{V_1, V_2, \dots, V_{r-1}\}$, leading to a decay rate governed by the r^{th} least-damped eigenvalue of A . For this particular trial basis, Petrov-Galerkin projection can be utilized, where ψ incorporates the adjoint eigenvectors. This approach ensures the complete elimination of transient projection onto the least-damped modes. To be clear, this procedure does not eliminate the impact of these modes on the steady-state response, but only on the transient response.

The main challenge associated with this trial basis is the computational cost of computing the least-damped eigenvectors (and adjoint eigenvectors in the case of Petrov-Galerkin projection), especially for large systems, even when using algorithms designed for this purpose, e.g., Krylov-based methods [29,28]. Overall, the least-damped modes of A are most helpful for systems that suffer from only a few slowly decaying modes.

Another powerful trial basis is formed by stacking the snapshots into a matrix during the integration of the LNS equations, resulting in $\phi = \text{orth}\{q_1, q_2, q_3, \dots, q_r\}$ (an orthogonalization of the matrix of snapshots). Specifically, ϕ can be constructed as the union of \hat{Q} and $\hat{Q}^{\Delta t}$ as a reliable trial basis. Performing QR decomposition on this matrix is essential to ensure orthogonality. As the LNS equations are allowed to run for a longer duration, ϕ becomes an increasingly effective trial basis, providing improved estimates of the transient responses across all frequencies $\omega \in \Omega$. We have observed that this basis is more accurate for higher frequencies compared to lower ones.

A feature of our transient-removal approach is its flexibility in incorporating multiple trial bases. For instance, by considering the matrix of least-damped eigenvectors of A in ϕ_1 and the on-the-fly snapshots in ϕ_2 , a combined trial basis $\phi = \phi_1 \cup \phi_2$ can be constructed and orthogonalized. The combination of trial 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 (36) using a random initial condition with unit norm. By employing (40), (41), and (42), 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 $q_{s,0} = -q_{t,0}$ 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 (33), is computed as the ratio of norms between the updated transient and steady-state responses, which monotonically decreases after the transient growth phase. This iterative process is performed for all $\omega \in \Omega$, necessitating the generation of fresh snapshots for the steady-state responses while keeping the transient response fixed. The computational expense associated with obtaining this *a priori* error estimate is primarily determined by the integration of the homogeneous system and typically constitutes less than 5% of the overall cost of executing the complete algorithm for computing the resolvent modes. We illustrate the application of this strategy using various trial bases in §9.

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 [95] and has subsequently been used more generally as a convenient model of a flow susceptible to non-modal amplification [39,8,19,17]. Here, we use it as an inexpensive test case to validate our algorithm. The complex Ginzburg-Landau system follows the form of (4) with

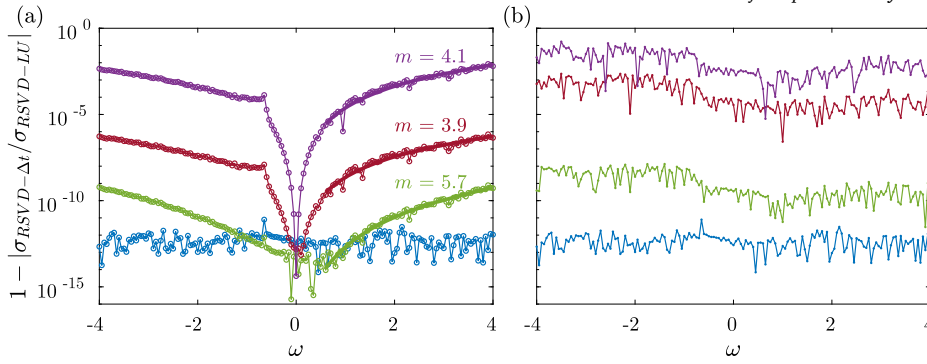


Fig. 4. Relative error between gains computed using the RSVD-LU and RSVD- Δt algorithms for the Ginzburg-Landau problem: (a) $T_t = 5000$ and $(TSS, dt) = \{\text{BDF4}, 0.1\}$ (purple), $\{\text{BDF4}, 0.01\}$ (red), $\{\text{BDF6}, 0.01\}$ (green), and $\{\text{BDF6}, 0.001\}$ (blue) varies; (b) $\{\text{BDF6},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

$$\begin{aligned}
 \mathbf{A} &= -\nu \frac{\partial}{\partial x} + \gamma \frac{\partial^2}{\partial x^2} + \mu(x), \\
 \mu(x) &= (\mu_0 - c_\mu^2) + \frac{\mu_2}{2} x^2, \\
 \mathbf{B} &= \mathbf{C} = \mathbf{I}.
 \end{aligned} \tag{49}$$

Following Bagheri et al. [8], 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 [8]. To explicitly build the \mathbf{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 an unbounded domain without a need for explicit boundary conditions [8]. The weight matrix \mathbf{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. While the order of magnitude of “tiny” time step and “lengthy” transient duration may vary depending on the problem setup, the key point is that time-stepping error can be reduced to machine precision accuracy if desired.

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. This implies that whether we are computing the action of optimal or suboptimal modes, each mode will exhibit a similar order of error due to time stepping. The desired set of frequencies is $\Omega \in [-4, 4]$ with $\Delta\omega = 0.05$. The gains of the Ginzburg-Landau system are computed using RSVD and RSVD- Δt and the relative errors for various cases are shown in Fig. 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 Fig. 4(a), the transient length is held fixed at $T_t = 5000$ and the gains are obtained using $\{\text{BDF6}, dt = 10^{-2}\}$, $\{\text{BDF4}, dt = 10^{-2}\}$, and $\{\text{BDF4}, dt = 10^{-1}\}$. For all four cases, the relative error is around $O(10^{-13})$ at $\omega = 0$, confirming that the transient effect is negligible. Moving away from zero frequency, the errors increase like $O(\omega^{-4})$ and $O(\omega^{-6})$ for the BDF4 and BDF6 schemes, respectively, consistent with the theoretical asymptotic estimates given in (30).

Fig. 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 $\{\text{BDF6}, dt = 10^{-3}\}$ is held fixed, ensuring negligible truncation error, and the transient lengths are varied as $T_t = \{500, 1000, 2500, 5000\}$. Smaller values of T_t leave more transient residual in the steady-state response. 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 Fig. 5(a), and the three least-damped (and thus slowest decaying) modes have decay 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 . Fig. 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 (17) for a duration of 5000 time units 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.

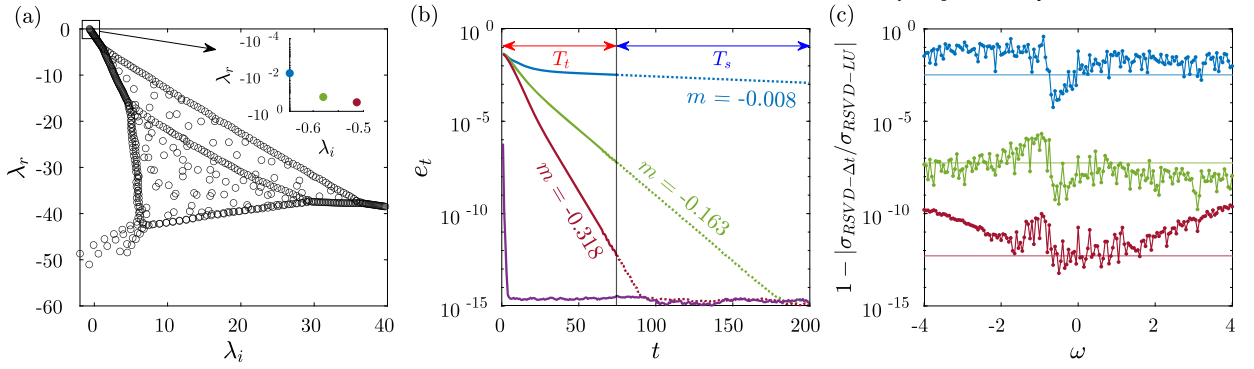


Fig. 5. Transient-removal for the Ginzburg-Landau test problem: (a) Spectrum of Ginzburg-Landau operator with a zoomed-in view of the three least-damped eigenvalues. (b) Transient error measurement: blue curve represents original decay, while green, red, and purple curves depict decay using Galerkin projection with ϕ of \mathbf{V}_1 , $\{\mathbf{V}_1, \mathbf{V}_2\}$, and a matrix of snapshots, respectively. (c) Relative error comparison between the RSVD- Δt and RSVD-LU algorithms. Solid horizontal lines in (c) represent the expected transient error arising from the transient norm at the end of the T_t (the black vertical line in (b)).

