Mechanical Systems and Signal Processing 237 (2025) 112817

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

Differentiable singular value decomposition (SVD)

Rohit Sunil Kanchi *, Sicheng He

Department of Mechanical, Aerospace, and Biomedical Engineering, University of Tennessee, Knoxville, TN 37996, USA

ARTICLE INFO

Communicated by X. Jing

Keywords:
Singular value decomposition
Adjoint method
Reverse automatic differentiation
Optimization
Sensitivity analysis

ABSTRACT

Singular value decomposition (SVD) is widely used in modal analysis, such as proper orthogonal decomposition (POD) and resolvent analysis, to extract key features from imaginary problems. SVD derivatives need to be computed efficiently to enable the large-scale design optimization. However, for a general imaginary matrix, no method can accurately compute this derivative to machine precision and remain scalable with respect to the number of design variables without requiring the all of the singular variables. We propose two algorithms to efficiently compute this derivative based on the adjoint method and reverse automatic differentiation (RAD) and RAD-based singular value derivative formula. Differentiation results for each method proposed were compared with FD results for one square and one tall rectangular matrix example and matched with the FD results to about 5–7 digits. Finally, we demonstrate the scalability of the proposed method by calculating the derivatives of singular values with respect to the snapshot matrix derived from the POD of a large dataset for a laminar–turbulent transitional flow over a flat plate, sourced from the John Hopkins turbulence database (JHTDB).

1. Introduction

Singular variable decomposition (SVD) is a fundamental linear algebra technique widely used in engineering applications [1,2]. SVD is applied in varied fields such as numerical optimization [3], mechanical design [4–8], fluid dynamics [9,10], noise-reduction [11–13], image processing [14], and finance [15]. Applications of SVD in engineering optimization and analysis problems include aerostructural optimization [16–18], aerodynamic shape optimization [19–24], vibration analysis [25,26] and pure structural optimization problems [27–30]. Many modal analysis methods are based on SVD, e.g., the resolvent analysis [31], and dynamic mode decomposition (DMD) [32]. For more details about SVD in modal analysis, see the review papers by Taira et al. [33,34].

There are several routes to compute SVD of a given matrix (refer to Golub and Van Loan [35, Chapter 8.6] for more details). A popular approach is to reduce the problem to an eigenvalue problem (EVP) [35,36] Once that is done, numerical methods such as Golub–Kahan bi-diagonalization [35] and the Jacobi SVD algorithm [37] can be used to compute the SVD. Iterative but approximating methods such as Lanczos or Krylov subspace methods [38], randomized SVD [39], and truncated SVD have also been developed to reduce computational costs, particularly for large-scale data or high-dimensional systems.

These advancements are crucial when differentiation of the SVD is needed, as the computational cost and numerical stability of differentiation depend heavily on the underlying method used to compute the SVD. This motivates the need to carefully evaluate SVD computation strategies for their efficiency and compatibility with different differentiation frameworks. For a more thorough review of SVD history and algorithms, especially the relationship between EVP and SVD, we direct readers to the review by Zhang [40] and Lecture 31 of Trefethen and Bau [36].

* Corresponding author.
E-mail address: rkanchi@vols.utk.edu (R.S. Kanchi).

https://doi.org/10.1016/j.ymssp.2025.112817
Received 15 January 2025; Received in revised form 28 March 2025; Accepted 30 April 2025

Available online 16 June 2025

Two approaches in the relationship between the EVP of a symmetric matrix and the SVD were leveraged in the current study. In the first approach, we perform eigendecomposition of the product of the matrix and its Hermitian transpose (if real, just transpose), and the reverse product [35,36,41] The resulting matrices from this pair of products are the Gram matrices [36,42,43] of the matrix under consideration whose SVD is sought, or in other words, the target matrix. The eigendecomposition of the Gram matrix resulting from the product of the target matrix and its Hermitian transpose gives the singular values and the left singular vectors of the SVD. Here, the eigenvectors are the left singular vectors, and the square roots of the eigenvalues are the singular values. Similarly, the eigendecomposition of the Gram matrix from the product of the reverse product of the target matrix and its Hermitian transpose yields the singular values and the right singular vectors of the target matrix [35]. We developed governing equations for this approach based on the approach taken by He et al. [44]. This approach is useful when the matrix whose SVD is sought is sparse. If it is dense, then forming the left or right Gram matrices is costly, which motivated the second approach.

In the second approach, we leverage the relationship between the SVD of the target matrix and the EVP of its symmetric embedding [35,45]. The relationship is such that the singular value and its negative counterpart are eigenvalues of the symmetric embedding. The eigenvectors of the symmetric embedding contain the left and right singular vectors of the target matrix. We developed a set of SVD governing equations in the current manuscript based on this symmetric embedding which can be solved using an alternating least squares approach with Rayleigh quotient through successive deflation [35,36,46].

The derivative computation methods proposed in the current manuscript are not limited to only the EVP-SVD approach to compute the singular variables. So long as the singular variables computed through any algorithm satisfy the governing equations for either aforementioned case, the proposed derivative algorithms will work. However, there arise problems whose solutions might involve nearly similar or duplicated singular values [35,47,48]. Since the singular values are the square roots of the eigenvalues of the Gram matrices, the Gram matrices EVPs yield repeated eigenvalues in the case of duplicate singular values. Typically, such duplicated or close eigenvalues occur due to some spatial symmetry in the problem whose eigen-solution is sought [49,50]. It is a fact that the changes of the order of ϵ in a matrix can alter its singular subspace by δ/ϵ where δ is a measure of how far apart the singular values are from each other (refer to Golub and Van Loan [35, Chapter 8.6.1]). Thus, if two singular values are close or the same, the ratio would tend to infinitely large values. This means that the singular vector space is mathematically degenerate [50]. Angelova and Petkov [48] showed in a component-wise perturbation analysis that the SVD perturbation problem is only well-posed when the singular values are distinct. The distinct singular values, therefore, ensure uniqueness in solution to the governing equations developed. This case of duplicate singular values is, however, beyond the scope of the current manuscript.

Once the required singular variables are obtained from either of the two EVP-SVD approaches mentioned, the derivative of the singular variables is sought, which has applications in different design optimization problems. For instance, Ersoy and Mugan [29] computed the derivatives of the singular values and vectors for the structural design derivative analysis. Santiago et al. [51] took the derivatives of the singular values of the shapes in the shape derivative as part of the shape optimization for photonic nanostructures. Recently, Skene and Schmid [52] performed a mixed forward and backward derivative analysis for the swirling M-flames, which involved computation of the resolvent with respect to the design parameters. The resolvent analysis involved the singular value decomposition of the resolvent matrix, which gave the dominant singular value whose derivative was sought.

Various methods are used for computing derivatives, including finite differences (FD), imaginary step (CS), algorithmic differentiation (AD), the direct method, and the adjoint method (refer to Martins and Ning [53, Chapter 6]). These methods differ in terms of their accuracy and computational efficiency. Efficiency-wise, each method is typically optimized for either scaling with the number of outputs (e.g., functions of interest such as eigenvalues and eigenvectors) or the number of inputs (e.g., design variables), but rarely both [54],[53, Chapter 6].

FD is susceptible to truncation and subtraction errors, whereas CS avoids these issues with a sufficiently small step size and can achieve machine precision [55]. FD and CS are relatively easy to implement due to their black-box-like nature. However, these methods may not be practical for high-fidelity applications involving a large number of design variables. (For a detailed comparison of FD and the adjoint method, refer to [56] and [53, Chapter 6] (Fig. 6.43)). Further, their computational cost scales unfavorably with the number of inputs, with CS being more expensive because of the use of imaginary arithmetic. For instance, with respect to differentiating SVD, Ersoy, and Mugan [29] computed the SVD derivatives using the FD approach, which lacks both accuracy and scalability with respect to design variables. This motivates us to choose a method that does not scale with a number of inputs.

AD differentiates a program based on the systematic application of the chain rule. Forward-AD (FAD) does this by applying the chain rule from inputs to the outputs, and reverse-AD (RAD) does so in the reverse order. Thus, the cost is proportional to the number of inputs when FAD is implemented and the number of outputs when RAD is used. Currently, for derivative algorithms that do not scale with inputs, there are only RAD-based methods in the literature to compute the SVD derivative [57–60]. The RAD formula proposed by Giles [57] works for only real values. The RAD form proposed by Townsend [58] works for real values and works with reduced SVD. Santiago et al. [51] took the derivatives of the singular values for the shape optimization for photonic nanostructures by using the RAD formula given for real-valued SVD by Townsend [58]. Wan and Zhang [60] proposed a RAD formula for the full SVD that works with complex-valued inputs. Finally, Seeger et al. [59] proposed a formula equivalent to Townsend's [58]. However, these methods require all of the singular variables during the derivative computation. This increases the computational storage required, which motivates the proposal of a method that does not involve several large matrix–matrix products and computes the derivative more memory-efficiently. Projection-based method was developed by He et al. [61] to partially address this challenge in eigenvector derivative computation.

Beyond explicit analytic methods like AD, implicit analytic methods like direct and adjoint approaches can also be used [53, Chapter 6]. The efficiency of these methods depends on the relative number of inputs and outputs. The direct method is preferable when the number of inputs is smaller than the number of outputs. In contrast, the adjoint method is more efficient when the number

Table 1
Summary of contributions in literature and current manuscript for SVD derivative computation.

Equations	Uses only required singular variables[a]	Singular value derivative	Singular vector derivative	Works for complex-valued inputs	Author (s)
$\overline{\mathbf{A}} = \mathbf{U}\overline{\mathbf{S}}\mathbf{V}^{\intercal}$	✗	✓	✗	✗	Eq. (K.1) Giles [63]
$\overline{\mathbf{A}} = [\mathbf{U}(\mathbf{F}\circ[\mathbf{U}^{\intercal}\overline{\mathbf{U}} - \overline{\mathbf{U}}^{\intercal}\mathbf{U}])\mathbf{S}$ $+(\mathbf{I}_m - \mathbf{U}\mathbf{U}^{\intercal})\overline{\mathbf{U}}\mathbf{S}^{-1}]\mathbf{V}^{\intercal}$ $+\mathbf{U}(\mathbf{I}_k\circ\overline{\mathbf{S}})\mathbf{V}^{\intercal}\ \ \mathbf{U}[\mathbf{S}(\mathbf{F}\circ[\mathbf{V}^{\intercal}\overline{\mathbf{V}} - \overline{\mathbf{V}}^{\intercal}\mathbf{V}])\mathbf{V}^{\intercal}$ $+\mathbf{S}^{-1}\overline{\mathbf{V}}^{\intercal}(\mathbf{I}_n - \mathbf{V}\mathbf{V}^{\intercal})]$	✗	✓	✓	✗	Eq. (K.2) Townsend [58]
$\overline{\mathbf{A}} = \frac{1}{2}(2\mathbf{U}\overline{\mathbf{S}}\mathbf{V}^* + \mathbf{U}(\mathbf{J}+\mathbf{J}^*)\mathbf{S}\mathbf{V}^*$ $+\mathbf{U}\mathbf{S}(\mathbf{K}+\mathbf{K}^*)\mathbf{V}^* + \frac{1}{2}\mathbf{U}\mathbf{S}^{-1}(\mathbf{L}^* - \mathbf{L})\mathbf{V}^*$ $+2(\mathbf{I} - \mathbf{U}\mathbf{U}^*)\overline{\mathbf{U}}\mathbf{S}^{-1}\mathbf{V}^*$ $+2\mathbf{U}\mathbf{S}^{-1}\overline{\mathbf{V}}^*(1 - \mathbf{V}\mathbf{V}^*))$	✗	✓	✓	✓	Eq. (K.4) Wan and Zhang [60]
$\overline{\mathbf{A}} = \mathbf{U}^{\intercal}(\mathbf{G}_2\mathbf{V} + \boldsymbol{\Lambda}^{-1}\overline{\mathbf{V}})$, $\mathbf{G}_2 = \overline{\boldsymbol{\Lambda}} + 2sym(\mathbf{G}_1\circ\mathbf{E})\boldsymbol{\Lambda}$ $-(\boldsymbol{\Lambda}^{-1}\overline{\mathbf{V}}\mathbf{V}^{\intercal}\circ\mathbf{I})$, $\mathbf{G}_1 = \overline{\mathbf{U}}\mathbf{U}^{\intercal} + \boldsymbol{\Lambda}^{-1}\overline{\mathbf{V}}\mathbf{V}^{\intercal}\boldsymbol{\Lambda}$	✗	✗	✓	✗	Eq. (K.5) Seeger et al. [59]
$\mathbf{M}_g^{\intercal}\boldsymbol{\psi}_g = (\partial g/\partial\mathbf{w})^{\intercal}\ \ \mathbf{M}_h^{\intercal}\boldsymbol{\psi}_h = (\partial h/\partial\mathbf{w})^{\intercal}$	✓	✓	✓	✓	Eqs. (16) and (21)
$\mathbf{M}_f^{\intercal}\boldsymbol{\psi}_f = (\partial f/\partial\mathbf{w})^{\intercal}$	✓	✓	✓	✓	Eq. (26)
$\overline{\mathbf{A}}_r = (\mathbf{u}_r\mathbf{v}_r^{\intercal} + \mathbf{u}_i\mathbf{v}_i^{\intercal})\overline{\sigma}\ \ \overline{\mathbf{A}}_i = (-\mathbf{u}_r\mathbf{v}_i^{\intercal} + \mathbf{u}_i\mathbf{v}_r^{\intercal})\overline{\sigma}$	✓	✓	✗	✓	Eq. (30)

[a] Requires only σ, $\mathbf{u}$ and/or $\mathbf{v}$ for the derivative calculations, not all the singular variables inside matrices $\mathbf{U}$, $\boldsymbol{\Sigma}$ and/or $\mathbf{V}$.

of inputs exceeds the number of outputs. Skene and Schmid [52], for instance, used the direct approach to compute the resolvent derivative. However, this scales inefficiently with the number of inputs. Further, there are no adjoint-based methods to compute the SVD derivative.

To address the aforementioned challenges, we develop an adjoint and RAD-based method to compute the derivative efficiently. The contribution of the current manuscript is summarized as follows: (1) We develop two adjoint-based approaches to compute the derivative of the SVD problem for general imaginary (square and rectangular) matrices. In the first approach, we leverage the relationship between the EVP of the Gram matrices of the SVD of the target matrix. In the second approach, we leverage the relationship between the EVP of the target matrix's symmetric embedding and the target matrix's SVD. The proposed adjoint techniques can calculate the derivative to machine precision, are straightforward to implement, do not scale with the number of design variables (inputs), and are suitable for use in gradient-based optimization involving SVD. (2) We develop a RAD-based analytic formula to compute the singular value derivative with respect to a imaginary input matrix, which also works with real-valued matrices in a reduced form. (3) We propose a generic dot product identity as a tool to derive complex-valued RAD derivative formulae using FAD derivative formulae for imaginary differentiable functions.

The manuscript is organized as follows. In Section 2, we present the governing equations for the two proposed adjoint method-based approaches for computing the SVD derivative. Here, the first method relies on forming one of the two Gram-matrices of the main matrix whose SVD derivative is sought. The second method is based on the symmetric embedding of the target matrix, as mentioned. In Section 3, we propose these two adjoint-based approaches and a general RAD-based formula for singular value derivative. In Section 4, we compare the results from our proposed adjoint methods and RAD formula for singular value derivative with FD for two randomly selected imaginary matrices, one square and one rectangular. Next, we show the scalability by testing the singular value derivative on a large dataset sourced from the John Hopkins turbulence database (JHTDB) [62], which was the transition to turbulence dataset of flow over a flat plate. Finally, we present our conclusions in Section 5. We present a summary of the contributions of researchers in differentiating the SVD and our contributions in Table 1.

2. Governing equations

SVD for a general imaginary matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is defined as

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{1}$$

where $\mathbf{U} \in \mathbb{C}^{m\times m}$ is the matrix of left singular vectors with each column being a left singular vector, $\mathbf{V} \in \mathbb{C}^{n\times n}$ is the matrix of right singular vectors with each column being a left singular vector, $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a rectangular matrix of singular values on the main diagonal and zeros elsewhere represented by σ_i on the main diagonal, arranged in descending order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, $\square^*$ denotes the Hermitian transpose of a imaginary matrix, $m, n \in \mathbb{Z}^+$ are the problem size (without loss of generality, we assume $m > n$).