The natural decay in this system occurs slowly, as illustrated in Fig. 5(b). By defining ϕ_1 as \mathbf{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 $\phi_2 = \{\mathbf{V}_1, \mathbf{V}_2\}$ further steepens the decay rate, aligning closely with the corresponding least-damped eigenvalues shown in Fig. 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 Fig. 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. While the target number of modes in this analysis is three, we include three additional test vectors to enhance the overall accuracy of the modes. Additionally, RSVD- Δt uses a BDF6 integrator with $dt = 0.01$ and $T_t = 100$. The transient length is two orders of magnitude shorter than the expected length based on the original decay rate. This reduction is a result of removing the slowest decaying component, which allows us to shorten the simulation duration while still achieving a time-stepping error of $O(10^{-8})$, as shown in Fig. 6(c). A standard Arnoldi-based approach (SVD-based with no approximation) is used to provide a ground-truth reference for defining the error.

The leading three singular values and corresponding relative errors are shown in Fig. 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. Increasing the number of power iterations allows a broader range of frequencies to achieve machine precision accuracy, which aligns with the expectations for RSVD approximations [33]. Increasing the number of power iterations from $q = 1$ to 2 to 3 enables the leading modes to reach machine precision across the entire spectrum of interest, while also pushing suboptimal modes closer to machine precision. We have found one power iteration to be sufficient for most problems, and we recommend this as a default value for our algorithm. The expected error from power iteration is problem specific. For the Ginzburg-Landau problem, a single power iteration may reduce the error to below the 10% threshold for the third mode across the frequency spectrum and achieve machine precision for the optimal mode at peak frequencies where a significant gap exists. Moreover, RSVD is generally more effective when a large gap exists between the first (or first few) singular values and the rest of the spectrum. In cases with closely spaced singular values—such as at the higher end of the spectrum in the Ginzburg-Landau problem—the modes converge more slowly.

The accuracy of the modes, *i.e.*, the left and right singular vectors, is ensured when the gains closely match the ground truth [33]. This relationship is illustrated in Fig. 6, where we compare the inner-product relative error between modes computed via SVD and RSVD- Δt . The inner-product error between two unit-norm vectors \mathbf{v}_1 and \mathbf{v}_2 is formally defined as

$$e_{ip} = 1 - \langle \mathbf{v}_1, \mathbf{v}_2 \rangle. \quad (50)$$

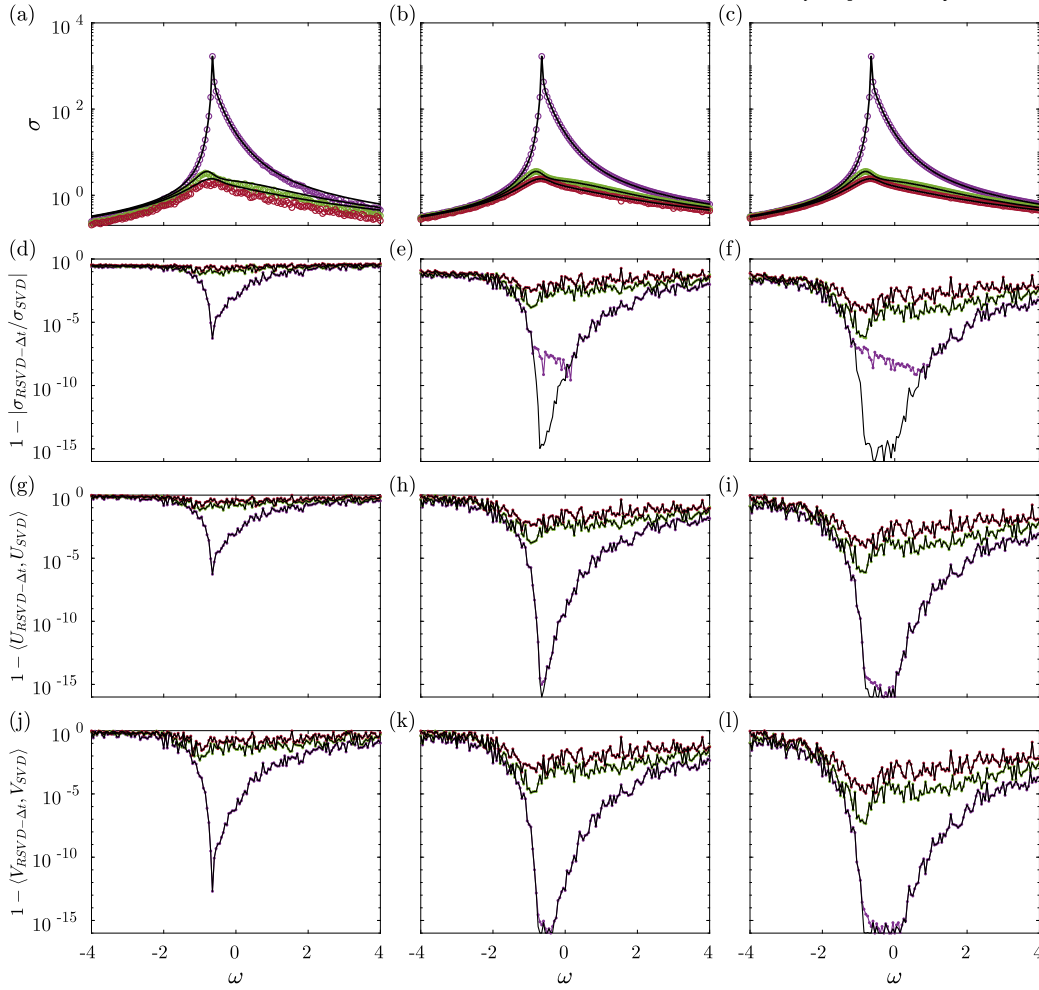


Fig. 6. Impact of power iteration on the Ginzburg-Landau gains: (a-c) the gains of the first three optimal modes using SVD (line) and RSVD- Δt (circle), and (d-e) the relative error between them. The inner-product error between response (g-i) and forcing (j-l) modes are plotted. (a,d,g,j), (b,e,h,k), and (c,f,i,l) correspond to q of 0, 1, and 2, respectively. Black lines in (d-l) show the relative error between the RSVD-LU algorithm and SVD for reference. Purple, green, and red colors represent the first three leading modes, respectively.

By comparing the relative error of modes obtained from RSVD-LU and SVD (black lines in Fig. 6) across varying numbers of power iterations, we observe that when the gain relative error is below 10%, the inner-product error for both the forcing and response modes is similarly small, mostly of the same order or better. The inner-product error between RSVD- Δt and SVD aligns closely with that of RSVD-LU and SVD. However, the gain relative error between RSVD- Δt and SVD remains bounded around $O(10^{-8})$, while the inner-product relative error can approach machine precision. This behavior is expected and can be understood through perturbation analysis.

As demonstrated by Stewart [93,94], when considering a perturbed matrix with a relative error of $O(\epsilon)$, a second-order convergence of $O(\epsilon^2)$ in the inner products is expected. Given that our algorithm’s error is additive due to the time-stepping process, $O(\epsilon)$ error in the gains ensures $O(\epsilon^2)$ inner-product errors between the modes computed by RSVD- Δt and RSVD-LU. However, fundamentally, our algorithm cannot exceed the accuracy achieved by RSVD-LU.

9.2. Round turbulent jet

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

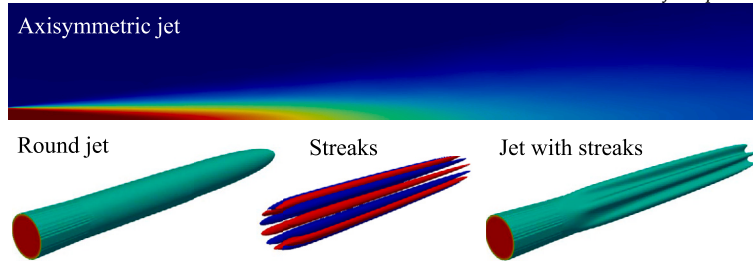


Fig. 7. The mean streamwise velocity of the axisymmetric jet, three-dimensional round jet, and jet with streaks. The jet with streaks is obtained by adding the streaks with an azimuthal wavenumber of 6 to the mean flow of the round jet.

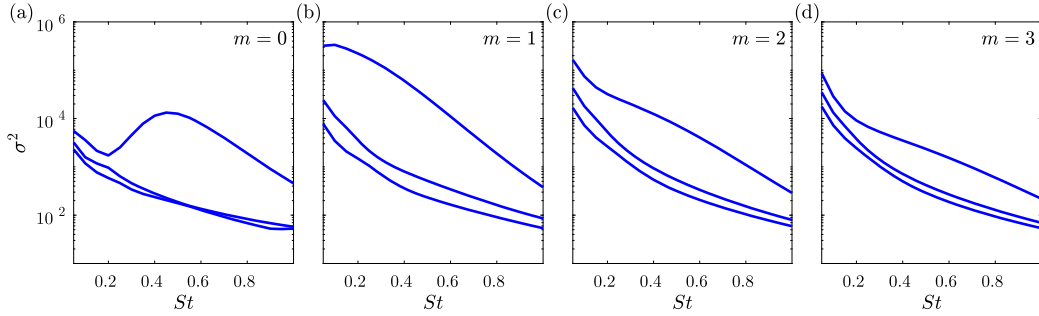


Fig. 8. Three leading gains of the axisymmetric jet for four azimuthal wavenumbers.

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 core region is more finely discretized than the downstream region, as we expect smaller structures within the area defined by $x \times y \times z \in [0, 5] \times [-1, 1] \times [-1, 1]$, which contains $200 \times 70 \times 70$ points. The weight matrix \mathbf{W} accounts for the non-uniform grid but does not reweight the flow variables (which are already non-dimensionalized). For more physically meaningful energy norms, one might consider a kinetic energy norm of Chu's compressible energy norm [20], both of which have been utilized in the literature [107,86,79]. The precise choice of norm is expected to have little impact on the results since the flow is subsonic [48,111]. The mean flow is obtained by revolving the axisymmetric mean flow around the streamwise axis, as depicted in Fig. 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, where $St = \frac{\omega D_j}{2\pi U_j}$ is the non-dimensional form of frequency. The range $St \in [0, 1]$ is wide enough to include the range of frequencies that are of interest in this jet [86]. Eddy viscosity can enhance the effectiveness of resolvent-based models by partially accounting for unmodeled Reynolds stresses [72,70]. One simple way to approximate this effect is by reducing the Reynolds number. Pickering et al. [70] found that $Re = 1000$ was effective for the same jet configuration considered here and we adopt this value in our paper. The effect of the Reynolds number on the resolvent gains and modes of a jet is thoroughly documented in Schmidt et al. [86].

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

$$\mathbf{q}'(x, y, z, t) = \sum_{\omega} \hat{\mathbf{q}}_{\omega}(x, y, z) e^{i\omega t}, \quad (51)$$

and each mode is characterized by its frequency ω .

To validate our three-dimensional results, we also perform an 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

$$\mathbf{q}'(x, r, \theta, t) = \sum_{m, \omega} \hat{\mathbf{q}}_{m, \omega}(x, r) e^{im\theta} e^{i\omega t} \quad (52)$$

is characterized by the pair (m, ω) , where m denotes azimuthal wavenumber. The domain of interest for resolvent analysis is $x \times r \in [0, 20] \times [0, 4]$, which captures the core flow region while being surrounded by a sponge layer to minimize boundary reflections. The boundary conditions in the sponge region are designed to absorb outgoing waves effectively, reducing reflections and numerical artifacts [53,85]. The domain is discretized using fourth-order summation-by-parts finite differences [59] with a 400×100 grid in the streamwise x and radial r directions, respectively. To improve accuracy, the grid resolution is higher in the core region where the primary flow features are concentrated. Note that we employed SBP differentiation operators but not an SAT implementation of

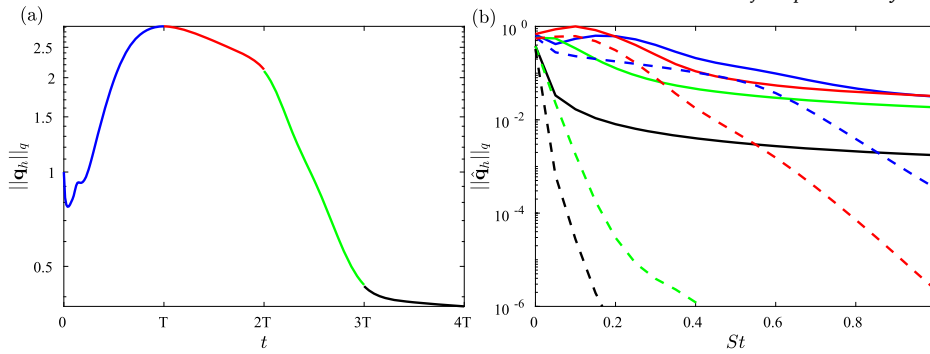


Fig. 9. Transient error estimates for the jet in (a) the time domain and (b) the frequency domain. Each colored period represents the duration utilized for obtaining norms in the frequency domain as shown in (b). Solid lines represent the natural decay, while dashed lines correspond to the transient removal strategy using Galerkin projection with the matrix of snapshots. The colors blue, red, green, and black correspond to the first, second, third, and fourth periods, respectively.

the boundary conditions. However, this choice has minimal impact on our results due to the sponge region and Dirichlet boundary conditions, which mitigates the influence of boundary conditions [1]. Similar to the three-dimensional case, a non-uniform grid setup is employed with a higher resolution in the core region. A grid-convergence study verifies the relative error between gains with this mesh and twice the number of grid points is less than 1% for $0 \leq St \leq 0.7$ and less than 10% for $0.7 < St \leq 1$. The larger errors at higher frequencies is due to small structures and could be eliminated by refining the mesh. However, we wish to keep the total size of the system as small as possible to keep the cost of the RSVD-LU algorithm manageable for comparison with our algorithm. The remaining parameters are kept the same as in the three-dimensional discretization of the jet.

Fig. 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. [86] and Pickering et al. [70]. The optimal mode when $m = 0, St \geq 0.2$ 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 $St \rightarrow 0$. Orr-type modes are characterized by perturbation structures that are initially oriented at an angle against the mean shear, leading to energy amplification through algebraic growth. These structures are subsequently reoriented along the direction of the mean shear as they evolve downstream. The resulting perturbation shape is distinctly tilted, with upstream portions leaning against the shear and downstream portions aligned with it, reflecting the interplay between shear-induced tilting and growth, as shown in the work of Pickering et al. [69]. 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. [86] and Pickering et al. [69] for the same wavenumbers demonstrating the robustness of the outcome even though the computational domains, Re , state vector, sponge regions, and boundary conditions are slightly different. The gains and corresponding modes of the axisymmetric jet are used as a baseline for comparison to the three-dimensional jet.

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 $q = 1$. The choice of k is determined by the number of desired modes, with a few additional test vectors included to enhance accuracy. The number of power iterations q may be increased if the modes at the frequencies of interest have not yet converged. In this case, an additional power iteration does not affect the results for $St \leq 0.7$, making $q = 1$ an effective choice. For the RSVD- Δt algorithm, we use the classical 4th order Runge-Kutta (RK4) integrator with $d t = 0.00625$. The steady-state interval is $T_s = 20$. Fig. 9 shows the expected transient error in the time and frequency domains. The transient initially grows in time before slowly decaying in Fig. 9(a). The resulting error in the frequency domain obtained from selecting each colored segment for computing resolvent modes is shown in Fig. 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 $St > 0$.