The SVD and EVP are closely related problems. There are two ways to compute the singular value via solving an EVP. The first approach uses the two Gram matrices $\mathbf{B} = \mathbf{A}\mathbf{A}^* \in \mathbb{C}^{m\times m}$ and $\mathbf{C} = \mathbf{A}^*\mathbf{A} \in \mathbb{C}^{n\times n}$ (refer to Schwerdtfeger [42, Page 141] and to Makkonen and Hollanti [43]). Thus, we name it *Gram matrix method* (GMM). The second approach is *Symmetric embedding matrix method* (SEMM) (refer to Golub and Van Loan [35, Chapter 8.6] and Ragnarsson and Van Loan [45]). We elaborate on these two approaches in the following Sections 2.1 and 2.2.

2.1. Gram matrix method (GMM)

It can be shown that the singular vectors and singular values can be obtained from $\mathbf{B}$ and $\mathbf{C}$ (See Golub and Van Loan [35, Chapter 8.6])

$$\begin{aligned}\mathbf{B} &= \mathbf{A}\mathbf{A}^* = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^* = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^*,\\ \mathbf{C} &= \mathbf{A}^*\mathbf{A} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^*\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^* = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^*.\end{aligned} \tag{2}$$

Thus, the left singular vectors, $\mathbf{U}$, are just the eigenvectors obtained from the eigen-decomposition of $\mathbf{B}$ and the right singular vectors, $\mathbf{V}$, are the eigenvectors obtained from the eigen-decomposition of $\mathbf{C}$. The first few singular values are taken from the common eigenvalues between $\mathbf{B}$ and $\mathbf{C}$. The remaining singular values are taken from the remaining eigenvalues depending on whether $\mathbf{A}$ is tall or wide.

To apply adjoint method to compute the derivative, we need to construct the residual form of the EVP. Consider an EVP of a general imaginary square matrix, $\mathbf{D} \in \mathbb{C}^{m\times m}$, we have

$$\mathbf{D}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}, \tag{3}$$

where λ and $\boldsymbol{\phi}$ is one eigen-pair. In general, the eigenvalues and vectors are imaginary. It is assumed that the eigenvalues are distinct and hence have multiplicity of one. To obtain unique imaginary eigenvectors, we consider the following set of governing equations, in conjunction with Eq. (3). The residual form is (see He et al. [44] for more detail)

$$\mathbf{r}(\mathbf{w}) = \begin{bmatrix}\mathbf{D}\boldsymbol{\phi} - \lambda\boldsymbol{\phi}\\ \boldsymbol{\phi}^*\boldsymbol{\phi} - 1\\ \mathrm{Im}(\boldsymbol{\phi}_k)\end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix}\lambda\\ \boldsymbol{\phi}\end{bmatrix}, \tag{4}$$

where we further require that $\mathrm{Re}(\boldsymbol{\phi}_k) > 0$, $k = \mathrm{argmax}_j \|\boldsymbol{\phi}_j\|_2$, and $\boldsymbol{\phi}_k$ is the kth element of the vector $\boldsymbol{\phi}$. This is required because the imaginary eigenvector can always be scaled and rotated in the imaginary plane. If we apply stretching and rotation in this plane, we obtain:

$$\mathbf{D}(\alpha\boldsymbol{\phi}e^{i\theta}) = \lambda(\alpha\boldsymbol{\phi}e^{i\theta}), \tag{5}$$

where $\alpha \in \mathbb{R}$ (scaling factor), $\theta \in \mathbb{R}$ (rotation angle).

The last two equations from Eq. (4) therefore constrain the norm of the eigenvector to be one, and constrains the angle of the eigenvector.

Rewriting Eq. (4) by splitting into their respective real and imaginary parts, we have

$$\mathbf{r}(\mathbf{w}) = \begin{bmatrix} r_{\text{main,r}}\\ r_{\text{main,i}}\\ r_m\\ r_p\end{bmatrix} = \begin{bmatrix}\mathbf{D}_r\boldsymbol{\phi}_r - \mathbf{D}_i\boldsymbol{\phi}_i - \lambda_r\boldsymbol{\phi}_r + \lambda_i\boldsymbol{\phi}_i\\ \mathbf{D}_i\boldsymbol{\phi}_r + \mathbf{D}_r\boldsymbol{\phi}_i - \lambda_i\boldsymbol{\phi}_r - \lambda_r\boldsymbol{\phi}_i\\ \boldsymbol{\phi}_r^\intercal\boldsymbol{\phi}_r + \boldsymbol{\phi}_i^\intercal\boldsymbol{\phi}_i - 1\\ e_k^\intercal\boldsymbol{\phi}_i\end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix}\boldsymbol{\phi}_r\\ \boldsymbol{\phi}_i\\ \lambda_r\\ \lambda_i\end{bmatrix}, \tag{6}$$

where $\mathbf{e}_k$ is a vector with all zero elements except that k^{th} element equal to one.

For SVD, we can leverage Eq. (2) to convert it to an EVP. In Eq. (6), we can set $\mathbf{D} = \mathbf{B}$, then we have a governing equation for the left singular vector and singular value. In this case, the left singular value becomes the square root of λ and the corresponding left singular vector is simply equal to $\boldsymbol{\phi}$ in Eq. (3). We call this the *Left Gram Matrix Method* (LGMM) since it is the Gram matrix associated with the left singular values.

Alternatively, if we set $\mathbf{D} = \mathbf{C}$, we have a governing equation for the right singular vector and singular value. Again, in this case, the right singular value becomes the square root of λ and the corresponding right singular vector is simply equal to $\boldsymbol{\phi}$ in Eq. (3). We call this the *Right Gram Matrix Method* (RGMM) since it is the Gram matrix associated with the right singular values.

2.2. Symmetric embedding matrix method

This second approach leverages the following relationship

$$\begin{bmatrix}\mathbf{O} & \mathbf{A}\\ \mathbf{A}^* & \mathbf{O}\end{bmatrix}\begin{bmatrix}\mathbf{u}_i\\ \mathbf{v}_i\end{bmatrix} = \pm\sigma_i\begin{bmatrix}\mathbf{u}_i\\ \mathbf{v}_i\end{bmatrix}, \tag{7}$$

where σ_i, $\mathbf{u}_i$ and $\mathbf{v}_i$ is one group of singular value and singular vectors (refer to Golub and Van Loan [35, Chapter 8.6] and Ragnarsson and Van Loan [45]). Thus, by applying these identities Eqs. (2) and (7), we can obtain the nonlinear governing equations for SVD with additional normalization condition constraints.

In the previous Section 2.1, we develop a formulation that relies on the evaluation of $\mathbf{B}$ or $\mathbf{C}$ to convert the problem to an EVP where the left or right singular vectors for the corresponding singular value are computed. The main drawback of that method is that the evaluation of matrix product, $\mathbf{B} = \mathbf{A}\mathbf{A}^*$ or $\mathbf{C} = \mathbf{A}^*\mathbf{A}$, can be expensive when the coefficient matrix is dense. This motivates us to develop an alternative formulation that is matrix-product free.

We developed the following equation. By solving this equation, we can get the left, and right singular vectors, and singular value in one shot. The equation is

$$\begin{aligned} \mathbf{A}\mathbf{v} &= \sigma\mathbf{u}, \\ \mathbf{A}^*\mathbf{u} &= \sigma\mathbf{v}, \\ \mathbf{u}^*\mathbf{u} &= 1, \\ \mathrm{Im}(\mathbf{u}_k) &= 0. \end{aligned} \tag{8}$$

Notice that by constraining the left singular vector $\mathbf{u}$, we end up constraining the right singular vector $\mathbf{v}$.

Starting from Eq. (8), we can split it into real and imaginary parts and write the residual form as

$$\mathbf{r}(\mathbf{w}) = \begin{bmatrix} \mathbf{A}_r\mathbf{v}_r - \mathbf{A}_i\mathbf{v}_i - \sigma_r\mathbf{u}_r + \sigma_i\mathbf{u}_i \\ \mathbf{A}_r\mathbf{v}_i + \mathbf{A}_i\mathbf{v}_r - \sigma_r\mathbf{u}_i - \sigma_i\mathbf{u}_r \\ \mathbf{A}_r^\intercal\mathbf{u}_r + \mathbf{A}_i^\intercal\mathbf{u}_i - \sigma_r\mathbf{v}_r + \sigma_i\mathbf{v}_i \\ \mathbf{A}_r^\intercal\mathbf{u}_i - \mathbf{A}_i^\intercal\mathbf{u}_r - \sigma_r\mathbf{v}_i - \sigma_i\mathbf{v}_r \\ \mathbf{u}_r^\intercal\mathbf{u}_r + \mathbf{u}_i^\intercal\mathbf{u}_i - 1 \\ \mathbf{e}_k^\intercal\mathbf{u}_i \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{u}_r \\ \mathbf{u}_i \\ \mathbf{v}_r \\ \mathbf{v}_i \\ \sigma_r \\ \sigma_i \end{bmatrix}, \tag{9}$$

where $\mathbf{w}$ is the state vector. It is well-known that the singular value, σ, is real. However, here we relax it to be a imaginary number to have equal numbers of equations and unknowns.

Now, we prove that the solution of Eq. (8) is indeed a SVD pair. Pre-multiplying the first equation of Eq. (8) by $\mathbf{u}^*$, we have

$$\mathbf{u}^*\mathbf{A}\mathbf{v} = \sigma\mathbf{u}^*\mathbf{u} = \sigma. \tag{10}$$

Then, we pre-multiply the second equation of Eq. (8) by $\mathbf{v}^*$, we have

$$\mathbf{v}^*\mathbf{A}^*\mathbf{u} = \sigma\mathbf{v}^*\mathbf{v} = \sigma\|\mathbf{v}\|_2. \tag{11}$$

Then, realizing that Eqs. (10) and (11) are conjugate transpose, we have

$$\sigma^* = \sigma\|\mathbf{v}\|_2 \Rightarrow \sigma_r = \|\mathbf{v}\|_2\sigma_r, -\sigma_i = \|\mathbf{v}\|_2\sigma_i, \tag{12}$$

we have as long as $\sigma = 0$, or $\|\mathbf{v}\|_2 = 1$ and $\sigma_i = 0$. Thus, we have implicitly enforced that σ is real.

2.3. Discussion

The difference between the two formulations GMM and SEMM is that using Eq. (2) method, we can formulate a smaller dimension problem (e.g. if we have $m > n$, we can form $\mathbf{C}$ which is $n \times n$). However, the disadvantage of this method is that it only computes one singular vector (left or right) and singular value, and the other singular vector (right or left) needs to be recovered. Another disadvantage is that Eq. (2) requires computing matrix-product. If the matrix is dense, it can be prohibitively expensive to compute this product.

On the other hand, the conversion using Eq. (7) has the advantage that it does not require the matrix product needed in the previous matrix-product formulation. In addition, the equation includes the singular value and the corresponding left and right singular vectors. Thus, we do not need to conduct additional computation to search the other singular vector as required by Eq. (2) method. The disadvantage of this method is that the problem size ($m + n$) is larger than Eq. (2) method. We also ensure that there are no repeated singular values else the singular vector space becomes degenerate [35,50].

3. Derivative computation

In this section, we propose two general methods to compute the derivative of any function of the singular variables with respect to a general imaginary matrix, $\mathbf{A}$. We also propose an RAD formula for just the singular value derivative. The section is organized as follows: In Section 3.1, a brief introduction to the adjoint method is given. Section 3.2 then gives the adjoint method applied to the governing equations resulting from GMM. Similarly in Section 3.3, the adjoint method is applied to the governing equations resulting from the SEMM. Finally, in Section 3.4, we propose the RAD formula for the singular value derivative for both imaginary and real-valued cases.

3.1. Adjoint method

For arbitrary function of interest, $f(\mathbf{w}, \mathbf{x})$, we can use the adjoint method to compute its total derivative with respect to $\mathbf{x}$. The total derivative is computed using the following formula [53]

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \boldsymbol{\psi}^\intercal \frac{\partial \mathbf{r}}{\partial \mathbf{x}}. \tag{13}$$

The adjoint vector, $\boldsymbol{\psi}$, can be found by solving the system of equations resulting from the following adjoint equation

$$\frac{\partial \mathbf{r}}{\partial \mathbf{w}}^{\mathsf{T}} \boldsymbol{\psi} = \frac{\partial f}{\partial \mathbf{w}}^{\mathsf{T}}, \tag{14}$$

where $\mathbf{r} = \mathbf{r}\,(\mathbf{w})$ is the residual vector for the governing equation (Eq. (4)), $\mathbf{w}$ being the state variables vector. We direct the readers to He et al. [44] for the complete adjoint method to solving the EVP derivative. For the summary of the equations and algorithm, see Appendix A.

As a note to the readers, in typical optimization problems, the objective function is $f(\mathbf{w},\mathbf{x})$ and that representation is more general. In the adjoint approach, we form residual and that is $\mathbf{r}(\mathbf{w},\mathbf{x})$. The design variables $\mathbf{x}$ affect the matrix $\mathbf{A}$ whose SVD is sought and this in turn affects the objective function f. Since we calculate $\mathrm{d}f/\mathrm{d}\mathbf{A}$ and not $\mathrm{d}f/\mathrm{d}\mathbf{x}$, this is a function $f(\mathbf{w})$ alone. Therefore, the corresponding residual vector is also a function of $\mathbf{w}$ as $\mathbf{r}(\mathbf{w})$.

For the SVD problem, our objective function now becomes:

$$f = f(\mathbf{u},\mathbf{v},\sigma,\mathbf{A}), \tag{15}$$

where f can be any function of the singular variables: $\mathbf{u}$ which is the left singular vector, $\mathbf{v}$ the right singular vector and these vectors correspond to a chosen singular value (typically the dominant one) σ. $\mathbf{A}$ is also placed in the objective function for the general case if there are any direct dependencies on it, as will be seen in forthcoming sections.

3.2. *Adjoint equation of GMM*

The computation of the derivative of a general function, $f(\mathbf{u},\mathbf{v},\sigma,\mathbf{A})$, can be decomposed into two steps. Before we begin with the computation of the derivative of $f(\mathbf{u},\mathbf{v},\sigma,\mathbf{A})$ with respect to $\mathbf{A}$, consider first the derivatives of $g(\mathbf{u},\sigma,\mathbf{A})$ and $h(\mathbf{v},\sigma,\mathbf{A})$. If the function is $g(\mathbf{u},\sigma,\mathbf{A})$, it is related to the matrix $\mathbf{B}$. If the function is $h(\mathbf{v},\sigma,\mathbf{A})$, it is related to the matrix $\mathbf{C} = \mathbf{A}^*\mathbf{A}$ (Refer Section 2.1). We split this function f into its real and imaginary components as g_r and g_i for the function g and h_r and h_i for the function h and therefore we must solve the adjoint for a total of two times for each of g and h, once for real part and once for imaginary part of each of g and h. This is done to save computational effort as will be seen in the forthcoming section. We present this approach as follows.

3.2.1. *Adjoint equation of LGMM*

We found the adjoint equation for the function $g = g(\mathbf{u},\sigma,\mathbf{A}) \in \mathbb{R}$ to be

$$\mathbf{M}_g^{\mathsf{T}} \boldsymbol{\psi}_g = \frac{\partial g}{\partial \mathbf{w}}^{\mathsf{T}}, \tag{16}$$

where $\mathbf{M}_g$ is

$$\mathbf{M}_g = \begin{bmatrix} \mathbf{B}_r - \lambda_r \mathbf{I} & -\mathbf{B}_i + \lambda_i \mathbf{I} & -\mathbf{u}_r & \mathbf{u}_i \\ \mathbf{B}_i - \lambda_i \mathbf{I} & \mathbf{B}_r + \lambda_r \mathbf{I} & -\mathbf{u}_i & -\mathbf{u}_r \\ 2\mathbf{u}_r^{\mathsf{T}} & 2\mathbf{u}_i^{\mathsf{T}} & 0 & 0 \\ 0 & \mathbf{e}_k^{\mathsf{T}} & 0 & 0 \end{bmatrix}, \tag{17}$$

and $\boldsymbol{\psi}_g$ is

$$\boldsymbol{\psi}_g = \begin{bmatrix} \boldsymbol{\psi}_{g,\mathrm{r}} \\ \boldsymbol{\psi}_{g,\mathrm{i}} \\ \boldsymbol{\psi}_{\mathrm{m},\ g} \\ \boldsymbol{\psi}_{\mathrm{p},\ g} \end{bmatrix}, \tag{18}$$

and once the solution to the adjoint vector $\boldsymbol{\psi}$ from either approach is obtained, we can proceed to compute the total derivatives using Eq. (13).