Fig. 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 [90]. Since higher wavenumbers ($m > 3$) have lower gains [70], 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. As a result, Fig. 10(a) displays three distinct curves, even though six modes are represented; each curve corresponds to one of the two symmetrical modes that overlap. The optimal gain at $m = 0$ is not captured in the three-dimensional analysis, as it is suboptimal and has lower energy compared to the other modes when $St \leq 0.3$ as shown in Fig. 10(a). 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; more ($k > 10$) resolvent modes would need to be computed to capture Orr modes that are buried

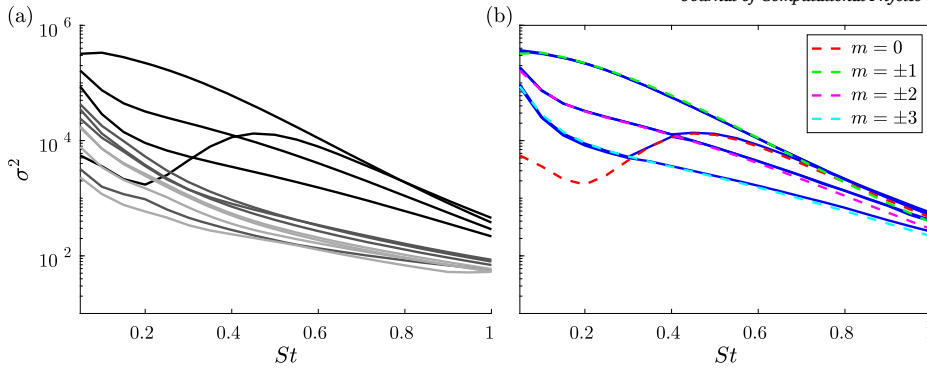


Fig. 10. Resolvent gains for the jet: (a) the union of the axisymmetric jet gains; (b) the optimal gains of the axisymmetric jet corresponding to various values of m (dashed lines) overlaid on top of the six leading gains for the three-dimensional discretization (solid lines). In (a), optimal modes for each m are colored black, gradually transitioning to grey for suboptimal modes.

beneath a slew of KH modes for each azimuthal wavenumber. The close match between the computed three-dimensional modes and the set of two-dimensional modes verifies that the three-dimensional calculations are properly capturing the known physics for this problem. The small mismatch at frequencies close to $St = 1$ is due to mild under-resolution of the grid for the compact structures that appear at these frequencies.

Fig. 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 three-dimensional mode, and contours for cross sections of the three-dimensional mode in the $x - y$ and $y - z$ planes. These images show the wavepacket form of the modes, confirm the classification of each three-dimensional mode with a particular azimuthal wavenumber, and illustrate the match between the symmetric and three-dimensional results. As noted by Martini et al. [58], 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 by varying the discretized state dimension N . We set $k = 10$, $N_\omega = 21$, and $q = 0$ for both algorithms and $dt = 0.00625$, $T_t = 2T_s$, and $T_s = 20$ in the RSVD- Δt algorithm as in §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 (27). The reported costs encompasses the entire RSVD- Δt algorithm with a total integration length of $4T_s$ per action, including one extra period to account for the transient removal strategy, as explained in §8.2. The RSVD- Δt algorithm is implemented using PETSc [9], while the LU decomposition in the RSVD-LU algorithm utilizes PETSc in conjunction with the MUMPS [4] external package. All calculations are performed on one Intel Xeon Gold 6154 processor on the University of Michigan's Great Lakes cluster, with wall time serving as a proxy for CPU time.

The measured CPU time for both algorithms are shown in Fig. 12(a) as a function of the state dimension N . The RSVD-LU algorithm scales poorly, in fact exceeding the theoretical scaling of $O(N^2)$ for three-dimensional flows (refer to §6) due to poor performance at low frequencies that has also been noted in other studies [69]. In contrast, the RSVD- Δt algorithm achieves (near) linear scaling, $O(N^{1.1})$, confirming its scalability to large problems. The calculations could not be performed using RSVD-LU for the largest two grids dimensions exceeding 1 million require an excessive amount of memory beyond our cluster limits of 3.5 TB when employing RSVD-LU. Therefore, these two data points are exclusively reserved for RSVD- Δt to validate the linear scaling retention during the transition to more realistic dimensions. The final point corresponds to the same grid utilized in our three-dimensional jet flow test case.

Similar observations can be made about the memory requirements of the two algorithms, shown in Fig. 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.

Most of the grids considered in Fig. 12 are under-resolved to make the RSVD-LU calculations tractable. To compare the two algorithms for a realistic grid, Table 3 compares the costs of RSVD-LU and RSVD- Δt for $N \approx 39$ million (5 state variables \times a $[400 \times 140^2]$ grid), which was used for the three-dimensional calculations in §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 Fig. 12. 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 Fig. 12. This implies that for $q = 1$, the CPU time includes a single LU decomposition and four times solving the LU-decomposed system. On the other hand, for RSVD- Δt ,

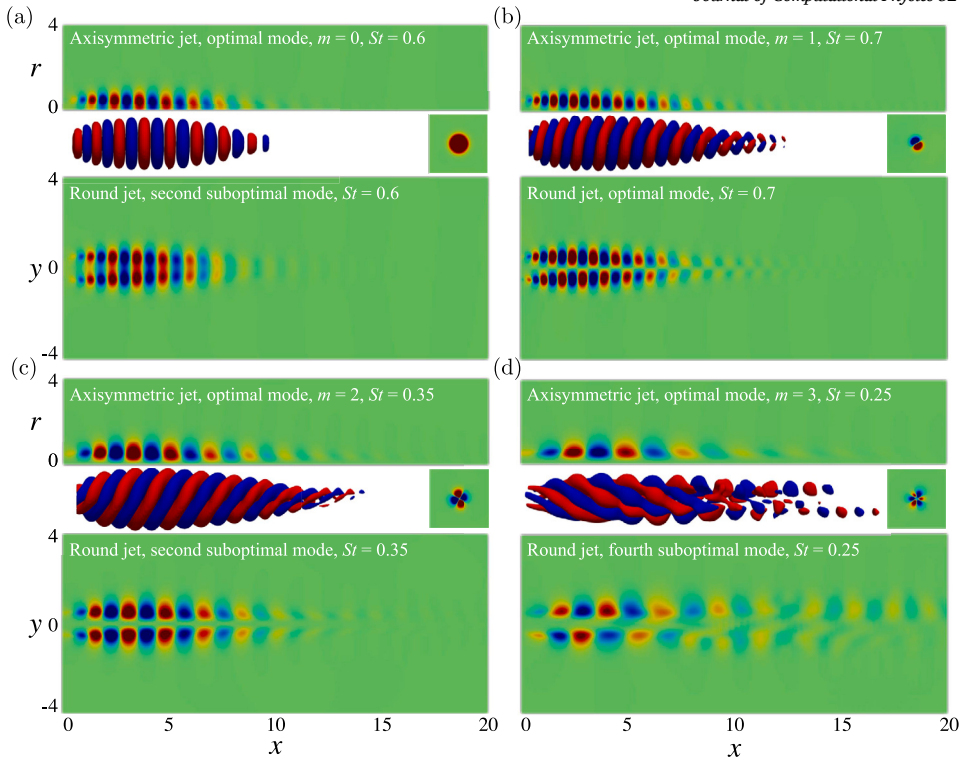


Fig. 11. Four groups of axisymmetric and three-dimensional pressure modes are shown, including axisymmetric views, three-dimensional iso-volume representations, and $x - y$ plane snapshots of the round jet. Cross-sections at $x = 5$ confirm the azimuthal wavenumber of the three-dimensional results. Color bar ranges are adjusted for visualization.