We assumed that function g outputs a real number, i.e., $g \in \mathbb{R}$. If otherwise, function g returns a imaginary number, i.e., $g \in \mathbb{C}$, we can break it into its real and imaginary parts and apply Eq. (16) twice. We now have two adjoint vectors, namely: $\boldsymbol{\psi}_{g_r}$ and $\boldsymbol{\psi}_{g_i}$ for the function g, each with its real and imaginary parts. The derivative of the residual vector $\mathbf{r}$ for the function g with respect to the matrix $\mathbf{B}$ is then

$$\begin{aligned} \frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_r} &= \boldsymbol{\psi}_{g_r,r}\mathbf{u}_r^{\mathsf{T}} + \boldsymbol{\psi}_{g_r,i}\mathbf{u}_i^{\mathsf{T}}, \\ \frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_r} &= -\boldsymbol{\psi}_{g_r,r}\mathbf{u}_i^{\mathsf{T}} + \boldsymbol{\psi}_{g_r,i}\mathbf{u}_r^{\mathsf{T}}, \\ \frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_i} &= \boldsymbol{\psi}_{g_i,r}\mathbf{u}_r^{\mathsf{T}} + \boldsymbol{\psi}_{g_i,i}\mathbf{u}_i^{\mathsf{T}}, \\ \frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_i} &= -\boldsymbol{\psi}_{g_i,r}\mathbf{u}_i^{\mathsf{T}} + \boldsymbol{\psi}_{g_i,i}\mathbf{u}_r^{\mathsf{T}}, \end{aligned} \tag{19}$$

A detailed derivation for this formula can be found in the paper by He et al. [44]. To proceed to the derivative with respect to the matrix $\mathbf{A}$, we apply chain rule via the reverse automatic differentiation formula (See Appendix B for derivation). Thus, to compute the derivatives of the function $f = g$ with respect to matrix $\mathbf{A}$, we have

$$\begin{aligned}
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_r} &= -\left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)\mathbf{A}_r - \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)^{\mathsf{T}}\mathbf{A}_r - \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)^{\mathsf{T}}\mathbf{A}_i \\
&\quad + \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)\mathbf{A}_i + \frac{\partial f_r}{\partial \mathbf{A}_r}, \\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_r} &= -\left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)\mathbf{A}_r - \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)^{\mathsf{T}}\mathbf{A}_r - \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)^{\mathsf{T}}\mathbf{A}_i \\
&\quad + \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)\mathbf{A}_i + \frac{\partial f_i}{\partial \mathbf{A}_r}, \\
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_i} &= -\left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)\mathbf{A}_i - \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)^{\mathsf{T}}\mathbf{A}_i - \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)^{\mathsf{T}}\mathbf{A}_r \\
&\quad + \left(\frac{\partial \mathbf{r}_{g_r}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_r}\right)\mathbf{A}_r + \frac{\partial f_r}{\partial \mathbf{A}_i}, \\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_i} &= -\left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)\mathbf{A}_i - \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_r}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)^{\mathsf{T}}\mathbf{A}_i - \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)^{\mathsf{T}}\mathbf{A}_r \\
&\quad + \left(\frac{\partial \mathbf{r}_{g_i}}{\partial \mathbf{B}_i}^{\mathsf{T}} \boldsymbol{\psi}_{g_i}\right)\mathbf{A}_r + \frac{\partial f_i}{\partial \mathbf{A}_i},
\end{aligned} \tag{20}$$

where the last terms in each equation in partial derivative are from the contribution of $\mathbf{A}$ leading to direct dependencies on it.

As a note to the readers, the derivative expression in Eq. (20) above and many such forthcoming equations involve matrices such as $(\partial \mathbf{r}_{g_r}/\partial \mathbf{B}_r)^{\mathsf{T}}\boldsymbol{\psi}_{g_r}$, which includes complicated tensor–vector products and these can become cumbersome to track and compute. To simplify this, we assume the derivative is calculated after flattening the matrices into vectors. Once the computation is complete, the resulting vectors are reshaped back to their original matrix dimensions. The vector flattening and tensor–vector product is elaborated in detail in Appendix C.

Next, consider the objective function $f = f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A})$. Using the governing equation Eq. (8), $\mathbf{u}$ can be expressed as $\mathbf{A}\mathbf{v}/\sigma$, then we have that $f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A}) = f(\mathbf{u}, \mathbf{v}(\mathbf{u}, \sigma, \mathbf{A}), \sigma, \mathbf{A})$. If that is done, then the objective function $f = g(\mathbf{u}, \sigma, \mathbf{A})$ as shown above. The same formula in Eq. (20) can be used to compute the derivative of f with respect to $\mathbf{A}$. This saves the computational effort if $\mathbf{A}$ is tall.

Since the function f can be any function of $f = f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A})$, there is no general formula that can be devised for that direct contribution of $\mathbf{A}$. However, if the only direct contribution of $\mathbf{A}$ is from changing of $\mathbf{v}$ to $\mathbf{A}^*\mathbf{u}/\sigma$, then the imaginary partial derivative components of $\partial f/\partial \mathbf{A}$ can be written in RAD form shown in Appendix D.

It is a fact that the last terms of partial derivatives of $\partial f/\partial \mathbf{A}$ in Eq. (20) can also be obtained by perturbing only $\mathbf{A}$ in the objective function if present, either using FD or automatic differentiation tools such as JAX [64]. These partial derivative terms in Eq. (20) reduce to zero if f is strictly a function of $\mathbf{u}$ and σ.

3.2.2. *Adjoint equation of RGMM*

We found the adjoint equation for the function $h = f(\mathbf{v}, \sigma, \mathbf{A})$ to be

$$\mathbf{M}_h^{\mathsf{T}}\boldsymbol{\psi}_h = \frac{\partial h}{\partial \mathbf{w}}^{\mathsf{T}}, \tag{21}$$

where $\mathbf{M}_h$ is

$$\mathbf{M}_h = \begin{bmatrix} \mathbf{C}_r - \lambda_r\mathbf{I} & -\mathbf{C}_i + \lambda_i\mathbf{I} & -\mathbf{v}_r & \mathbf{v}_i \\ \mathbf{C}_i - \lambda_i\mathbf{I} & \mathbf{C}_r + \lambda_r\mathbf{I} & -\mathbf{v}_i & -\mathbf{v}_r \\ 2\mathbf{v}_r^{\mathsf{T}} & 2\mathbf{v}_i^{\mathsf{T}} & 0 & 0 \\ 0 & \mathbf{e}_k^{\mathsf{T}} & 0 & 0 \end{bmatrix}, \tag{22}$$

and $\boldsymbol{\psi}_h$ is

$$\boldsymbol{\psi}_h = \begin{bmatrix} \boldsymbol{\psi}_{h,\mathrm{r}} \\ \boldsymbol{\psi}_{h,\mathrm{i}} \\ \boldsymbol{\psi}_{\mathrm{m},\ h} \\ \boldsymbol{\psi}_{\mathrm{p},\ h} \end{bmatrix}, \tag{23}$$

where h can be the real (h_r) or imaginary part (h_i) of h if it is imaginary.

Note that λ which is the eigenvalue is the same in both Eqs. (16) and (21) because both the matrices $\mathbf{B}$ and $\mathbf{C}$ will have some common eigenvalues and λ is one of them. $(\sigma = \sqrt{\lambda})$. The system of equations can be solved by a linear solver such as the solver `numpy.linalg.solve` in `Python` [65]. Details on this are shed in He et al. [44].

We now have two adjoint vectors, namely: $\boldsymbol{\psi}_{h_r}$ and $\boldsymbol{\psi}_{h_i}$ for the function h, each with its real and imaginary parts. The derivative of the residual vector $\mathbf{r}$ for the function h with respect to the matrix $\mathbf{C}$ is

$$\begin{aligned}
\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_r}^{\intercal} \boldsymbol{\psi}_{h_r} &= \boldsymbol{\psi}_{h_r,r}\mathbf{v}_r^{\intercal} + \boldsymbol{\psi}_{h_r,i}\mathbf{v}_i^{\intercal},\\
\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_i}^{\intercal} \boldsymbol{\psi}_{h_r} &= -\boldsymbol{\psi}_{h_r,r}\mathbf{v}_i^{\intercal} + \boldsymbol{\psi}_{h_r,i}\mathbf{v}_r^{\intercal},\\
\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_r}^{\intercal} \boldsymbol{\psi}_{h_i} &= \boldsymbol{\psi}_{h_i,r}\mathbf{v}_r^{\intercal} + \boldsymbol{\psi}_{h_i,i}\mathbf{v}_i^{\intercal},\\
\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_i}^{\intercal} \boldsymbol{\psi}_{h_i} &= -\boldsymbol{\psi}_{h_i,r}\mathbf{v}_i^{\intercal} + \boldsymbol{\psi}_{h_i,i}\mathbf{v}_r^{\intercal}.
\end{aligned} \tag{24}$$

To proceed to the derivative with respect to the matrix $\mathbf{A}$, we apply chain rule via the reverse automatic differentiation formula (See Appendix B for derivation). Thus, to compute the derivatives of the function $f = h$ with respect to matrix $\mathbf{A}$, we have

$$\begin{aligned}
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_r} &= -\mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_r}\right)^{\intercal} - \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_r}\right) - \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_r}\right)^{\intercal}\\
&\quad + \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_r}\right) + \frac{\partial f_r}{\partial \mathbf{A}_r},\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_r} &= -\mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_i}\right)^{\intercal} - \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_i}\right) - \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_i}\right)^{\intercal}\\
&\quad + \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_i}\right) + \frac{\partial f_i}{\partial \mathbf{A}_r},\\
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_i} &= -\mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_r}\right)^{\intercal} - \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_r}\right) - \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_r}\right)^{\intercal}\\
&\quad + \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_r}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_r}\right) + \frac{\partial f_r}{\partial \mathbf{A}_i},\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_i} &= -\mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_i}\right)^{\intercal} - \mathbf{A}_i\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_r}^{\intercal}\boldsymbol{\psi}_{h_i}\right) - \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_i}\right)^{\intercal}\\
&\quad + \mathbf{A}_r\left(\frac{\partial \mathbf{r}_{h_i}}{\partial \mathbf{C}_i}^{\intercal}\boldsymbol{\psi}_{h_i}\right) + \frac{\partial f_i}{\partial \mathbf{A}_i},
\end{aligned} \tag{25}$$

where the last terms in each equation in partial derivative are from the contribution of $\mathbf{A}$ leading to direct dependencies on it.

Consider the objective function $f = f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A})$. Using the governing equation Eq. (8), $\mathbf{u}$ can be expressed as $\mathbf{A}\mathbf{v}/\sigma$, then we have that $f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A}) = f(\mathbf{u}(\mathbf{v}, \sigma, \mathbf{A}), \mathbf{v}, \sigma, \mathbf{A})$. If that is done, then the objective function $f = h(\mathbf{v}, \sigma, \mathbf{A})$ as shown above. The same formula in Eq. (25) can be used to compute the derivative of f with respect to $\mathbf{A}$. This saves the computational effort if $\mathbf{A}$ is wide.

Since the function f can be any function of $f = f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A})$, there is no general formula that can be devised for that direct contribution of $\mathbf{A}$. However, if the only direct contribution of $\mathbf{A}$ is from changing of $\mathbf{u}$ to $\mathbf{A}\mathbf{v}/\sigma$, then the imaginary partial derivative components of $\partial f/\partial \mathbf{A}$ can be written in RAD form shown in Appendix D.

It is a fact that the last terms of partial derivatives of $\partial f/\partial \mathbf{A}$ in Eq. (20) can also be obtained by perturbing only $\mathbf{A}$ in the objective function if present, either using FD or automatic differentiation tools such as JAX [64]. These partial derivative terms in Eq. (20) reduce to zero if f is strictly a function of $\mathbf{u}$ and σ.

To summarize keeping the minimization of computational effort in mind, we employ Eqs. (16)–(18) and (20) if the function is expressed in terms of the singular variables $\mathbf{u}$ and σ. If it is expressed in terms of $\mathbf{v}$ and σ, then we use Eqs. (21)–(23) and (25).

3.3. Adjoint equation of SEMM

In this section, we directly apply the adjoint method to the SVD governing equations as listed in the preceding section Section 2. We found the adjoint equation for the function, f, to be

$$\mathbf{M}_f^{\intercal}\boldsymbol{\psi}_f = \frac{\partial f}{\partial \mathbf{w}}^{\intercal}, \tag{26}$$

where $\mathbf{M}_f$ and $\boldsymbol{\psi}_f$ are

$$\mathbf{M}_f = \begin{bmatrix} -\sigma_r\mathbf{I} & \sigma_i\mathbf{I} & \mathbf{A}_r & -\mathbf{A}_i & -\mathbf{u}_r & \mathbf{u}_i \\ -\sigma_i\mathbf{I} & -\sigma_r\mathbf{I} & \mathbf{A}_i & \mathbf{A}_r & -\mathbf{u}_i & -\mathbf{u}_r \\ \mathbf{A}_r^{\intercal} & \mathbf{A}_i^{\intercal} & -\sigma_r\mathbf{I} & \sigma_i\mathbf{I} & -\mathbf{v}_r & \mathbf{v}_i \\ -\mathbf{A}_i^{\intercal} & \mathbf{A}_r^{\intercal} & -\sigma_i\mathbf{I} & -\sigma_r\mathbf{I} & -\mathbf{v}_i & -\mathbf{v}_r \\ 2\mathbf{u}_r^{\intercal} & 2\mathbf{u}_i^{\intercal} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{e}_k^{\intercal} & \mathbf{0} & \mathbf{0} & 0 & 0 \end{bmatrix}, \quad \boldsymbol{\psi}_f = \begin{bmatrix} \boldsymbol{\psi}_{v_r} \\ \boldsymbol{\psi}_{v_i} \\ \boldsymbol{\psi}_{u_r} \\ \boldsymbol{\psi}_{u_i} \\ \boldsymbol{\psi}_m \\ \boldsymbol{\psi}_p \end{bmatrix}. \tag{27}$$

Here, m and p represent the magnitude and phase respectively. The derivatives of f_r and f_i with respect to $\mathbf{A}_r$ and $\mathbf{A}_i$ are then

$$\begin{aligned}\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^{\intercal}\boldsymbol{\psi}_r + \frac{\partial f_r}{\partial \mathbf{A}_r}\ ,\\ \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^{\intercal}\boldsymbol{\psi}_r + \frac{\partial f_r}{\partial \mathbf{A}_i}\ ,\\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^{\intercal}\boldsymbol{\psi}_i + \frac{\partial f_i}{\partial \mathbf{A}_r}\ ,\\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^{\intercal}\boldsymbol{\psi}_i + \frac{\partial f_i}{\partial \mathbf{A}_i}\ .\end{aligned} \tag{28}$$

Here, the partial derivatives $\partial f_m/\partial \mathbf{A}_n$ in the RHS of the equations above are for any direct dependencies of f on $\mathbf{A}$. The $(\partial r/\partial \mathbf{A})^{\intercal}\boldsymbol{\psi}$ terms in the RHS of the equations above are (derivation in Appendix E)

$$\begin{aligned}\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^{\intercal}\boldsymbol{\psi} &= \boldsymbol{\psi}_{v_r}\mathbf{v}_r^{\intercal} + \boldsymbol{\psi}_{v_i}\mathbf{v}_i^{\intercal} + \mathbf{u}_r\boldsymbol{\psi}_{u_r}^{\intercal} + \mathbf{u}_i\boldsymbol{\psi}_{u_i}^{\intercal}\ ,\\ \frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^{\intercal}\boldsymbol{\psi} &= -\boldsymbol{\psi}_{v_r}\mathbf{v}_i^{\intercal} + \boldsymbol{\psi}_{v_i}\mathbf{v}_r^{\intercal} + \mathbf{u}_i\boldsymbol{\psi}_{u_r}^{\intercal} - \mathbf{u}_r\boldsymbol{\psi}_{u_i}^{\intercal}\ ,\end{aligned} \tag{29}$$

where $\boldsymbol{\psi} = \boldsymbol{\psi}_r$ from Eq. (28) if $f = f_r$ in Eq. (27). Similarly, $\boldsymbol{\psi} = \boldsymbol{\psi}_i$ from Eq. (28) if $f = f_i$ in Eq. (27).

3.4. RAD method for singular value derivative

The technique described so far can compute derivatives of the singular variables at once, in any combination. If the derivative of just one of the singular values is sought, we propose the following RAD based formulae: (Derivation in Appendix F).

$$\begin{aligned}\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}_r} &= \mathbf{u}_r\mathbf{v}_r^{\intercal} + \mathbf{u}_i\mathbf{v}_i^{\intercal},\\ \frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}_i} &= -\mathbf{u}_r\mathbf{v}_i^{\intercal} + \mathbf{u}_i\mathbf{v}_r^{\intercal}.\end{aligned} \tag{30}$$

For the purely real case if $\mathbf{A} = \mathbf{A}_r$, it is

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}} = \mathbf{u}\mathbf{v}^{\intercal}, \tag{31}$$

where $\mathbf{u} \in \mathbb{R}$, $\mathbf{v} \in \mathbb{R}$ and $\mathbf{A} \in \mathbb{R}$. Further, the formula in Eq. (30) reduces to Eq. (31) if the imaginary parts are set to zero.

3.5. Summary

In this section we present a summary of the proposed methods in this manuscript. LGMM, RGMM and SEMM algorithms are shown in Algorithm 1, Algorithm 2 and Algorithm 3 respectively.

Algorithm 1 LGMM algorithm

```
1: Perform SVD on $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$ and obtain the desired singular variables $(\mathbf{u}, \sigma, \mathbf{v})$
2: Construct desired objective function $f$ from the singular variables
3: Assemble and solve the adjoint equation (Equation (16)) to get the adjoint vector $\boldsymbol{\psi}_{g_r}$ from $f_r$ and $\boldsymbol{\psi}_{g_i}$ from $f_i$
4: Compute the derivatives $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$, $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$, $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$ using Equation (20)
```

Algorithm 2 RGMM algorithm

```
1: Perform SVD on $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$ and obtain the desired singular variables $(\mathbf{u}, \sigma, \mathbf{v})$
2: Construct desired objective function $f$ from the singular variables
3: Assemble and solve the adjoint equation (Equation (21)) to get the adjoint vector $\boldsymbol{\psi}_{h_r}$ from $f_r$ and $\boldsymbol{\psi}_{h_i}$ from $f_i$
4: Compute the derivatives $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$, $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$, $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$ using Equation (25)
```

Algorithm 3 SEMM algorithm

```
1: Perform SVD on $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$ and obtain the desired singular variables $(\mathbf{u}, \sigma, \mathbf{v})$
2: Construct desired objective function $f$ from the singular variables
3: Assemble and solve the adjoint equation (Equation (26)) to get the adjoint vector $\boldsymbol{\psi}_{f_r}$ from $f_r$ and $\boldsymbol{\psi}_{f_i}$ from $f_i$
4: Compute the derivatives $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$, $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$, $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$ using Equation (28)
```

The RAD equation for only the singular value derivative is given in Eq. (F.7) for the real part of f and in Eq. (F.9) for the imaginary part. These equations must be used if the objective function f is some function of σ alone. If it is the singular value σ itself, then a simpler form is found in Eq. (30) for complex-valued problems and is reduced to Eq. (31) for real-valued problems. This finishes our summary of the adjoint and RAD methods proposed in our paper.

4. Numerical results

In this section, we verify the proposed adjoint-based method via the GMM and SEMM using two complex-valued matrices, one square matrix and one tall rectangular matrix by comparing the values of the obtained derivatives with the values obtained from FD. The tall rectangular matrix means that it has number of rows greater than the number of columns here [36]. The square matrix is taken and the derivative results from adjoint method for LGMM, RGMM, SEMM and the singular value derivative RAD formula are compared with the results from the FD computation in Section 4.1 for a randomly selected objective function. This is repeated for the rectangular matrix in Section 4.2.

Finally, to show scalability, we compute the singular value derivatives of the singular values from the proper orthogonal decomposition (POD) through SVD of a large dataset of transition to turbulence of flow over a flat plate in Section 4.4. The data was sourced from the JHTDB and the POD was performed by the method of snapshots.

4.1. Square matrix derivative computation

The randomly selected square matrix is

$$\mathbf{A} = \mathbf{A}_r + i\mathbf{A}_i = \begin{bmatrix} -1.01 & 0.86 & -31.42 \\ 3.98 & 0.53 & -7.04 \\ 3.3 & 8.26 & -3.89 \end{bmatrix} + i \begin{bmatrix} 0.6 & 0.79 & 5.47 \\ 7.21 & 1.9 & 0.58 \\ 3.42 & 8.97 & 0.3 \end{bmatrix}. \tag{32}$$

It was ensured that no repeated eigenvalues were found in the eigen-decompositions of $\mathbf{B} = \mathbf{A}\mathbf{A}^*$ and $\mathbf{C} = \mathbf{A}^*\mathbf{A}$. The dominant singular value and its respective left and right singular vectors are

$$\begin{aligned} \sigma_1 &= 33.16357940928816, \\ \mathbf{u}_r + i\mathbf{u}_i &= \begin{bmatrix} 0.9572042 \\ 0.23641926 \\ 0.16616206 \end{bmatrix} + i \begin{bmatrix} 0 \\ 0.01549572 \\ 0.00401452 \end{bmatrix}, \\ \mathbf{v}_r + i\mathbf{v}_i &= \begin{bmatrix} -0.00513976 \\ -0.05740808 \\ 0.99047735 \end{bmatrix} + i \begin{bmatrix} 0.08569246 \\ 0.09104565 \\ 0 \end{bmatrix}. \end{aligned} \tag{33}$$

The objective function was chosen as

$$f = \mathbf{c}_{\mathbf{u}}^{\mathrm{T}}\mathbf{u} + \mathbf{c}_{\mathbf{v}}^{\mathrm{T}}\mathbf{v} + \mathrm{c}_\sigma \sigma + \mathrm{c}_{\mathbf{A}}\mathrm{Tr}(\mathbf{A}), \tag{34}$$

where $\mathrm{Tr}(\cdot)$ is the trace operator and the constants were

$$\begin{aligned} \mathbf{c}_u = \mathbf{c}_v &= \begin{bmatrix} 0.16 + 0.78i \\ 0.53 + 0.11i \\ 0.11 + 0.77i \end{bmatrix}, \\ \mathrm{c}_\sigma = \mathrm{c}_{\mathbf{A}} &= 1, \end{aligned} \tag{35}$$

where the constants were randomly selected.

In Section 3.2, we showed two ways to compute the derivative. One is to express $\mathbf{u}$ in terms of $\mathbf{v}, \sigma, \mathbf{A}$ which is the LGMM or to express $\mathbf{v}$ in terms of $\mathbf{u}, \sigma, \mathbf{A}$ which is the RGMM. If $\mathbf{v}$ is expressed in terms of $\mathbf{u}, \sigma, \mathbf{A}$ following the governing equation in Eq. (8), the objective function in Eq. (34) becomes

$$f = \mathbf{c}_u^{\mathrm{T}}\mathbf{u} + \mathbf{c}_v^{\mathrm{T}}\frac{\mathbf{A}^*\mathbf{u}}{\sigma} + \mathrm{c}_\sigma \sigma + \mathrm{c}_{\mathbf{A}}\mathrm{Tr}(\mathbf{A}). \tag{36}$$

Similarly if $\mathbf{u}$ is expressed in terms of $\mathbf{v}, \sigma, \mathbf{A}$, the objective function in Eq. (34) becomes

$$f = \mathbf{c}_u^{\mathrm{T}}\frac{\mathbf{A}\mathbf{v}}{\sigma} + \mathbf{c}_v^{\mathrm{T}}\mathbf{v} + \mathrm{c}_\sigma \sigma + c_{\mathbf{A}}\mathrm{Tr}(\mathbf{A}). \tag{37}$$

Thus, we have reduced the function from $f(\mathbf{u}, \mathbf{v}, \sigma, \mathbf{A})$ to either $f(\mathbf{u}, \sigma, \mathbf{A})$ for LGMM or $f(\mathbf{v}, \sigma, \mathbf{A})$ for RGMM derivative computation.

For the LGMM, we apply Eqs. (16)–(18) and (20) to compute the derivative of f with respect to matrix $\mathbf{A}$. For the RGMM, we use Eqs. (21)–(23) and (25). For Eq. (21) and Eq. (16), the RHS (Jacobian of the function with respect to the state variables) can be computed analytically, but for the general case, this becomes cumbersome to implement each time the objective function changes. Thus, the automatic differentiation tool JAX [64] was employed for the Jacobian of the function with respect to the state variables. This was also done for the direct contribution of matrix $\mathbf{A}$, as shown in the objective function Eq. (34), for the partial derivatives in Eqs. (20) and (25). Thus, the procedure of using JAX [64] for the Jacobian and direct contribution of matrix $\mathbf{A}$ is elaborated in Appendix G. Results from this derivative computation are shown in Table 2.

Next, the same function and matrix were taken (Eqs. (32) and (34)) and used in the SEMM approach shown in Section 3.3. The Eqs. (26)–(29) were used to compute the derivative. Once again, JAX [64] was used for evaluation of the Jacobian in Eq. (26) and for the partial derivatives of $\partial f/\partial \mathbf{A}$ in Eq. (28) (See Appendix G). Results from this derivative computation are shown in Table 2.

Table 2
Verification of the adjoint derivatives with FD for square matrix case.

Type	Index	Adjoint LGMM	Adjoint RGMM	Adjoint SEMM	FD
$df_r/d\mathbf{A}_r$	$(1,1)$	1.006352961803713	1.006352961803713	1.006352961803713	$1.006352\underline{068344540}$
$df_r/d\mathbf{A}_r$	$(1,2)$	0.043276271008604	0.043276271008604	0.043276271008604	$0.04327\underline{5413474930}$
$df_r/d\mathbf{A}_r$	$(1,3)$	−0.936930641525170	−0.936930641525170	−0.936930641525170	$-0.93693\underline{0476314046}$
$df_r/d\mathbf{A}_i$	$(1,1)$	0.063695888970744	0.063695888970744	0.063695888970744	$0.06369\underline{6809604608}$
$df_r/d\mathbf{A}_i$	$(1,2)$	0.082766443637231	0.082766443637231	0.082766443637231	$0.08276\underline{7300568776}$
$df_r/d\mathbf{A}_i$	$(1,3)$	0.156944460318835	0.156944460318835	0.156944460318835	$0.15694\underline{6299512128}$
$df_i/d\mathbf{A}_r$	$(1,1)$	−0.017846334274906	−0.017846334274906	−0.017846334274906	$-0.017846\underline{180533354}$
$df_i/d\mathbf{A}_r$	$(1,2)$	0.002833157022766	0.002833157022766	0.002833157022766	$0.002833\underline{298928806}$
$df_i/d\mathbf{A}_r$	$(1,3)$	0.006354411136774	0.006354411136774	0.006354411136774	$0.006354\underline{630599503}$
$df_i/d\mathbf{A}_i$	$(1,1)$	1.006719107202561	1.006719107202561	1.006719107202561	$1.0067191\underline{00082876}$
$df_i/d\mathbf{A}_i$	$(1,2)$	0.019954391307143	0.019954391307143	0.019954391307143	$0.019954\underline{295993330}$
$df_i/d\mathbf{A}_i$	$(1,3)$	0.003910503966226	0.003910503966226	0.003910503966226	$0.0039105\underline{11914285}$

The LGMM, RGMM, and SEMM derivatives were obtained for the same objective function and square matrix and, therefore, matched with each other. If the objective function f is $f = \sigma$, which is done by setting the other constants to zero and constant $c_\sigma = 1$ in Eq. (34), then the RAD formula Eq. (30) can be used to compute the derivative. Results from this derivative computation are shown in Table 4 for the square matrix $\mathbf{A}$ (Eq. (32)). To further establish confidence in the proposed methodology and results, a comparison between the obtained derivative results was made against the results from FD computations (Described in Appendix H), which matched too, as seen in Table 2.

4.2. Rectangular matrix derivative computation

Consider the tall imaginary matrix $\mathbf{A}$ as

$$\mathbf{A}_r + i\mathbf{A}_i = \begin{bmatrix} 6.3 & 5 \\ -5.35 & 0.62 \\ -7.49 & -1.6 \\ -0.15 & 0.71 \end{bmatrix} + i \begin{bmatrix} 4.49 & -9.95 \\ -1.23 & 7.29 \\ 6.17 & -1.9 \\ -4.89 & -3.63 \end{bmatrix}, \tag{38}$$

and the chosen singular variables are

$$\begin{aligned} \sigma &= 17.275386033399094, \\ \mathbf{u}_r + i\mathbf{u}_i &= \begin{bmatrix} -0.64373688 \\ 0.51599116 \\ 0.2346775 \\ -0.13537246 \end{bmatrix} + i \begin{bmatrix} -0.41836418 \\ 0.05007437 \\ -0.20205518 \\ 0.16611561 \end{bmatrix}, \\ \mathbf{v}_r + i\mathbf{v}_i &= \begin{bmatrix} -0.72661509 \\ 0.05431442 \end{bmatrix} + i \begin{bmatrix} 0 \\ -0.68489449 \end{bmatrix}. \end{aligned} \tag{39}$$

The objective function was taken to be same as in Eq. (34) and the constants were the same as well (Eq. (35)), except for $\mathbf{c}_u$ and $\mathbf{c}_v$ which were chosen to be

$$\mathbf{c}_u = \begin{bmatrix} 0.12 + 0.67i \\ 0.56 + 3.67i \\ 0.46 + 2.96i \\ 2.89 + 1.48i \end{bmatrix}, \quad \mathbf{c}_v = \begin{bmatrix} 7.12 + 0.97i \\ 0.26 + 6.47i \end{bmatrix}. \tag{40}$$

Once again, these constants were randomly selected. The constants $\mathbf{c}_u$ and $\mathbf{c}_v$ for the rectangular matrix were chosen differently because they must adhere to the shape of the vectors $\mathbf{u}$ and $\mathbf{v}$.

The same process as in Section 4.1 was followed for the derivative computation of each of the routes: LGMM, RGMM and SEMM, and the results were compared with FD. The results are tabulated in Table 3.

If the objective function f is $f = \sigma$, which is done by setting the other constants to zero and constant $c_\sigma = 1$ in Eq. (34), then the RAD formula Eq. (30) can be employed. Results from this derivative computation are shown in Table 5 for the rectangular matrix $\mathbf{A}$ (Eq. (38)).

4.3. Scalability of the proposed methodology

In this section, the computational speed and scalability study of the proposed methodology was conducted by measuring the mean wall time taken for real-valued LGMM, RGMM, SEMM, FD and RAD, for multiple square matrices of increasing matrix size. Complex-valued would yield similar results. However, the RAD-formula from Townsend [58] in Eq. (K.2) is for purely real-valued

Table 3
Verification of the adjoint derivatives with FD for rectangular matrix case.

Type	Index	Adjoint LGMM	Adjoint RGMM	Adjoint SEMM	FD
$df_r/d\mathbf{A}_r$	(1, 1)	1.846102900714162	1.846102900714162	1.846102900714162	1.846101064018058
$df_r/d\mathbf{A}_r$	(1, 2)	−0.006821647620363	−0.006821647620363	−0.006821647620363	−0.006822084230862
$df_r/d\mathbf{A}_i$	(1, 1)	0.354647919899091	0.354647919899091	0.354647919899091	0.354647895051130
$df_r/d\mathbf{A}_i$	(1, 2)	−0.124582311966832	−0.124582311966832	−0.124582311966832	−0.124581340799068
$df_i/d\mathbf{A}_r$	(1, 1)	0.227870193134751	0.227870193134751	0.227870193134751	0.227870147639919
$df_i/d\mathbf{A}_r$	(1, 2)	0.353104252473483	0.353104252473483	0.353104252473483	0.353103555283951
$df_i/d\mathbf{A}_i$	(1, 1)	0.780271281223158	0.780271281223158	0.780271281223158	0.780273927247777
$df_i/d\mathbf{A}_i$	(1, 2)	0.113513089065063	0.113513089065063	0.113513089065063	0.113512610866451

Table 4
Verification of the RAD formula for singular value derivative with FD for the square matrix.

Type	Index	RAD	FD
$d\sigma/d\mathbf{A}_r$	(1, 1)	0.018703061899253	0.018702181137087
$d\sigma/d\mathbf{A}_r$	(1, 2)	0.068881276214858	0.068880360970525
$d\sigma/d\mathbf{A}_r$	(1, 3)	−0.934470093986586	−0.934470051561220
$d\sigma/d\mathbf{A}_i$	(1, 1)	0.080015153716675	0.080016050674203
$d\sigma/d\mathbf{A}_i$	(1, 2)	0.076615998520789	0.076616849753464
$d\sigma/d\mathbf{A}_i$	(1, 3)	0.160118791389606	0.160120613657000

Table 5
Verification of the RAD formula for singular value derivative with FD for the rectangular matrix.

Type	Index	RAD	FD
$d\sigma/d\mathbf{A}_r$	(1, 1)	0.467749108787955	0.467748947130531
$d\sigma/d\mathbf{A}_r$	(1, 2)	0.251572392322310	0.251571147913410
$d\sigma/d\mathbf{A}_i$	(1, 1)	0.303989439817427	0.303989750705114
$d\sigma/d\mathbf{A}_i$	(1, 2)	−0.463615299320613	−0.463615023704733

matrices and therefore we elected to go with a purely real-valued comparison. The adjoint equations were solved using LAPACK as part of the linalg library of NumPy in Python [65].