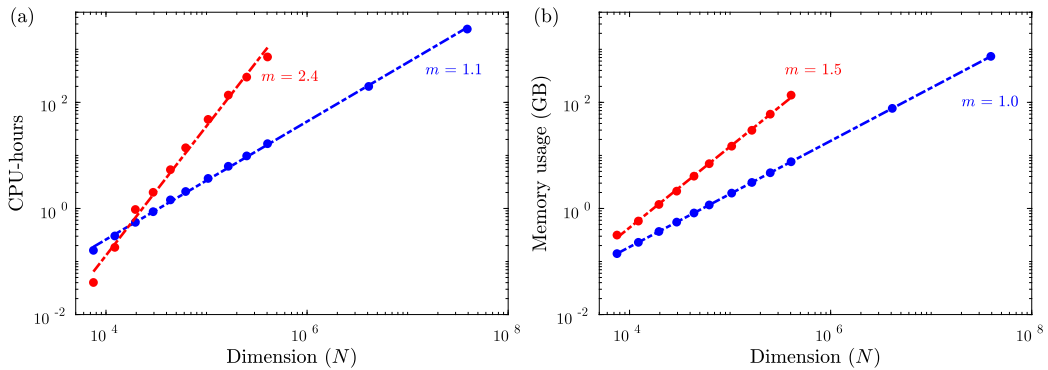


Fig. 12. Computational cost as a function of the state dimension N for the three-dimensional jet: (a) CPU-hours and (b) memory usage for the RSVD-LU (red) and RSVD- Δt (blue) algorithms.

the CPU time and memory usage are directly taken from our simulation, which employed 300 Intel Xeon Gold 6154 processors on the University of Michigan’s Great Lakes cluster.

The RSVD-LU algorithm exhibits a CPU time that is more than three orders of magnitude higher than that of the RSVD- Δt algorithm. Specifically, using 300 cores, the wall-time for RSVD- Δt is approximately 61 hours (< 3 days), while the RSVD-LU algorithm requires over 75 300 000 CPU-hours, which translates to around 251 000 hours (~ 28 years) wall-time, assuming perfect linear speed-up using 300 parallel cores. Of course, this wall-time can be brought down by increasing the number of cores, but it is clear that super-computing resources would be required to make the wall-time acceptable, which is antithetical to role of resolvent analysis as a reduced-order model. This disparity becomes even more pronounced as N increases due to the linear CPU scaling of RSVD- Δt and the quadratic scaling of the RSVD-LU algorithm for three-dimensional problems. Table 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 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-

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

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

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 [60] and Jiménez [42] 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 [43]. Nogueira et al. [65] and Pickering et al. [69] 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, streaks are shown to stabilize the KH wavepackets in a parallel plane shear layer [54] and Tollmien–Schlichting waves in the Blasius boundary layer [23]. Similar findings on a high-speed turbulent jet by Wang et al. [112] 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 [69]. 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. [112].

The updated mean flow with the streaks added has 6-fold rotational symmetry and, following Sinha et al. [87], can be written as

$$\tilde{q}(x, r, \theta) = \sum_{j=-\infty}^{\infty} \hat{q}_{lj}(x, r) e^{ilj\theta}. \tag{53}$$

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 $\mathbf{q}_\omega^L = \{\hat{q}_{L-lj,\omega}\}_{j=-\infty}^{l=\infty}$, 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 shape of the streaks is 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. [70] 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. [65].

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 Fig. 7). The amplitude is defined as the ratio of the peak streamwise velocity of streaks over the maximum velocity at the nozzle exit. This serves as a free parameter that can be investigated across various values. According to Wang et al. [112], 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.

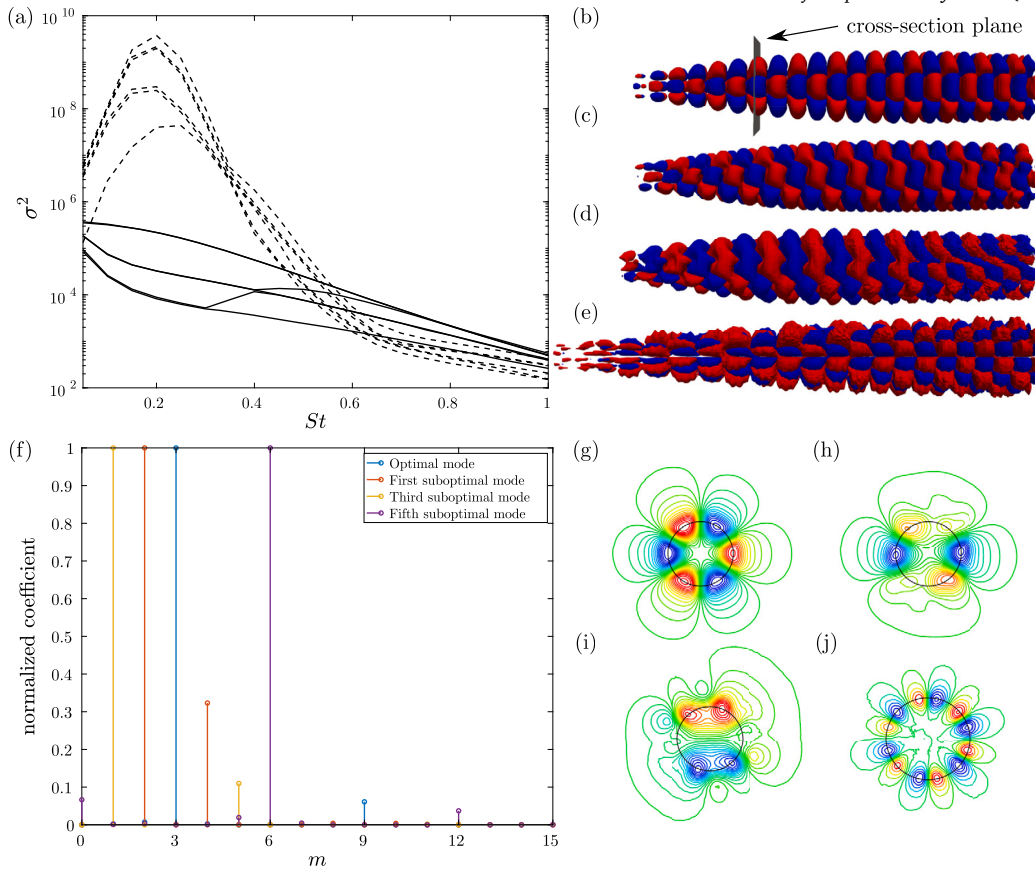


Fig. 13. Results for the jet with streaks: (a) resolvent gains for the round jet (solid line) and jet with streaks (dashed line); (b-e) the optimal, first, third, and fifth suboptimal pressure responses at $St = 0.2$; (g-j) contours of the pressure responses on cross-section at $x = 8.5$ corresponding 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).

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. For the same reasons mentioned for the round jet, we use 10 test vectors and are interested in computing the top six leading modes. Regarding the number of 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 our transient removal strategy using Galerkin projection with the matrix of snapshots with a duration $T_t = 3T_s$.

The gains for the round jet and jet with streaks are compared in Fig. 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 the leading gain is maximum, are shown in Fig. 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 $L + l = 3 + 6 = 9$ with a smaller weight, which is consistent with our axisymmetric analysis. The first suboptimal mode includes $(L, L - l) = (2, -4)$, and its pair contains $(L, L + l) = (-2, 4)$, so both couplings and pairings are as expected. Similarly, the third mode is a coupling between $(L, L - l) = (1, -5)$, and the fourth mode is with $(L, L + l) = (-1, 5)$. Lastly, the fifth mode is unpaired and captures the $(L, L + l) = (0, 6)$ azimuthal wavenumbers with a small signature of $L + 2l = 12$.

From the perspective of computational cost, the jet with streaks is similar to the three-dimensional discretization of the round jet. 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. As a point of comparison, the LES of a similar jet with $N \approx 64$ million consumed 464 thousand CPU hours [14]. 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 LU decomposition of $(i\omega \mathbf{I} - \mathbf{A})$ 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 $(i\omega \mathbf{I} - \mathbf{A})$ is the primary contributor to peak memory usage in the RSVD-LU and other common algorithms. In contrast, the RSVD- Δt algorithm primarily utilizes RAM to store input and output matrices in Fourier space, resulting in linear growth of memory consumption with dimension. To minimize the required memory, we utilize streaming calculations, which maintains low memory requirements with minimal computational impact. If memory limitations persist, the set of desired frequencies can be split into d groups to further reduce the required memory by a factor of d .