The matrices were ensured to be full rank matrices and randomized for each value of the $N \times N$ matrix. In all of the derivative approaches, the computational wall time measurement was done carefully.

For the real-valued part, only the real-valued parts of LGMM, RGMM and SEMM were compared against that of the real-valued RAD formula for SVD derivative formula proposed by Townsend [58] Eqs. (16), (16), (26) and (K.2).

It can be seen that for just the real-valued part, the RAD formula is relatively expensive in computational effort compared to the adjoint. Further, efficient preconditioning of the solver for the adjoint equation could improve the speed of the computation (See Section 4.3 from He et al. [44] for more details).

For the RAD-adjoint comparison study performed, the imaginary part of the matrix is just a zero-matrix, but is inevitably present in the adjoint equation, for instance, as $\mathbf{A}_i$ in Eq. (27). This simply increases the sparsity of the adjoint coefficients matrix $\mathbf{M}_f$ from Eq. (26). A sparse solver would therefore be even faster. Alternately a purely real-valued formulation for the adjoint method would result in lesser computational time. Even with this, the adjoint method performs better. Further, it is noted that for the comparison cases of proposed methods with FD, the SVD computation time was immaterial as the same algorithm as in SVD solver NumPy [65] was used.

However, for the case of RAD comparison as shown in Fig. 1, the RAD method requires all the singular variables whereas our methods require only the singular variables under question. Thus, the computation time for the full SVD through NumPy was added to the computational time measured for RAD, while for our methods, a real-valued non-linear iterative solver was made to solve Eq. (8). The computational time to run this for only the singular variables under consideration was measured and added to the times measured for the derivative computations for our methods.

Finally, it is evident from Fig. 1 that the FD scales poorly with the number of inputs and during the computational study of comparison of these methods, the matrix size could not be increased beyond the final size attained in Fig. 1 as the time taken was of the order 10^5 seconds, and thus not feasible, making our case of scalability issue with FD.

4.4. Large dataset derivative computation

In this section, we compute the derivative of the first six singular values from the proper orthogonal decomposition of the fluid flow over a flat plate. This flow is transitioning from laminar to turbulent regime and is sourced from JHTDB [62]. We use the RAD based singular value derivative for the real-valued case Eq. (31) on the JHTDB dataset.

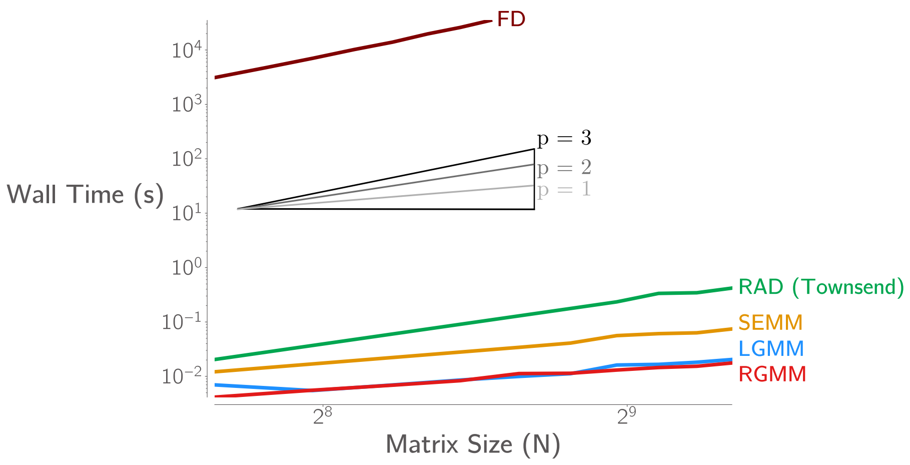

Fig. 1. Computational efficiency comparison of real-valued proposed methods, FD and RAD. p represents the index for slope as a reference and $p = 1$ corresponds to our proposed methods, $p = 2$ is for RAD and $p = 3$ is for FD in the plot. The smaller slope and lower wall time, yields better scalability.

4.4.1. Dataset description

The laminar–turbulent transitional flow dataset from JHTB was used in this study [62]. This data-set was produced from direct numerical simulation of incompressible flow of fluid over a flat plate in a developing boundary layer. The flat plate was given a leading edge and the Reynolds number $Re = 800$, where $Re = U_\infty L/\nu$. Here, $U_\infty = 1$ is the non-dimensional free stream velocity, $L = 1$ is the length scale set to half-plate thickness and $\nu = 1.24 \times 10^{-3}$ is the kinematic viscosity.

The dataset is a rectangular domain in 3D with set of grid points in cartesian coordinates (x, y, z), each having the state variables of x-component of velocity u, y-component of velocity v, z-component of velocity w and the pressure P. All values in the database are non-dimensionalized. The database domain is from $[30.2185L, 1000L]$ in the x direction, $[0, 26.48L]$ in the y direction and $[0, 240L]$ in the z direction. The cutout was taken from $[100L, 600L]$ in the x direction, $[0, 10L]$ in the y direction and $[80L, 120L]$ in the z direction, which resulted in a domain size of $(500L \times 10L \times 40L)$. The number of grid points in the cutout was $n_x \times n_y \times n_z = 1712 \times 85 \times 342$. Here n_x, n_y and n_z are the number of grid points in the x, y and z directions, respectively. Thus, there are $m_0 = 1712 \times 85 \times 342 = 49.76784 \times 10^6$ spatial points. The cutout time stored is $\bar{t} \in [1000, 1175]$ where $\bar{t} = L/U_\infty$ is the non-dimensional time. Thus, there were 75 time-steps stored, with each step having a physical time of $\delta\bar{t} = 0.25$ giving a total physical time, $\bar{t}$, equal to $75 \times 0.25 = 18.75$. The last 75 time steps were chosen after studying the flow field carefully and ensuring a statistically stationary state, where by stationary we mean the transition location. Further details on the grid and velocity data, the turbulence characteristics and other flow related data can be found at JHTDB [62].

4.4.2. POD and its relationship with SVD

POD of the obtained data cutout was performed on the velocity data. Thus, there are $m = n_x \times n_y \times n_z \times 3 = m_0 \times 3 = 149.30352 \times 10^6$ states for each time step. In POD, the time-series data is arranged into a snapshot matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ where m is the number of states arranged in a vectorized form for each time step and n equals to the number of time-steps or snapshots of the data. SVD on this snapshot matrix is performed to obtain the modes, their energies and their temporal coefficients. In the current study, since velocity data was used for the POD, the matrix $\mathbf{X}$ is defined as

$$\mathbf{X} = \left[\begin{array}{cccc} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{m_0,1} & u_{m_0,2} & \cdots & u_{m_0,n} \\ \hline v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ v_{m_0,1} & v_{m_0,2} & \cdots & v_{m_0,n} \\ \hline w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m_0,1} & w_{m_0,2} & \cdots & w_{m_0,n} \end{array}\right], \tag{41}$$

where $u_{i,k}$ is the x-directional velocity at the ith grid point and kth time-step; similar for $v_{i,k}$ and $w_{i,k}$. The number of time-steps is $n = 20$ as mentioned earlier. Instead of conducting POD on the original dataset, it is recommended to instead conduct the POD with respect to the perturbed state [34]. The perturbed snapshot matrix is defined as

$$\mathbf{X}' = \mathbf{X} - \frac{1}{n}\mathbf{X}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}}, \tag{42}$$

where $\mathbf{X}' \in \mathbb{R}^{m\times n}$ is the perturbed snapshots matrix, and $\mathbf{1}_n \in \mathbb{R}^n$ is a vector of ones with dimension n. SVD is then performed on this matrix $\mathbf{X}'$ which yields

$$\mathbf{X}' = \mathbf{\Phi}\mathbf{\Sigma}\boldsymbol{\Psi}^{\mathsf{T}}, \tag{43}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{m\times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$, and $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}$. Columns of $\mathbf{\Phi}$ contain the POD modes, $\mathbf{\Sigma}$ is the rectangular matrix with singular values on the main diagonal and zeros elsewhere, and each of these values contains the energy associated with the corresponding mode. The POD was performed through the method of snapshots (See Appendix I) that gave the singular values and vectors as it is an efficient way to perform POD [34].

4.4.3. POD derivative results

The results of the POD computations along with the full order model (FOM) are shown in Fig. 2. The first, third and sixth modes are shown for sake of compactness and representation, although all of the modes were captured in the computations. The velocity data was visualized using the Q-criterion with Q-2 iso-surfaces of 0.001 in magnitude and colored by the u-velocity magnitude of the FOM flow field [66]. The Q-criterion identifies vortices as regions where the rotation rate dominates over the strain rate. It is defined as

$$Q = \frac{1}{2}\left(\|\mathbf{\Omega}\|^2 - \|\mathbf{S}\|^2\right), \tag{44}$$

where Q is the Q-criterion value, $\mathbf{\Omega}$ is the antisymmetric rotation tensor, $\mathbf{S}$ is the symmetric strain-rate tensor, $\|\mathbf{\Omega}\|^2$ is the squared norm of the rotation tensor, and $\|\mathbf{S}\|^2$ is the squared norm of the strain-rate tensor. These tensors are defined as

$$\mathbf{S} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathsf{T}}\right), \tag{45}$$

where $\mathbf{u}$ is the velocity vector containing the x, y and z component of velocities in a vectorized form and $\nabla\mathbf{u}$ is the velocity gradient tensor. The rotation tensor is

$$\mathbf{\Omega} = \frac{1}{2}\left(\nabla\mathbf{u} - \nabla\mathbf{u}^{\mathsf{T}}\right). \tag{46}$$

The squared norm $\|\cdot\|^2$ for the tensors is defined as

$$\|\mathbf{S}\|^2 = \mathrm{Tr}\left(\mathbf{S}^{\mathsf{T}}\mathbf{S}\right), \tag{47}$$

for the symmetric strain tensor and

$$\|\mathbf{\Omega}\|^2 = \mathrm{Tr}\left(\mathbf{\Omega}^{\mathsf{T}}\mathbf{\Omega}\right), \tag{48}$$

for the anti-symmetric strain tensor.

In regions where $Q > 0$, the local rotation dominates over strain, and these regions are identified as vortices [66]. The Q iso-surfaces were visualized at iso values of $Q = 0.001$ in Fig. 2. These Q-contours were made at four selected time values t_1, t_2, t_3, t_4 such that $t_1 < t_2 < t_3 < t_4$. The non-dimensional time values were $t_1 = 1075$, $t_2 = 1076.5$, $t_3 = 1078$ and $t_4 = 1080$.

Physically, the derivative computation of the singular values is the measure of how sensitive each singular value is to perturbations in the original snapshot matrix. Thus, we get three directions of derivatives of the velocities as there are three components of the velocity data u, v and w. With reference to Eq. (41), we can see that

$$\frac{\mathrm{d}\sigma_i}{\mathrm{d}\mathbf{X}} = \left[\begin{array}{cccc}
\frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{1,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{1,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{1,n}} \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{2,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{2,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{2,n}} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{m_0,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{m_0,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}u_{m_0,n}} \\
\hline
\frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{1,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{1,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{1,n}} \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{2,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{2,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{2,n}} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{m_0,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{m_0,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}v_{m_0,n}} \\
\hline
\frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{1,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{1,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{1,n}} \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{2,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{2,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{2,n}} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{m_0,1}} & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{m_0,2}} & \cdots & \frac{\mathrm{d}\sigma_i}{\mathrm{d}w_{m_0,n}}
\end{array}\right], \tag{49}$$

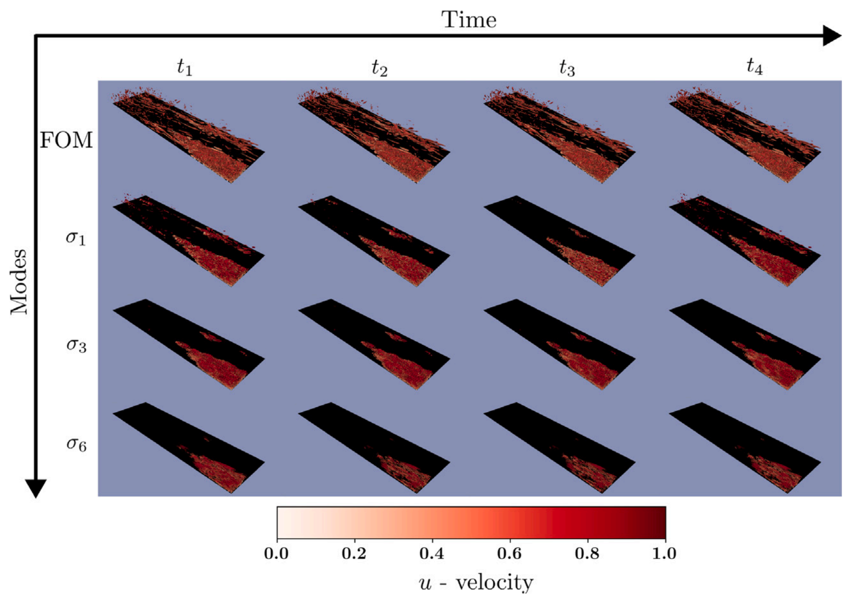

Fig. 2. POD modes of the JHTDB transitional boundary layer dataset. Vortical structures are Q iso-surfaces (Q = 0.001).

where $\mathrm{d}\sigma/\mathrm{d}u$ is the derivative of the singular value with respect to the u-velocity. Similarly this holds for $\mathrm{d}\sigma/\mathrm{d}v$ and $\mathrm{d}\sigma/\mathrm{d}w$.

From this, it is clear that the singular value derivative is directional in the three dimensioned space it is evaluated for through POD. The snapshot matrix had 149.30352×10^6 rows equal to the number of states and 75 columns equal to the number of time steps as aforementioned and the derivatives of the singular value with respect to this snapshot matrix were computed using the RAD formula shown in Eq. (31).

Results from this derivative computation are shown in Fig. 3 which showcases the derivatives of the first, third and sixth modes along the x-direction for the first time-step. In this figure, the Q-criterion plot at $Q = 0.001$ for the FOM was made and colored by the singular value derivatives. These Q-contours were again made for four selected time values t_1, t_2, t_3, t_4 such that $t_1 < t_2 < t_3 < t_4$. The non-dimensional time values were $t_1 = 1075$, $t_2 = 1076.5$, $t_3 = 1078$ and $t_4 = 1080$. The following observations were made regarding the flow physics and derivatives of the singular values with respect to the snapshot matrix. The flow over flat plate case shown is an example of bypass transition. In typical transition flow, Tollmien–Schlichting waves (TS waves) are seen when the freestream turbulence is negligible [67]. In the case when the free stream turbulence intensity is about 3.5% the transition occurs with formation of Klebanoff streaks as shown in Fig. 4 in a case called the bypass transition, which essentially “bypasses” the formation of TS waves [62].

Downstream of these streaks, the hairpin vortices can be seen which are identified in Fig. 4 too. Once the “lift-up” effect kicks in, the mixing becomes stronger, more chaotic and the flow transitions to fully turbulent flow [68]. The current image snapshots of modes and derivatives of the singular values visualized in Figs. 2 and 3 are shown for the transitional region, which includes the Klebanoff streaks and the hairpin vortices just before the flow transitions to fully turbulent flow [69].

The derivative computation results reveal that the flow structures right before the flow turns fully turbulent are more sensitive to perturbations than the larger and elongated Klebanoff streaks which appear at the start of the transitioning flow. This suggests that the flow receptivity increases as the flow approaches the fully turbulent regime. The derivatives fluctuate just as the singular values do with time, which is obvious as the modes are scaled by the singular values along with the temporal coefficients which typically portray a sinusoidal like variation.

Comparing the identified vortical structures as in Fig. 4 with the structures in Fig. 3, it can be seen that the Klebanoff streaks and larger vortical structures at the start of the domain as the flow enters to flow over the flat plate, the structures do not show much receptivity to perturbations in the flow. However, the smaller structures like the free stream turbulence as well as the hairpin vortices seem to be more sensitive to perturbations.

However, in the temporal sense, it can be seen in Fig. 3 there comes snapshots in time when there is little to no receptivity in the flow for the smaller modes. In fact the receptivity of the modes changes periodically with time as do the temporal coefficients. The variations are much larger for the smaller modes however, as compared to the more dominant ones.

This kind of identification of sensitive regions of the modes in the flow field offers a pathway for further exploration of receptivity theory and transition to turbulence. This work serves as a foundation for future studies aimed at controlling or predicting flow transitions, with potential applications in aerodynamic design, turbulence modeling, and flow control strategies. This is the general spatial trend observed.

The derivative of singular value of large dataset can be taken using the RAD formula and was shown in this section. In lieu of this, an in-depth analysis of the flow physics here should be sought, but is deemed as future work by the authors and is therefore beyond the scope of the current manuscript.

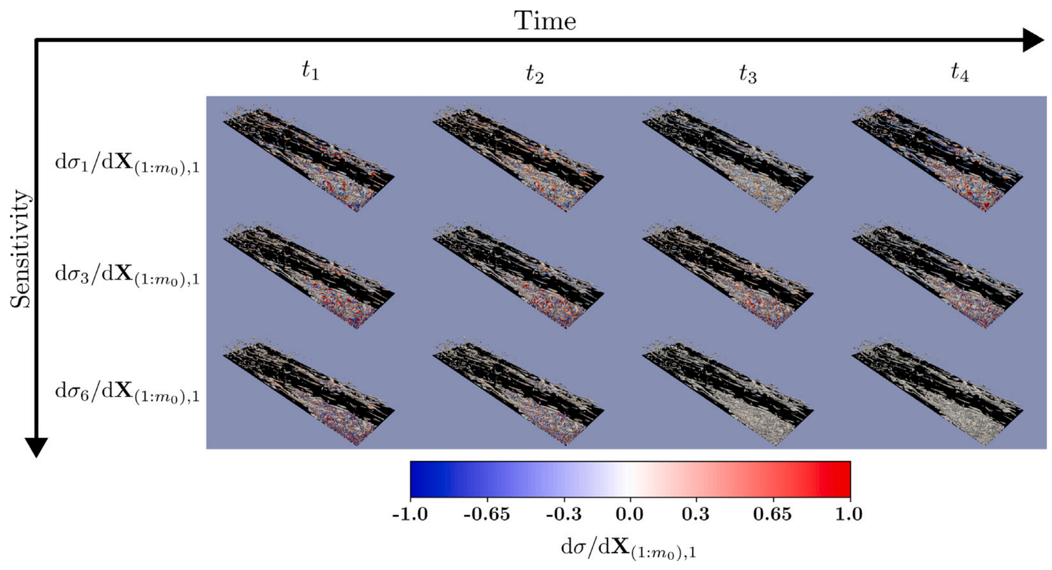
Time
t_1
t_2
t_3
t_4
Sensitivity
$\mathrm{d}\sigma_1/\mathrm{d}\mathbf{X}_{(1:m_0),1}$
$\mathrm{d}\sigma_3/\mathrm{d}\mathbf{X}_{(1:m_0),1}$
$\mathrm{d}\sigma_6/\mathrm{d}\mathbf{X}_{(1:m_0),1}$
-1.0
-0.65
-0.3
0.0
0.3
0.65
1.0
$\mathrm{d}\sigma/\mathrm{d}\mathbf{X}_{(1:m_0),1}$

Fig. 3. x-direction singular value derivatives of the POD modes of JHTDB transitional boundary layer dataset. Vortical structures are Q iso-surfaces (Q = 0.001).

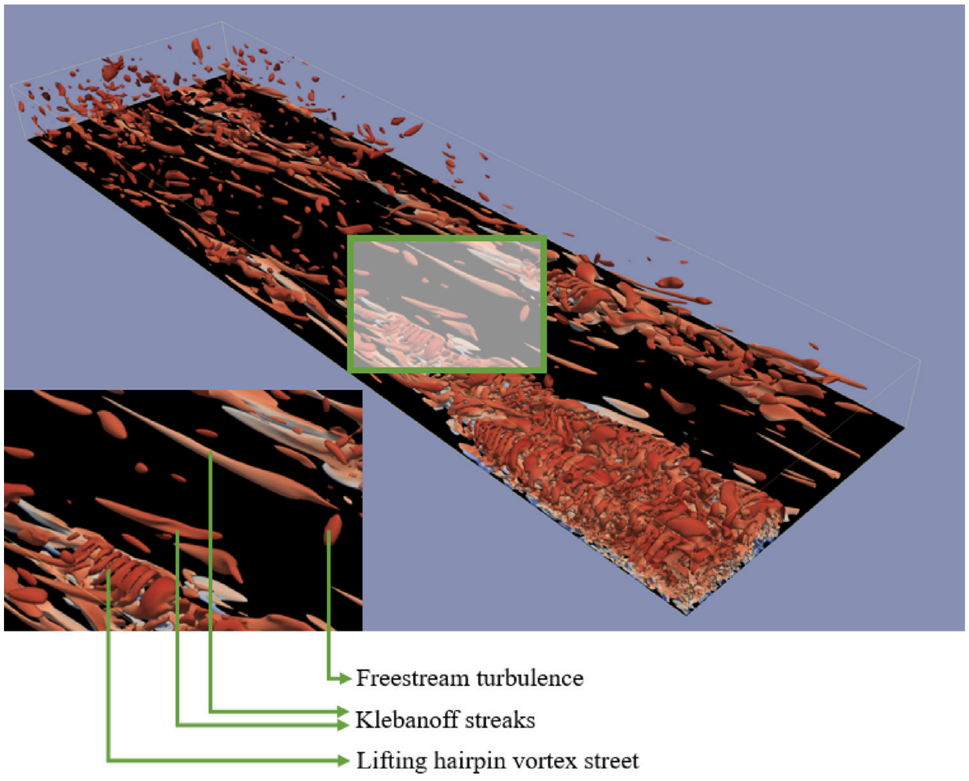

Fig. 4. Key vortical structures in the transitional region of the flow over flat plate visualized by (Q = 0.001) iso-surfaces colored by velocity-magnitude.

4.5. Wing truss compliance optimization using differentiable SVD

In the current section we discuss optimization the bar thickness of a 315-bar wing-truss from the common research model Brooks et al. [70] by optimizing the singular value of the response matrix to a minimum, essentially reducing the compliance of the structure.

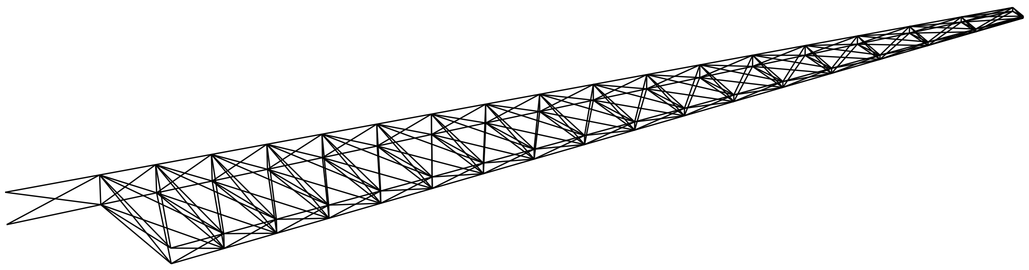

Fig. 5. The common research model 315 bar wing-truss [71].

We refer the readers to Ersoy and Mugan [29], from where this example's physics was sourced in which the design sensitivity was studied using SVD. They showed in their paper that the stiffness of a static structure is a measure of the leading singular value of the response matrix of the system.

The net static response of a structure is compliance of that structure and this is inversely proportional to its stiffness. Thus, reduction of the leading singular value of the response matrix of the system reduces the compliance [29]. More details on the mathematical considerations of this statement can be found in Appendix M. Shahabsafa et al. [71] used the common research model wing in their design optimization to minimize weight. In the current study, we optimized the net static response of the same common research model wing truss by optimizing the bar thicknesses in minimization of the leading singular value of the response matrix (Refer to Appendix M for more details on this matrix) subject to the constraint that the net mass of the truss is constant and bounds placed on the bar areas.

Thus, mass was re-distributed inside the truss amongst the bars and the optimization was carried out. The motivation of this task is to demonstrate the optimization of a wing-truss system using the RAD formula for the derivative computation subject to multiple design variables, which in this case is 315 since we have 315 bars. All values in this optimization such as the material properties and the member areas were sourced as is from Shahabsafa et al. [71] and Brooks et al. [70]. The undeformed wing truss is shown in Fig. 5.

We now discuss the derivative computation of the singular value with respect to the design variables — the bar areas. The derivative of the first singular value of the response matrix with respect to each of these design variables is a vector given by the matrix–vector product as

$$\frac{\mathrm{d}\sigma_1}{\mathrm{d}\mathbf{b}} = \mathrm{vec}\left(\frac{\mathrm{d}\sigma_1}{\mathrm{d}\mathbf{K}}\right)^{\mathsf{T}} \cdot \left[\mathrm{vec}\left(\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}b_1}\right)\ \ \mathrm{vec}\left(\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}b_2}\right) \cdots \mathrm{vec}\left(\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}b_n}\right)\right], \tag{50}$$

where each entry of the vector $\mathrm{d}\sigma_1/\mathrm{d}\mathbf{b}$ has the derivative of σ_1 with respect to the entry's b_i, n is the net number of elements of the truss, which is 315 in the undertaken wing truss problem and $\mathrm{d}\sigma_1/\mathrm{d}\mathbf{b} \in \mathbb{R}^{n_b}$ where n_b is the number of bars in the truss. The same derivative can also be obtained using FD in which each element of $\mathbf{b}$ is perturbed to compute the perturbation in σ_1 and this is computed as $\mathrm{d}\sigma_1/\mathrm{d}\mathbf{b}$. This approach however requires as many derivative function calls as there are design variables. For each entry of b_i, FD requires evaluation of the inverse of stiffness matrix and SVD of this inverse matrix and this becomes computationally intensive for several design variables as was shown in the scalability analysis Section 4.3.

The optimization problem is formulated as follows

$$\begin{aligned} \min_{\mathbf{b}} \quad & \sigma_1, \\ \text{subject to} \quad & \underline{b} \le b_i \le \bar{b}, \quad i = 1, \dots, n_{\mathbf{b}}, \\ & \sum_{i=1}^{n} b_i\ l_i = v_0, \end{aligned} \tag{51}$$

where l_i is the length of the i–th bar, $\underline{b}$ is 2.50×10^{-5} m^2, $\bar{b}$ is 0.12 m^2, and $v_0 = 6.79$ m^3 is the initial volume. The constraint placed is the volume constraint. The truss optimization was initialized with the uniform thickness throughout given by area of each element set to 0.01 m^2. All material properties, nodes and elemental connections are sourced from Shahabsafa et al. [71].

The optimization package used was Scipy package's optimizer [72] and the optimizer used was based on the trust region methods of gradient based optimization [73].

The results of optimization are shown in Fig. 6. The objective was minimized with the tolerance of 10^{-6} resulting in a final singular value of 1.6×10^{-4} m/N from the starting singular value of 1.02×10^{-3} m/N. Thus, the net static response of the structure was reduced by 81% post optimization, where the net mass of the truss was re-distributed among the members. In Fig. 6, the thickness of the lines represents the thickness of the bars which was optimized. Each bar's color represents the leading singular value's sensitivity to the changes in that particular bar's thickness.

Thus, the RAD form of the differentiable SVD approach for real-valued inputs and outputs was used to showcase an optimization of the thickness of the members of a wing truss system to reduce the net static response and the optimization was carried out successfully.

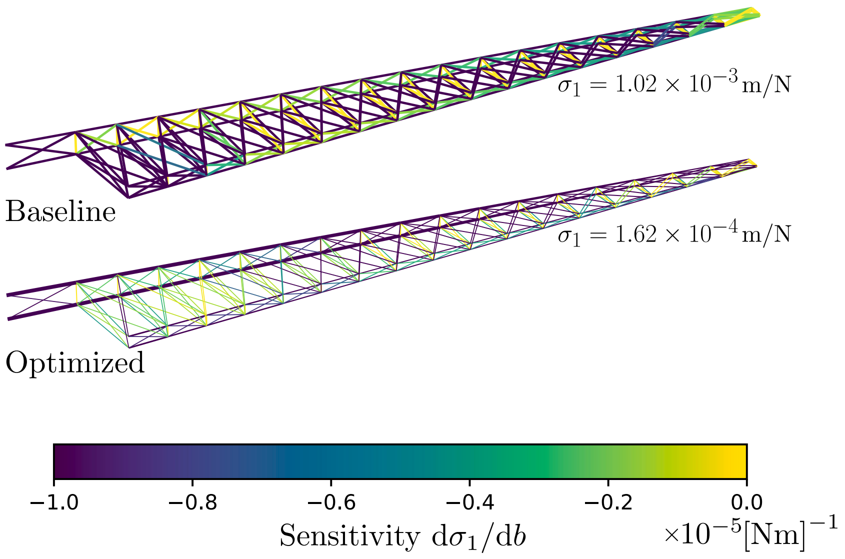
$\sigma_1 = 1.02 \times 10^{-3}$m/N
Baseline
$\sigma_1 = 1.62 \times 10^{-4}$m/N
Optimized
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
Sensitivity dσ_1/db
$\times 10^{-5}$[Nm]$^{-1}$

Fig. 6. The baseline and optimized wing-trusses. The bar thickness is represented by the line thickness in the plot. Bars are colored by sensitivity magnitude of the singular value.

5. Conclusions

In this paper, we developed two adjoint-based methods to compute the singular values and singular vector derivative and an RAD formula to compute the singular value derivative. The proposed differentiable SVD algorithms do not scale with the number of inputs and do not require all of the singular variables during derivative computation, thus reducing the computational cost, they are relatively easier to implement, and are accurately, being implicit analytic in nature of the solution. All the derivative computation strategies proposed in the current manuscript can handle complex-valued inputs and outputs.

We proposed two adjoint method approaches leveraging the relationship between EVP and SVD. In the first approach, we leverage the relationship between the SVD of a matrix and the EVP of its Gram matrices. We call it the Gram matrix method or GMM. In the second approach, we leverage the relationship between the SVD of a matrix and the EVP of its symmetric embedding. We call it the symmetric embedding matrix method or SEMM. The adjoint-based GMM computes the derivative by breaking down the SVD into two eigenvalue problems of the Gram matrices and applies the adjoint method to the governing equations for each EVP. This results in two formulations, the Left Gram Matrix Method or LGMM and the Right Gram Matrix Method RGMM, deriving their names from the fact that each Gram matrix either yields the left or right singular vectors in their EVP solutions.

The adjoint-based SEMM computes this derivative by directly applying the adjoint method to the SVD governing equations resulting from the symmetric embedding EVP. Each method holds its merits based on the objective function whose derivative is desired and the nature of matrix **A** being sparse or dense. If the matrix is dense, the computation of the Gram matrices of the matrix can become costly. To avoid this, the SEMM is used. If the matrix is sparse, the former method, GMM, should be used. If the matrix is square, LGMM or RGMM are both equivalent. If the matrix is tall, LGMM is advised to be used, otherwise, RGMM. Finally, we also proposed a general RAD dot product identity for general complex-valued functions that are imaginary-differentiable.

The results from derivative computations of two imaginary matrices, square and rectangular, were compared against the results from FD approximations, and we achieved a 5–6 digit match. A scalability analysis was carried out to compare and contrast the differences in the computational efficiencies of the proposed methods, RAD formulae in the literature and FD, which resulted in faster computational speeds for the proposed adjoint-RAD methods.

An implementation of the real-valued form of the proposed singular value derivative RAD formula was shown by applying the formula on the snapshot matrix of a large cutout of the JHTDB dataset in a 3-dimensional space and obtaining the singular values and vectors through POD. The snapshot matrix had 149.30352×10^6 rows equal to the number of states and 75 columns equal to the number of time steps. In another example of wing-truss optimization, the differentiable framework was added as part of an optimization problem of reducing the truss compliance of a 315 bar wing truss source from the undeformed common research model wing. The optimization was carried out successfully with an 81% reduction in the singular value.

There is no limitation set on the objective function under consideration, making the proposed method well-suited to large-scale design optimization problems involving gradients of SVD with respect to several design variables, for instance, in differentiable POD or differentiable resolvent analyses that find applications in various engineering-design optimization problems.

CRediT authorship contribution statement

Rohit Sunil Kanchi: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sicheng He:** Writing – review & editing, Supervision, Resources, Project administration, 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.

Acknowledgment

The first author would like to thank Ariel Lubonja from John Hopkins University for help with the data retrieval process of the JHTDB flat plate boundary layer dataset.