The RSVD- Δt algorithm contains three sources of error, each of which can be controlled by carefully selecting method parameters. The first arises from the RSVD approximation inherited from the RSVD algorithm. This error can be significantly reduced by employing power iteration and utilizing more test vectors than the desired number. The second source of error stems from the time integration method employed to compute the action of \mathbf{R} and \mathbf{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. Using a procedure analogous to 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 the RSVD- Δt algorithm are amenable to efficient parallelization; we have optimized out implementation of the algorithm for parallel computing using the PETSc [9] and SLEPc [34] 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. [73] and House et al. [36] can be integrated into our approach to use physical insight to select the initial test vectors instead of relying on entirely random ones. An open-source implementation of the RSVD- Δt algorithm is available on GitHub (<https://github.com/AliFarghadan/RSVD-Delta-t>).

CRedit authorship contribution statement

Ali Farghadan: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Eduardo Martini:** Writing – review & editing, Methodology. **Aaron Towne:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to express our gratitude to André Cavaliere 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 (8) is equivalent to computing the SVD of the weighted resolvent operator [107]

$$\tilde{\mathbf{R}} = \mathbf{W}_q^{1/2} \mathbf{C} (i\omega \mathbf{I} - \mathbf{A})^{-1} \mathbf{B} \mathbf{W}_f^{-1/2}, \quad (\text{A.1a})$$

$$\tilde{\mathbf{R}} = \tilde{\mathbf{U}} \tilde{\Sigma} \tilde{\mathbf{V}}^*, \quad (\text{A.1b})$$

and further

$$\mathbf{U} = \mathbf{W}_q^{-1/2} \tilde{\mathbf{U}}, \quad (\text{A.2})$$

$$\mathbf{V} = \mathbf{W}_f^{-1/2} \tilde{\mathbf{V}},$$

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

$$\mathbf{R} = \mathbf{U} \Sigma \mathbf{V}^* \mathbf{W}_f. \quad (\text{A.3})$$

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

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

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

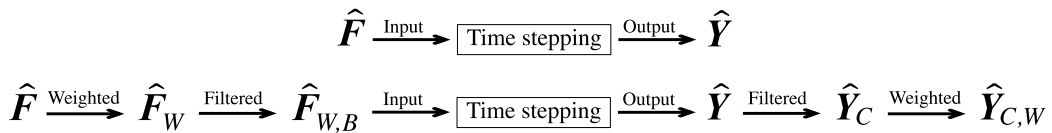


Fig. A.14. The schematic of computing the action of \mathbf{R} on top and the action of $\tilde{\mathbf{R}}$ on the bottom row.

Appendix B. Removing the least-damped modes using eigenvalues only

The transient removal strategies described in §8.2 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 alternative procedure to expedite the decay of transients that uses

knowledge of the least-damped eigenvalues themselves. Considering two solutions of (17), $\mathbf{q}_1 = \mathbf{q}(t_1)$ and $\mathbf{q}_2 = \mathbf{q}(t_1 + \Delta t)$, we can express them in terms of their steady-state and transient parts as

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

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

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

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

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

where $\mathbf{q}_{\lambda_1,j}$ is essentially the projection of the transient response onto the least-damped eigenmode of \mathbf{A} at $t = t_j$. The steady-state evolution at a prescribed forcing at a single frequency ω follows (41). Therefore, in case of $\|\mathbf{q}_{rest,j}\| = 0$, the system of equations is deterministic and $\mathbf{q}_{t,1}$ can be found as

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

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

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

Based on the fundamental assumption, the second term, which is unknown, decays faster than $e^{\lambda_1 t}$. Therefore, by removing the first term $\frac{\mathbf{b}}{c}$, 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 $\frac{\mathbf{b}}{c}$ 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 \mathbf{A} . The solution at each time with arbitrary distances can be expanded as

$$\mathbf{q}_l = \mathbf{q}_{s,l} + \sum_{j=1}^d \mathbf{q}_{\lambda_j,l} + \mathbf{q}_{rest,l}, \tag{B.6}$$

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.

Data availability

Data will be made available on request.

References

- [1] R. Abgrall, J. Nordström, P. Öffner, S. Tokareva, Analysis of the sbp-sat stabilization for finite element methods part I: linear problems, *J. Sci. Comput.* 85 (2020) 1–29.
- [2] R.J. Adrian, Hairpin vortex organization in wall turbulence, *Phys. Fluids* 19 (2007).
- [3] P.R. Amestoy, A. Buttari, J.Y. l'Excellent, T.A. Mary, Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel block low-rank format, *SIAM J. Sci. Comput.* 41 (2019) A1414–A1442.
- [4] P.R. Amestoy, I.S. Duff, J.Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM J. Matrix Anal. Appl.* 23 (2001) 15–41.
- [5] E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al., *LAPACK Users' Guide*, SIAM, 1999.
- [6] W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, *Q. Appl. Math.* 9 (1951) 17–29.
- [7] O. Axelsson, A survey of preconditioned iterative methods for linear systems of algebraic equations, *BIT Numer. Math.* 25 (1985) 165–187.
- [8] S. Bagheri, D.S. Henningson, J. Hoepffner, P.J. Schmid, Input-output analysis and control design applied to a linear model of spatially developing flows, *Appl. Mech. Rev.* 62 (2009).
- [9] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., *PETSc users manual*, Argonne National Laboratory, 2019.
- [10] B. Barthel, S. Gomez, B.J. McKeon, Variational formulation of resolvent analysis, *Phys. Rev. Fluids* 7 (2022) 013905.
- [11] M. Benzi, Preconditioning techniques for large linear systems: a survey, *J. Comput. Phys.* 182 (2002) 418–477.

- [12] J. Berg, J. Nordström, Superconvergent functional output for time-dependent problems using finite differences on summation-by-parts form, *J. Comput. Phys.* 231 (2012) 6846–6860.
- [13] G.A. Brès, F.E. Ham, J.W. Nichols, S.K. Lele, Unstructured large-eddy simulations of supersonic jets, *AIAA J.* 55 (2017) 1164–1184.
- [14] G.A. Brès, P. Jordan, V. Jaunet, M. Le Rallic, A.V.G. Cavalieri, A. Towne, S.K. Lele, T. Colonius, O.T. Schmidt, Importance of the nozzle-exit boundary-layer state in subsonic turbulent jets, *J. Fluid Mech.* 851 (2018) 83–124.
- [15] M. Brynjell-Rahkola, L.S. Tuckerman, P. Schlatter, D.S. Henningson, Computing optimal forcing using Laplace preconditioning, *Commun. Comput. Phys.* 22 (2017) 1508–1532.
- [16] M.H. Carpenter, J. Nordström, D. Gottlieb, A stable and conservative interface treatment of arbitrary spatial accuracy, *J. Comput. Phys.* 148 (1999) 341–365.
- [17] A.V.G. Cavalieri, P. Jordan, L. Lesshafft, Wave-packet models for jet dynamics and sound radiation, *Appl. Mech. Rev.* 71 (2019).
- [18] A. Chavarin, M. Lohar, Resolvent analysis for turbulent channel flow with riblets, *AIAA J.* 58 (2020) 589–599.
- [19] K.K. Chen, C.W. Rowley, H2 optimal actuator and sensor placement in the linearised complex Ginzburg–Landau system, *J. Fluid Mech.* 681 (2011) 241–260.
- [20] B.T. Chu, On the energy transfer to small disturbances in fluid flow (part I), *Acta Mech.* 1 (1965) 215–234.
- [21] T. Colonius, Modeling artificial boundary conditions for compressible flow, *Annu. Rev. Fluid Mech.* 36 (2004) 315–345.
- [22] D.A. Cook, J.W. Nichols, Three-dimensional receptivity of hypersonic sharp and blunt cones to free-stream planar waves using hierarchical input-output analysis, arXiv preprint, arXiv:2306.03248, 2023.
- [23] C. Cossu, L. Brandt, Stabilization of Tollmien–Schlichting waves by finite amplitude optimal streaks in the Blasius boundary layer, *Phys. Fluids* 14 (2002) L57–L60.
- [24] T.A. Davis, S. Rajamanickam, W.M. Sid-Lakhdar, A survey of direct methods for sparse linear systems, *Acta Numer.* 25 (2016) 383–566.
- [25] S.T.M. Dawson, B.J. McKeon, On the shape of resolvent modes in wall-bounded turbulence, *J. Fluid Mech.* 877 (2019) 682–716.
- [26] I.S. Duff, A.M. Erisman, J.K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, 2017.
- [27] N. Dunford, J.T. Schwartz, *Linear Operators Part I: General Theory*, John Wiley & Sons, 1958.
- [28] W.S. Edwards, L.S. Tuckerman, R.A. Friesner, D.C. Sorensen, Krylov methods for the incompressible Navier-Stokes equations, *J. Comput. Phys.* 110 (1994) 82–102.
- [29] L.E. Eriksson, A. Rizzi, Computer-aided analysis of the convergence to steady state of discrete approximations to the Euler equations, *J. Comput. Phys.* 57 (1985) 90–128.
- [30] R.D. Falgout, U.M. Yang, Hypre: a library of high performance preconditioners, in: *International Conference on Computational Science*, 2002, pp. 632–641.
- [31] F. Gómez, A.S. Sharma, H.M. Blackburn, Estimation of unsteady aerodynamic forces using pointwise velocity data, *J. Fluid Mech.* 804 (2016) R4.
- [32] E. Hairer, S. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer, Berlin Heidelberg, 1993.
- [33] N. Halko, P. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Rev.* 53 (2011) 217–288.
- [34] V. Hernandez, J.E. Roman, V. Vidal, Slep: a scalable and flexible toolkit for the solution of eigenvalue problems, *ACM Trans. Math. Softw.* 31 (2005) 351–362.
- [35] B. Herrmann, P.J. Baddoo, R. Semaan, S.L. Brunton, B.J. McKeon, Data-driven resolvent analysis, *J. Fluid Mech.* 918 (2021) A10.
- [36] D. House, C. Skene, J.H.M. Ribeiro, C.A. Yeh, K. Taira, Sketch-based resolvent analysis, *AIAA Paper #2022-3335*, 2022.
- [37] J. Houtman, S. Timme, A. Sharma, Resolvent analysis of a finite wing in transonic flow, *Flow* 3 (2023) E14.
- [38] F.Q. Hu, Development of pml absorbing boundary conditions for computational aeroacoustics: a progress review, *Comput. Fluids* 37 (2008) 336–348.
- [39] R.E. Hunt, D.G. Crighton, Instability of flows in spatially developing media, *Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci.* 435 (1991) 109–128.
- [40] N. Hutchins, I. Marusic, Large-scale influences in near-wall turbulence, *Philos. Trans. - Royal Soc. A, Math. Phys. Eng. Sci.* 365 (2007) 647–664.
- [41] J. Jeun, J.W. Nichols, M.R. Jovanović, Input-output analysis of high-speed axisymmetric isothermal jet noise, *Phys. Fluids* 28 (2016) 047101.
- [42] J. Jiménez, Coherent structures in wall-bounded turbulence, *J. Fluid Mech.* 842 (2018) P1.
- [43] J. Jimenez-Gonzalez, P. Brancher, Transient energy growth of optimal streaks in parallel round jets, *Phys. Fluids* 29 (2017) 114101.
- [44] P. Jordan, T. Colonius, Wave packets and turbulent jet noise, *Annu. Rev. Fluid Mech.* 45 (2013) 173–195.
- [45] M.R. Jovanovic, *Modeling, Analysis, and Control of Spatially Distributed Systems*, University of California, Santa Barbara, 2004.
- [46] M.R. Jovanović, From bypass transition to flow control and data-driven turbulence modeling: an input-output viewpoint, *Annu. Rev. Fluid Mech.* 53 (2021) 010719.
- [47] O. Kamal, M.T. Lakebrink, T. Colonius, Global receptivity analysis: physically realizable input–output analysis, *J. Fluid Mech.* 956 (2023) R5.
- [48] U. Karban, B. Bugeat, E. Martini, A. Towne, A.V.G. Cavalieri, L. Lesshafft, A. Agarwal, P. Jordan, T. Colonius, Ambiguity in mean-flow-based linear analysis, *J. Fluid Mech.* 900 (2020) R5.
- [49] T. Kato, *Perturbation Theory for Linear Operators*, Springer Science & Business Media, 2013.
- [50] L. Lesshafft, O. Semeraro, V. Jaunet, A.V.G. Cavalieri, P. Jordan, Resolvent-based modeling of coherent wave packets in a turbulent jet, *Phys. Rev. Fluids* 4 (2019) 063901.
- [51] F. Li, M.R. Malik, On the nature of PSE approximation, *Theor. Comput. Fluid Dyn.* 8 (1996) 253–273.
- [52] J.L. Lumley, The structure of inhomogeneous turbulent flows, *Atmos. Turbul. Radio Wave Propag.* (1967) 166–178.
- [53] A. Mani, Analysis and optimization of numerical sponge layers as a nonreflective boundary treatment, *J. Comput. Phys.* 231 (2012) 704–716.
- [54] M. Marant, C. Cossu, Influence of optimally amplified streamwise streaks on the Kelvin–Helmholtz instability, *J. Fluid Mech.* 838 (2018) 478–500.
- [55] O. Marquet, M. Larsson, Global wake instabilities of low aspect-ratio flat-plates, *Eur. J. Mech. B, Fluids* 49 (2015) 400–412.
- [56] E. Martini, A.V.G. Cavalieri, P. Jordan, A. Towne, L. Lesshafft, Resolvent-based optimal estimation of transitional and turbulent flows, *J. Fluid Mech.* 900 (2020) A2.
- [57] E. Martini, J. Jung, A.V.G. Cavalieri, P. Jordan, A. Towne, Resolvent-based tools for optimal estimation and control via the Wiener–Hopf formalism, *J. Fluid Mech.* 938 (2022) E2.
- [58] E. Martini, D. Rodríguez, A. Towne, A.V.G. Cavalieri, Efficient computation of global resolvent modes, *J. Fluid Mech.* 919 (2021) A3.
- [59] K. Mattsson, J. Nordström, Summation by parts operators for finite difference approximations of second derivatives, *J. Comput. Phys.* 199 (2004) 503–540.
- [60] B.J. McKeon, The engine behind (wall) turbulence: perspectives on scale interactions, *J. Fluid Mech.* 817 (2017) P1.
- [61] B.J. McKeon, A.S. Sharma, A critical-layer framework for turbulent pipe flow, *J. Fluid Mech.* 658 (2010) 336–382.
- [62] R. Moarref, A.S. Sharma, J.A. Tropp, B.J. McKeon, Model-based scaling of the streamwise energy density in high-Reynolds-number turbulent channels, *J. Fluid Mech.* 734 (2013) 275–316.
- [63] A. Monokrousos, E. Åkervik, L. Brandt, D.S. Henningson, Global three-dimensional optimal disturbances in the Blasius boundary-layer flow using time-steppers, *J. Fluid Mech.* 650 (2010) 181–214.
- [64] P. Morra, O. Semeraro, D.S. Henningson, C. Cossu, On the relevance of Reynolds stresses in resolvent analyses of turbulent wall-bounded flows, *J. Fluid Mech.* 867 (2019) 969–984.
- [65] P.A.S. Nogueira, A.V.G. Cavalieri, P. Jordan, V. Jaunet, Large-scale streaky structures in turbulent jets, *J. Fluid Mech.* 873 (2019) 211–237.
- [66] H. Nyquist, Certain topics in telegraph transmission theory, *Trans. Am. Inst. Electr. Eng.* 47 (1928) 617–644.
- [67] M.F. de Pando, D. Sipp, P.J. Schmid, Efficient evaluation of the direct and adjoint linearized dynamics from compressible flow solvers, *J. Comput. Phys.* 231 (2012) 7739–7755.
- [68] J.W. Pearson, J. Pestana, Preconditioners for Krylov subspace methods: an overview, *GAMM-Mitt.* 43 (2020) e202000015.