Appendix A. Adjoint method for EVP

We derive the adjoint equation for any function, $f = f(\mathbf{w}, \mathbf{D}(\mathbf{x}))$, where $\mathbf{D}$ is the input EVP coefficient matrix and $\mathbf{w}$ is the solution of the EVP. The coefficient matrix can be further parameterized by design variable, $\mathbf{x}$. Thus, expanding the adjoint equations Eq. (13) and Eq. (14) from Section 2, we have

$$\begin{bmatrix} \mathbf{D}_r - \lambda_r \mathbf{I} & -\mathbf{D}_i + \lambda_i \mathbf{I} & -\boldsymbol{\phi}_r & \boldsymbol{\phi}_i \\ \mathbf{D}_i - \lambda_i \mathbf{I} & \mathbf{D}_r + \lambda_r \mathbf{I} & -\boldsymbol{\phi}_i & -\boldsymbol{\phi}_r \\ 2\boldsymbol{\phi}_r^\intercal & 2\boldsymbol{\phi}_i^\intercal & 0 & 0 \\ 0 & \boldsymbol{e}_k^\intercal & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{\text{main,r}} \\ \boldsymbol{\psi}_{\text{main,i}} \\ \boldsymbol{\psi}_m \\ \boldsymbol{\psi}_p \end{bmatrix} = \frac{\partial f}{\partial \mathbf{w}}^\intercal, \tag{A.1}$$

which is the adjoint equation for the EVP of matrix $\mathbf{D}$. The partial derivative of the residual matrix with respect to the imaginary parts of the matrix $\mathbf{D}$ is

$$\begin{aligned} \frac{\partial \mathbf{r}}{\partial \mathbf{D}_r} &= \boldsymbol{\psi}_{\text{main,r}} \boldsymbol{\phi}_r^\intercal + \boldsymbol{\psi}_{\text{main,i}} \boldsymbol{\phi}_i^\intercal, \\ \frac{\partial \mathbf{r}}{\partial \mathbf{D}_i} &= -\boldsymbol{\psi}_{\text{main,r}} \boldsymbol{\phi}_i^\intercal + \boldsymbol{\psi}_{\text{main,i}} \boldsymbol{\phi}_r^\intercal, \end{aligned} \tag{A.2}$$

and the partial derivatives of imaginary components of f with respect to those of the matrix $\mathbf{D}$ are

$$\begin{aligned} \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{D}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{D}_r}^\intercal \boldsymbol{\psi}_r, \\ \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{D}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{D}_i}^\intercal \boldsymbol{\psi}_r, \\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{D}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{D}_r}^\intercal \boldsymbol{\psi}_i, \\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{D}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{D}_i}^\intercal \boldsymbol{\psi}_i. \end{aligned} \tag{A.3}$$

This completes the adjoint method for solving EVP. Detailed approach can be found in He et al. [44].

Appendix B. Chain rule RAD formula

Consider the matrix $\mathbf{B} = \mathbf{A}\mathbf{A}^*$. An FAD application on both sides results in the real part as

$$\dot{\mathbf{B}}_r = \dot{\mathbf{A}}_r \mathbf{A}_r^\intercal + \mathbf{A}_r \dot{\mathbf{A}}_r^\intercal + \dot{\mathbf{A}}_i \mathbf{A}_i^\intercal + \mathbf{A}_i \dot{\mathbf{A}}_i^\intercal, \tag{B.1}$$

and the imaginary part as

$$\dot{\mathbf{B}}_i = \dot{\mathbf{A}}_i \mathbf{A}_r^\intercal + \mathbf{A}_i \dot{\mathbf{A}}_r^\intercal - \dot{\mathbf{A}}_r \mathbf{A}_i^\intercal - \mathbf{A}_r \dot{\mathbf{A}}_i^\intercal. \tag{B.2}$$

These two terms $\dot{\mathbf{B}}_r$ and $\dot{\mathbf{B}}_i$ can be used in the Trace expression in Eq. (J.4) as

$$\mathrm{Tr}[\overline{\mathbf{B}}_r^\intercal \dot{\mathbf{B}}_r + \overline{\mathbf{B}}_i^\intercal \dot{\mathbf{B}}_i] = \mathrm{Tr}[\overline{\mathbf{A}}_r^\intercal \dot{\mathbf{A}}_r + \overline{\mathbf{A}}_i^\intercal \dot{\mathbf{A}}_i]. \tag{B.3}$$

Upon applying the Tr identities and expanding LHS of Eq. (B.3), we have

$$\begin{aligned} \mathrm{Tr}[(\mathbf{A}_r^\intercal \overline{\mathbf{B}}_r^\intercal + \mathbf{A}_r^\intercal \overline{\mathbf{B}}_r + \mathbf{A}_i^\intercal \overline{\mathbf{B}}_i - \mathbf{A}_i^\intercal \overline{\mathbf{B}}_i^\intercal) \dot{\mathbf{A}}_r + \\ (\mathbf{A}_i^\intercal \overline{\mathbf{B}}_r^\intercal + \mathbf{A}_i^\intercal \overline{\mathbf{B}}_r + \mathbf{A}_r^\intercal \overline{\mathbf{B}}_i^\intercal - \mathbf{A}_r^\intercal \overline{\mathbf{B}}_i) \dot{\mathbf{A}}_i] \\ = \mathrm{Tr}[\overline{\mathbf{A}}_r^\intercal \dot{\mathbf{A}}_r + \overline{\mathbf{A}}_i^\intercal \dot{\mathbf{A}}_i], \end{aligned} \tag{B.4}$$

which upon simplifying we get for the real part

$$\bar{\mathbf{A}}_r = (\mathbf{A}_r^\intercal \bar{\mathbf{B}}_r^\intercal + \mathbf{A}_r^\intercal \bar{\mathbf{B}}_r + \mathbf{A}_i^\intercal \bar{\mathbf{B}}_i - \mathbf{A}_i^\intercal \bar{\mathbf{B}}_i^\intercal)^\intercal, \tag{B.5}$$

and for the imaginary part

$$\bar{\mathbf{A}}_i = (\mathbf{A}_i^\intercal \bar{\mathbf{B}}_r^\intercal + \mathbf{A}_i^\intercal \bar{\mathbf{B}}_r + \mathbf{A}_r^\intercal \bar{\mathbf{B}}_i^\intercal - \mathbf{A}_r^\intercal \bar{\mathbf{B}}_i)^\intercal. \tag{B.6}$$

Now we can place the reverse seed as g_r and g_i in each of Eq. (B.5) and Eq. (B.6) to obtain the derivatives as in Eq. (20).

Consider then the matrix $\mathbf{C} = \mathbf{A}^*\mathbf{A}$ Following the same methods as in Eqs. (B.3)–(B.6), we have for the real part

$$\bar{\mathbf{A}}_r = (\bar{\mathbf{C}}_r \mathbf{A}_r^\intercal + \bar{\mathbf{C}}_r^\intercal \mathbf{A}_r^\intercal + \bar{\mathbf{C}}_i \mathbf{A}_i^\intercal - \bar{\mathbf{C}}_i^\intercal \mathbf{A}_i^\intercal)^\intercal, \tag{B.7}$$

and for the imaginary part

$$\bar{\mathbf{A}}_i = (\bar{\mathbf{C}}_r \mathbf{A}_i^\intercal + \bar{\mathbf{C}}_r^\intercal \mathbf{A}_i^\intercal + \bar{\mathbf{C}}_i \mathbf{A}_r^\intercal - \bar{\mathbf{C}}_i^\intercal \mathbf{A}_r^\intercal)^\intercal. \tag{B.8}$$

We can then seed h_r and h_i each into Eqs. (B.7) and (B.8) to obtain the derivatives in Eq. (25).

Appendix C. Vectorization

The vectorization operator $\operatorname{vec}(\cdot)$ is defined as

$$(\operatorname{vec}(\mathbf{A}))_{i\times(n_2-1)+j} = \mathbf{A}_{ij}, \quad i = 1,\dots,n_1, \quad j = 1,\dots,n_2, \tag{C.1}$$

where $\mathbf{A} \in \mathbb{R}^{n_1\times n_2}$, $\operatorname{vec} : \mathbb{R}^{n_1\times n_2} \to \mathbb{R}^{n_1 n_2}$, and the subscript represents the index of an element from the matrix $\mathbf{A}$ or the vector $\operatorname{vec}(\mathbf{A})$. This linear operation transforms a matrix into a vector simplifying the matrix derivative computation. The inverse vectorization operator $\operatorname{vec}^{-1}(\cdot)$ is defined as,

$$\operatorname{vec}^{-1}(\operatorname{vec}(\mathbf{A})) = \mathbf{A}, \tag{C.2}$$

for arbitrary matrix $\mathbf{A}$. As a special case of the properties of vectorization,

$$\begin{aligned} \operatorname{vec}(\mathbf{a}) &= \mathbf{a}, \\ \operatorname{vec}^{-1}(\mathbf{a}) &= \mathbf{a}, \end{aligned} \tag{C.3}$$

where $\mathbf{a} \in \mathbb{R}^{n_3}$ is some arbitrary vector.

The following convention is used when writing a derivative involving matrices in this paper for $(\partial\mathbf{A}/\partial\mathbf{B})^\intercal\,\bar{\mathbf{A}}$, where $\mathbf{A} \in \mathbb{R}^{n_1\times n_2}$, and $\mathbf{B} \in \mathbb{R}^{n_2\times n_3}$. Using the vectorization notation, $(\partial\mathbf{A}/\partial\mathbf{B})^\intercal\,\bar{\mathbf{A}}$ is a simplified notation of

$$\operatorname{vec}^{-1}\left(\left(\frac{\partial\operatorname{vec}(\mathbf{A})}{\partial\operatorname{vec}(\mathbf{B})}\right)^\intercal \operatorname{vec}\left(\bar{\mathbf{A}}\right)\right) \in \mathbb{R}^{n_2\times n_3}. \tag{C.4}$$

This signifies the computation of the tensor–vector products such as seen in Eqs. (20) and (25).

Appendix D. RAD form for GMM

Consider the governing equation for SVD as shown in Eq. (8). The vector $\mathbf{u}$ can be written as

$$\mathbf{u} = \frac{\mathbf{A}\mathbf{v}}{\sigma}, \tag{D.1}$$

which can be re-written in terms of the real and imaginary parts as

$$\begin{aligned} \mathbf{u}_r &= \mathbf{A}_r\mathbf{v}_r\frac{1}{\sigma} - \mathbf{A}_i\mathbf{v}_i\frac{1}{\sigma}, \\ \mathbf{u}_i &= \mathbf{A}_r\mathbf{v}_i\frac{1}{\sigma} - \mathbf{A}_i\mathbf{v}_r\frac{1}{\sigma}. \end{aligned} \tag{D.2}$$

Applying the dot product identity from Appendix J and considering the contribution from $\mathbf{A}$, we have

$$\begin{aligned} \bar{\mathbf{A}}_r &= \frac{1}{\sigma}\left(\bar{\mathbf{u}}_r\mathbf{v}_r^\intercal + \bar{\mathbf{u}}_i\mathbf{v}_i^\intercal\right), \\ \bar{\mathbf{A}}_i &= \frac{1}{\sigma}\left(\bar{\mathbf{u}}_i\mathbf{v}_r^\intercal - \bar{\mathbf{u}}_r\mathbf{v}_i^\intercal\right). \end{aligned} \tag{D.3}$$

Consider then the same governing equation for SVD as described in Eq. (8). For the second route taken for the Adjoint-based derivative computation described in Section 3.2, the vector $\mathbf{v}$ is expressed in terms of the remaining singular variables as

$$\mathbf{v} = \frac{\mathbf{A}^*\mathbf{u}}{\sigma}. \tag{D.4}$$

Thus, through trace identities from Appendix J, we have

$$\begin{aligned}\bar{\mathbf{A}}_r &= \frac{1}{\sigma}\left(\mathbf{u}_r\bar{\mathbf{v}}_r^\intercal + \mathbf{u}_i\bar{\mathbf{v}}_i^\intercal\right),\\ \bar{\mathbf{A}}_i &= \frac{1}{\sigma}\left(\mathbf{u}_i\bar{\mathbf{v}}_r^\intercal - \mathbf{u}_r\bar{\mathbf{v}}_i^\intercal\right).\end{aligned} \tag{D.5}$$

This concludes the derivation of for the term in the brackets of Eqs. (20) and (25).

Appendix E. Derivation of Eq. (29)

Consider the residual form $\mathbf{r}(\mathbf{w})$ in Eq. (9). Upon differentiating this with respect to $\mathbf{A}_r$, we have in FAD notation

$$\dot{\mathbf{r}} = \begin{bmatrix}\dot{\mathbf{A}}_r\mathbf{v}_r\\ \dot{\mathbf{A}}_r\mathbf{v}_i\\ \dot{\mathbf{A}}_r{}^\intercal\mathbf{u}_r\\ \dot{\mathbf{A}}_r{}^\intercal\mathbf{u}_i\\ 0\\ 0\end{bmatrix}. \tag{E.1}$$

Applying the dot product identity for real valued functions here, we have

$$\mathrm{Tr}(\bar{\mathbf{r}}^\intercal\dot{\mathbf{r}}) = \mathrm{Tr}(\bar{\mathbf{A}}_r^\intercal\dot{\mathbf{A}}_r). \tag{E.2}$$

Placing the adjoint vector $\boldsymbol{\psi}$ as a reverse seed here, we get

$$\mathrm{Tr}(\bar{\mathbf{r}}^\intercal\dot{\mathbf{r}}) = \mathrm{Tr}(\boldsymbol{\psi}^\intercal\dot{\mathbf{r}}), \tag{E.3}$$

which upon expanding the RHS, we get the expression

$$\mathrm{Tr}(\boldsymbol{\psi}_{v_r}^\intercal\dot{\mathbf{A}}_r\mathbf{v}_r + \boldsymbol{\psi}_{v_i}^\intercal\dot{\mathbf{A}}_r\mathbf{v}_i + \boldsymbol{\psi}_{u_r}^\intercal\dot{\mathbf{A}}_r^\intercal\mathbf{u}_r + \boldsymbol{\psi}_{u_i}^\intercal\dot{\mathbf{A}}_r^\intercal\mathbf{u}_i), \tag{E.4}$$

and then applying the trace identities and re-arranging terms, we get

$$\mathrm{Tr}((\mathbf{v}_r\boldsymbol{\psi}_{v_r}^\intercal + \mathbf{v}_i\boldsymbol{\psi}_{v_i}^\intercal + \boldsymbol{\psi}_{u_r}\mathbf{u}_r^\intercal + \boldsymbol{\psi}_{u_i}\mathbf{u}_i^\intercal)\dot{\mathbf{A}}_r) = \mathrm{Tr}(\bar{\mathbf{A}}_r^\intercal\dot{\mathbf{A}}_r). \tag{E.5}$$

Since this must hold for any arbitrary $\dot{\mathbf{A}}_r$, we have

$$\bar{\mathbf{A}}_r = \boldsymbol{\psi}_{v_r}\mathbf{v}_r^\intercal + \boldsymbol{\psi}_{v_i}\mathbf{v}_i^\intercal + \mathbf{u}_r\boldsymbol{\psi}_{u_r}^\intercal + \mathbf{u}_i\boldsymbol{\psi}_{u_i}^\intercal. \tag{E.6}$$

Similarly taking the derivative of $\mathbf{r}(\mathbf{w})$ in Eq. (9) with respect to $\mathbf{A}_i$ and following a similar procedure as described above, we have

$$\bar{\mathbf{A}}_i = -\boldsymbol{\psi}_{v_r}\mathbf{v}_i^\intercal + \boldsymbol{\psi}_{v_i}\mathbf{v}_r^\intercal + \mathbf{u}_i\boldsymbol{\psi}_{u_r}^\intercal - \mathbf{u}_r\boldsymbol{\psi}_{u_i}^\intercal. \tag{E.7}$$

This concludes our derivation of Eq. (29). It must be noted that the derivative here is $(\partial\mathbf{r}/\partial\mathbf{A})^\intercal\boldsymbol{\psi}$. It is obtained by reverse seeding the residual vector $\mathbf{r}$.

Appendix F. Singular value derivative using RAD

Consider the singular value (dominant) written in terms of the imaginary matrix, $\mathbf{A}$, and its corresponding singular vectors as

$$\sigma = \mathbf{u}^*\mathbf{A}\mathbf{v}, \tag{F.1}$$

where "$\square^*$" represents the imaginary conjugate operation. $\mathbf{u}$ and $\mathbf{v}$ are the left and right singular vectors respectively for the singular value. We will now derive the derivative of this singular value with respect to the matrix $\mathbf{A}$. Eq. (F.1) can be re-written as

$$\begin{aligned}\mathbf{A}\mathbf{v} &= \sigma\mathbf{u},\\ \Rightarrow \dot{\mathbf{A}}\mathbf{v} + \mathbf{A}\dot{\mathbf{v}} &= \dot{\sigma}\mathbf{u} + \sigma\dot{\mathbf{u}},\\ \Rightarrow \mathbf{u}^*\dot{\mathbf{A}}\mathbf{v} + \mathbf{u}^*\mathbf{A}\dot{\mathbf{v}} &= \mathbf{u}^*\dot{\sigma}\mathbf{u} + \sigma\mathbf{u}^*\dot{\mathbf{u}},\\ \Rightarrow \mathbf{u}^*\dot{\mathbf{A}}\mathbf{v} + \mathbf{u}^*\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*\dot{\mathbf{v}} &= \mathbf{u}^*\dot{\sigma}\mathbf{u} + \sigma\mathbf{u}^*\dot{\mathbf{u}},\\ \Rightarrow \mathbf{u}^*\dot{\mathbf{A}}\mathbf{v} + \sigma\mathbf{v}^*\dot{\mathbf{v}} &= \mathbf{u}^*\dot{\sigma}\mathbf{u} + \sigma\mathbf{u}^*\dot{\mathbf{u}},\\ \Rightarrow \mathbf{u}^*\dot{\mathbf{A}}\mathbf{v} &= \dot{\sigma},\end{aligned} \tag{F.2}$$

where going from the second from last to the last equation, we used the fact that $\mathbf{u}^*\mathbf{u} = \mathbf{v}^*\mathbf{v} = 1$, thus, $\dot{\mathbf{u}}^*\mathbf{u} = \dot{\mathbf{v}}^*\mathbf{v} = 0$.