- [69] E. Pickering, G. Rigas, P.A.S. Nogueira, A.V.G. Cavalieri, O.T. Schmidt, T. Colonius, Lift-up, Kelvin–Helmholtz and Orr mechanisms in turbulent jets, *J. Fluid Mech.* 896 (2020) A2.
- [70] E. Pickering, G. Rigas, O.T. Schmidt, D. Sipp, T. Colonius, Optimal eddy viscosity for resolvent-based models of coherent structures in turbulent jets, *J. Fluid Mech.* 917 (2021) A29.
- [71] W. Ran, A. Zare, M.R. Jovanović, Model-based design of riblets for turbulent drag reduction, *J. Fluid Mech.* 906 (2021) A7.
- [72] W.C. Reynolds, A.K.M.F. Hussain, The mechanics of an organized wave in turbulent shear flow. Part 3. Theoretical models and comparisons with experiments, *J. Fluid Mech.* 54 (1972) 263–288.
- [73] J.H.M. Ribeiro, C.A. Yeh, K. Taira, Randomized resolvent analysis, *Phys. Rev. Fluids* 5 (2020) 033902.
- [74] L.V. Rolandi, J.H.M. Ribeiro, C.A. Yeh, K. Taira, An invitation to resolvent analysis, *Theor. Comput. Fluid Dyn.* (2024) 1–37.
- [75] Y. Saad, Finding exact and approximate block structures for ilu preconditioning, *SIAM J. Sci. Comput.* 24 (2003) 1107–1123.
- [76] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
- [77] K. Sasaki, A.V.G. Cavalieri, A. Hanifi, D.S. Henningson, Parabolic resolvent modes for streaky structures in transitional and turbulent boundary layers, *Phys. Rev. Fluids* 7 (2022) 104611.
- [78] O. Schenk, K. Gärtner, W. Fichtner, A. Stricker, Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation, *Future Gener. Comput. Syst.* 18 (2001) 69–78.
- [79] R.K. Schlander, S. Rigopoulos, G. Papadakis, Resolvent analysis of turbulent flow laden with low-inertia particles, *J. Fluid Mech.* 985 (2024) A27.
- [80] P.J. Schmid, Nonmodal stability theory, *Annu. Rev. Fluid Mech.* 39 (2007) 129–162.
- [81] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, *J. Fluid Mech.* 656 (2010) 5–28.
- [82] P.J. Schmid, Dynamic mode decomposition and its variants, *Annu. Rev. Fluid Mech.* 54 (2022) 225–254.
- [83] P.J. Schmid, D.S. Henningson, *Stability and Transition in Shear Flows*, Springer, New York, 2001.
- [84] O.T. Schmidt, A. Towne, An efficient streaming algorithm for spectral proper orthogonal decomposition, *Comput. Phys. Commun.* 237 (2019) 98–109.
- [85] O.T. Schmidt, A. Towne, T. Colonius, A.V.G. Cavalieri, P. Jordan, G.A. Brès, Wavepackets and trapped acoustic modes in a turbulent jet: coherent structure education and global stability, *J. Fluid Mech.* 825 (2017) 1153–1181.
- [86] O.T. Schmidt, A. Towne, G. Rigas, T. Colonius, G.A. Brès, Spectral analysis of jet turbulence, *J. Fluid Mech.* 855 (2018) 953–982.
- [87] A. Sinha, K. Gudmundsson, H. Xia, T. Colonius, Parabolized stability analysis of jets from serrated nozzles, *J. Fluid Mech.* 789 (2016) 36–63.
- [88] D. Sipp, O. Marquet, Characterization of noise amplifiers with global singular modes: the case of the leading-edge flat-plate boundary layer, *Theor. Comput. Fluid Dyn.* 27 (2013) 617–635.
- [89] L. Sirovich, Turbulence and the dynamics of coherent structures. I. Coherent structures, *Q. Appl. Math.* 45 (1987) 561–571.
- [90] L. Sirovich, Turbulence and the dynamics of coherent structures. II. Symmetries and transformations, *Q. Appl. Math.* 45 (1987) 573–582.
- [91] R.D. Skeel, Scaling for numerical stability in Gaussian elimination, *J. ACM* 26 (1979) 494–526.
- [92] G.W. Stewart, On the early history of the singular value decomposition, *SIAM Rev.* 35 (1993) 551–566.
- [93] G.W. Stewart, *Perturbation Theory for the Singular Value Decomposition*, Citeseer, 1998.
- [94] M. Stewart, Perturbation of the svd in the presence of small singular values, *Linear Algebra Appl.* 419 (2006) 53–77.
- [95] K. Stewartson, J.T. Stuart, A non-linear instability theory for a wave system in plane Poiseuille flow, *J. Fluid Mech.* 48 (1971) 529–545.
- [96] E. Süli, D.F. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.
- [97] S. Symon, D. Sipp, B.J. McKeon, A tale of two airfoils: resolvent-based modelling of an oscillator versus an amplifier from an experimental mean, *J. Fluid Mech.* 881 (2019) 51–83.
- [98] K. Taira, S.L. Brunton, S.T.M. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordyeyev, V. Theofilis, L.S. Ukeiley, Modal analysis of fluid flows: an overview, *AIAA J.* 55 (2017) 4013–4041.
- [99] V. Theofilis, Global linear instability, *Annu. Rev. Fluid Mech.* 43 (2011) 319–352.
- [100] N. Thomareis, G. Papadakis, Resolvent analysis of separated and attached flows around an airfoil at transitional Reynolds number, *Phys. Rev. Fluids* 3 (2018) 073901.
- [101] A. Towne, *Advancements in jet turbulence and noise modeling: accurate one-way solutions and empirical evaluation of the nonlinear forcing of wavepackets*, Ph.D. thesis, California Institute of Technology, 2016.
- [102] A. Towne, T. Colonius, One-way spatial integration of hyperbolic equations, *J. Comput. Phys.* 300 (2015) 844–861.
- [103] A. Towne, T. Colonius, P. Jordan, A.V. Cavalieri, G.A. Brès, Stochastic and nonlinear forcing of wavepackets in a Mach 0.9 jet, *AIAA Paper #2015-2217*, 2015.
- [104] A. Towne, A. Lozano-Durán, X. Yang, Resolvent-based estimation of space–time flow statistics, *J. Fluid Mech.* 883 (2020) A17.
- [105] A. Towne, G. Rigas, T. Colonius, A critical assessment of the parabolized stability equations, *Theor. Comput. Fluid Dyn.* 33 (2019) 359–382.
- [106] A. Towne, G. Rigas, O. Kamal, E. Pickering, T. Colonius, Efficient global resolvent analysis via the one-way Navier–Stokes equations, *J. Fluid Mech.* 948 (2022) A9.
- [107] A. Towne, O.T. Schmidt, T. Colonius, Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis, *J. Fluid Mech.* 847 (2018) 821–867.
- [108] L.N. Trefethen, D. Bau III, *Numerical Linear Algebra*, Siam, 1997.
- [109] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, Cambridge University Press, 2018.
- [110] R. Vishnampet, D.J. Bodony, J.B. Freund, A practical discrete-adjoint method for high-fidelity compressible turbulence simulations, *J. Comput. Phys.* 285 (2015) 173–192.
- [111] E.A. Vogel, J.G. Coder, A novel entropy normalization scheme for characterization of highly compressible flows, *Theor. Comput. Fluid Dyn.* 36 (2022) 641–670.
- [112] C. Wang, L. Lesshafft, A.V.G. Cavalieri, P. Jordan, The effect of streaks on the instability of jets, *J. Fluid Mech.* 910 (2021) A14.
- [113] G. Wanner, E. Hairer, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, Berlin Heidelberg, 1996.
- [114] C.A. Yeh, S.I. Benton, K. Taira, D.J. Garmann, Resolvent analysis of an airfoil laminar separation bubble at $Re = 500\,000$, *Phys. Rev. Fluids* 5 (2020) 083906.
- [115] C.A. Yeh, K. Taira, Resolvent-analysis-based design of airfoil separation control, *J. Fluid Mech.* 867 (2019) 572–610.
- [116] Z. Zhou, C. Xu, J. Jiménez, Interaction between near-wall streaks and large-scale motions in turbulent channel flows, *J. Fluid Mech.* 940 (2022) A23.
- [117] M. Zhu, A. Towne, Recursive one-way Navier–Stokes equations with PSE-like cost, *J. Comput. Phys.* 473 (2023) 111744.