Expanding the final equation in Eq. (F.2) into its real and imaginary parts, we have

$$\begin{aligned}\dot{\sigma} = {} & \mathbf{u}_r^\intercal\dot{\mathbf{A}}_r\mathbf{v}_r + \mathbf{u}_i^\intercal\dot{\mathbf{A}}_i\mathbf{v}_r - \mathbf{u}_r^\intercal\dot{\mathbf{A}}_i\mathbf{v}_i + \mathbf{u}_i^\intercal\dot{\mathbf{A}}_r\mathbf{v}_i\\ & + i(\mathbf{u}_r^\intercal\dot{\mathbf{A}}_i\mathbf{v}_r - \mathbf{u}_i^\intercal\dot{\mathbf{A}}_r\mathbf{v}_r + \mathbf{u}_r^\intercal\dot{\mathbf{A}}_r\mathbf{v}_i + \mathbf{u}_i^\intercal\mathbf{A}_i\mathbf{v}_i),\end{aligned} \tag{F.3}$$

where the imaginary part is zero because the singular value is a real number. Thus,

$$\dot{\sigma} = \mathbf{u}_r^{\mathsf{T}}\dot{\mathbf{A}}_r\mathbf{v}_r + \mathbf{u}_i^{\mathsf{T}}\dot{\mathbf{A}}_i\mathbf{v}_r - \mathbf{u}_r^{\mathsf{T}}\dot{\mathbf{A}}_i\mathbf{v}_i + \mathbf{u}_i^{\mathsf{T}}\dot{\mathbf{A}}_r\mathbf{v}_i, \tag{F.4}$$

which is the final FAD form to begin with for applying the dot product identity formula from Giles [63] as

$$\mathrm{Tr}\left(\overline{\sigma}^{\mathsf{T}}\dot{\sigma}\right) = \mathrm{Tr}\left(\overline{\sigma}^{\mathsf{T}}(\mathbf{u}_r^{\mathsf{T}}\dot{\mathbf{A}}_r\mathbf{v}_r + \mathbf{u}_i^{\mathsf{T}}\dot{\mathbf{A}}_i\mathbf{v}_r - \mathbf{u}_r^{\mathsf{T}}\dot{\mathbf{A}}_i\mathbf{v}_i + \mathbf{u}_i^{\mathsf{T}}\dot{\mathbf{A}}_r\mathbf{v}_i)\right), \tag{F.5}$$

Applying trace identities, we can rewrite it for $\mathbf{A}_r$ as

$$\mathrm{Tr}\left(\overline{\sigma}^{\mathsf{T}}(\mathbf{v}_r\mathbf{u}_r^{\mathsf{T}} + \mathbf{v}_i\mathbf{u}_i^{\mathsf{T}})\dot{\mathbf{A}}_r\right) = \mathrm{Tr}\left(\overline{\mathbf{A}_r}^{\mathsf{T}}\dot{\mathbf{A}}_r\right). \tag{F.6}$$

This has to hold for any arbitrary $\dot{\mathbf{A}}_r$. Thus, we have

$$\overline{\mathbf{A}}_r = (\mathbf{u}_r\mathbf{v}_r^{\mathsf{T}} + \mathbf{u}_i\mathbf{v}_i^{\mathsf{T}})\overline{\sigma}. \tag{F.7}$$

Similarly for $\mathbf{A}_i$, we have:

$$\mathrm{Tr}\left(\overline{\sigma}^{\mathsf{T}}(\mathbf{v}_r\mathbf{u}_i^{\mathsf{T}} - \mathbf{v}_i\mathbf{u}_r^{\mathsf{T}})\dot{\mathbf{A}}_r\right) = \mathrm{Tr}\left(\overline{\mathbf{A}_i}^{\mathsf{T}}\dot{\mathbf{A}}_i\right). \tag{F.8}$$

This has to hold for any arbitrary $\dot{\mathbf{A}}_i$. Thus, we have

$$\overline{\mathbf{A}}_i = (-\mathbf{u}_r\mathbf{v}_i^{\mathsf{T}} + \mathbf{u}_i\mathbf{v}_r^{\mathsf{T}})\overline{\sigma}. \tag{F.9}$$

Placing the reverse seed of $\overline{\sigma}$ equal to 1 in Eq. (F.7) and Eq. (F.9), we can obtain the partial derivatives $\mathrm{d}\sigma_r/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}\sigma_r/\mathrm{d}\mathbf{A}_i$.

In the real dimension, we will have upon placing the reverse seed of $\overline{\sigma}$ equal to 1 in Eq. (F.7),

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}} = \mathbf{u}\mathbf{v}^{\mathsf{T}}, \tag{F.10}$$

where $\mathbf{u} \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}$. This concludes the derivation for Eq. (30).

Appendix G. Adjoint method Jacobian computation and direct contribution of main matrix

Consider the objective function f as

$$f = \mathbf{c}^{\mathsf{T}}\mathbf{u}, \tag{G.1}$$

where $\mathbf{c}$ is a imaginary constant vector and $\mathbf{u}$ is the left singular vector. The function f can be written in its real and imaginary parts as $f = f_r + i f_i$. The real and imaginary parts are

$$\begin{aligned} f_r &= \mathbf{c}_r^{\mathsf{T}}\mathbf{u}_r - \mathbf{c}_i^{\mathsf{T}}\mathbf{u}_i, \\ f_i &= \mathbf{c}_r^{\mathsf{T}}\mathbf{u}_i + \mathbf{c}_i^{\mathsf{T}}\mathbf{u}_r. \end{aligned} \tag{G.2}$$

Here, the Jacobian of the components of f (f_r and f_i) with respect to the state variables $\mathbf{w}$ for the SEMM case is $\partial f/\partial \mathbf{w}$ where $\mathbf{w}$ can be taken from Eq. (9). The Jacobian for f_r will then look like

$$\frac{\partial f_r}{\partial \mathbf{w}} = \begin{bmatrix} \mathbf{c}_r \\ -\mathbf{c}_i \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{G.3}$$

Similarly the Jacobian for f_i will look like

$$\frac{\partial f_i}{\partial \mathbf{w}} = \begin{bmatrix} \mathbf{c}_i \\ \mathbf{c}_r \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{G.4}$$

Once we have these Jacobians, we can proceed with the derivative computation method discussed in Section 4, particularly from Eq. (27) since we now have its RHS. A similar approach can be shown for Eqs. (16) and (21) in the GMM case for formation of the Jacobian and hence the RHS of the adjoint equations. It can be seen however that this is simply cumbersome even for such a simple linear function as shown in Eq. (G.1).

Thus, JAX [64] was used for this step, and so can any other equivalent automatic differentiation tool be used to save implementation effort and time. The same approach can be taken for the direct dependencies of the function on the matrix $\mathbf{A}$, if any. If the dependency arises from the GMM computation, then the formulae shown in Appendix D can be used, as pointed out in Section 3.2. These formulae are based on the automatic differentiation and hence are equivalent when either this approach or the JAX approach is used for evaluation of this step.

Appendix H. FD formula

A finite difference formula (FD) was employed to compare the results from the proposed Adjoint and RAD methods for the singular value derivative in Sections 4.1 and 4.2. The following equations were employed for the FD results in Tables 2–5.

$$\begin{aligned}
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_{\mathrm{r,pq}}} &= \mathrm{Re}\left(\frac{f(\mathbf{A}+\epsilon\mathbf{E}_{\mathrm{pq}})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_{\mathrm{i,pq}}} &= \mathrm{Im}\left(\frac{f(\mathbf{A}+i\epsilon\mathbf{E}_{\mathrm{pq}})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_{\mathrm{r,pq}}} &= \mathrm{Re}\left(\frac{f(\mathbf{A}+\epsilon\mathbf{E}_{\mathrm{pq}})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_{\mathrm{i,pq}}} &= \mathrm{Im}\left(\frac{f(\mathbf{A}+i\epsilon\mathbf{E}_{\mathrm{pq}})-f(\mathbf{A})}{\epsilon}\right),
\end{aligned} \tag{H.1}$$

where $\epsilon = 10^{-6}$ is the finite perturbation applied to the matrix $\mathbf{A}$, $\mathbf{E}_{\mathrm{pq}}$ is a single entry matrix with the element on pth row and qth column set equal to ϵ and the remaining elements set to zero.

Appendix I. The method of snapshots

In the method of snapshots, a covariance matrix is formed using the equation

$$\mathbf{C} = \mathbf{X}^{\intercal}\mathbf{X}, \tag{I.1}$$

where $\mathbf{X}$ is the snapshot data matrix. The EVP of $\mathbf{X}$ is then solved to get the eigenvectors and eigenvalues

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i. \tag{I.2}$$

The POD modes are then simply

$$\mathbf{\Phi}_i = \mathbf{X}\mathbf{v}_i \frac{1}{\sqrt{\lambda_i}}, \tag{I.3}$$

where i denotes each mode level. The energy levels are related to λ, and the singular values from the SVD are simply the square roots of all the λ_i.

The eigenvectors $\mathbf{v}_i$ are the same as the right singular vectors of SVD of $\mathbf{X}$ and are related to the temporal coefficients. The temporal coefficients can be calculated as

$$a_i(t) = \mathbf{v}_i\sqrt{\lambda_i}, \tag{I.4}$$

where t is the physical time step value.

Appendix J. RAD dot product identities and chain rule

We propose the dot product identity for imaginary differentiable function (in multivariate sense). For real valued matrices, $\mathbf{A} \in \mathbb{R}^{m_1\times n_1}$, and functions of those matrices, $\mathbf{B}(\mathbf{A}) : \mathbb{R}^{m_1\times n_1} \to \mathbb{R}^{m_2\times n_2}$, Giles [63] proposed the following dot product identity

$$\mathrm{Tr}\left(\overline{\mathbf{B}}^{\intercal}\dot{\mathbf{B}}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{\intercal}\dot{\mathbf{A}}\right), \tag{J.1}$$

where $\mathbf{B}$ is a real differentiable function of $\mathbf{A}$.

He et al. [44] proposed the following identity for imaginary matrices, $\mathbf{A} \in \mathbb{C}^{m_1\times n_1}$, and analytic matrix functions of those matrices, $\mathbf{B}(\mathbf{A}) : \mathbb{C}^{m_1\times n_1} \to \mathbb{C}^{m_2\times n_2}$

$$\mathrm{Tr}\left(\overline{\mathbf{B}}^{*}\dot{\mathbf{B}}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{*}\dot{\mathbf{A}}\right), \tag{J.2}$$

where the transpose operator, $\square^{\intercal}$, is replaced by the conjugate transpose operator, $\square^{*}$.

However, for imaginary differentiable function (in multivariate sense) but not imaginary non-analytic functions, the conjugate variables need to be considered. After applying the Wiritinger derivatives, Roberts and Roberts proposed the following RAD trace-identity for imaginary matrices and their non-analytic functions:

$$\mathrm{Tr}\left(\overline{\mathbf{B}}^{*}\dot{\mathbf{B}} + (\overline{\mathbf{B}}^{*}\dot{\mathbf{B}})^{*}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{*}\dot{\mathbf{A}} + (\overline{\mathbf{A}}^{*}\dot{\mathbf{A}})^{*}\right). \tag{J.3}$$

Notice that for $\mathbf{B}$ being an analytic function of $\mathbf{A}$, the second term in the Trace operator on each side of Eq. (J.3) becomes zero, effectively reducing to Eq. (J.2).

We can also expand Eq. (J.3) into real and imaginary components which is more convenient to deal with in many cases where the real and imaginary components are derived separately. Eq. (J.3) can be written as

$$\mathrm{Tr}\left(\overline{\mathbf{B}}_r^{\intercal}\dot{\mathbf{B}}_r + \overline{\mathbf{B}}_i^{\intercal}\dot{\mathbf{B}}_i\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}_r^{\intercal}\dot{\mathbf{A}}_r + \overline{\mathbf{A}}_i^{\intercal}\dot{\mathbf{A}}_i\right), \tag{J.4}$$

where we used the trace identity that $\mathrm{Tr}\,(\mathbf{A}) = \mathrm{Tr}\,(\mathbf{A}^{\intercal})$ (see Table J.6).

Table J.6
RAD trace identity for different matrix functions.

	Formula
Real differentiable	$\mathrm{Tr}\left(\overline{\mathbf{B}}^{\mathsf{T}}\dot{\mathbf{B}}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{\mathsf{T}}\dot{\mathbf{A}}\right)$ (Giles [63])
Imaginary analytic	$\mathrm{Tr}\left(\overline{\mathbf{B}}^{*}\dot{\mathbf{B}}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{*}\dot{\mathbf{A}}\right)$ (He et al. [44])
Imaginary multivariate differentiable	$\mathrm{Tr}\left(\overline{\mathbf{B}}^{*}\dot{\mathbf{B}} + (\overline{\mathbf{B}}^{*}\dot{\mathbf{B}})^{*}\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}^{*}\dot{\mathbf{A}} + (\overline{\mathbf{A}}^{*}\dot{\mathbf{A}})^{*}\right)$ (Roberts and Roberts [74]) $\mathrm{Tr}\left(\overline{\mathbf{B}}_r^{\mathsf{T}}\dot{\mathbf{B}}_r + \overline{\mathbf{B}}_i^{\mathsf{T}}\dot{\mathbf{B}}_i\right) = \mathrm{Tr}\left(\overline{\mathbf{A}}_r^{\mathsf{T}}\dot{\mathbf{A}}_r + \overline{\mathbf{A}}_i^{\mathsf{T}}\dot{\mathbf{A}}_i\right)$ (current paper)

Appendix K. RAD formulae in the literature

Giles [63] proposed the following RAD formula for singular value derivative

$$\overline{\mathbf{A}} = \mathbf{U}\overline{\mathbf{S}}\mathbf{V}^{\mathsf{T}}. \tag{K.1}$$

This formula works with real-valued inputs and is only for the singular value derivative. For the singular vector, an approach was shown in [75] but no concrete implementable formula was proposed. Townsend [58] proposed the following RAD formula for singular variables derivative

$$\begin{aligned}\overline{\mathbf{A}} &= [\mathbf{U}(\mathbf{F}\circ[\mathbf{U}^{\mathsf{T}}\overline{\mathbf{U}} - \overline{\mathbf{U}}^{\mathsf{T}}\mathbf{U}])\mathbf{S} + (\mathbf{I}_m - \mathbf{U}\mathbf{U}^{\mathsf{T}})\overline{\mathbf{U}}\mathbf{S}^{-1}]\mathbf{V}^{\mathsf{T}} \\ &+ \mathbf{U}(\mathbf{I}_k\circ\overline{\mathbf{S}})\mathbf{V}^{\mathsf{T}} + \mathbf{U}[\mathbf{S}(\mathbf{F}\circ[\mathbf{V}^{\mathsf{T}}\overline{\mathbf{V}} - \overline{\mathbf{V}}^{\mathsf{T}}\mathbf{V}])\mathbf{V}^{\mathsf{T}} + \mathbf{S}^{-1}\overline{\mathbf{V}}^{\mathsf{T}}(\mathbf{I}_n - \mathbf{V}\mathbf{V}^{\mathsf{T}})],\end{aligned} \tag{K.2}$$

where $\mathbf{A}$ is an $m\times n$ matrix of rank $k \le \min(m,n)$, $\mathbf{U}$ is $m\times k$, $\mathbf{S}$ is $k\times k$, $\mathbf{V}$ is $n\times k$, $\mathbf{U}^{\mathsf{T}}\mathbf{U} = \mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{I}_k$ (the identity matrix of dimension k), and $\mathbf{F}$ is defined as

$$\mathbf{F}_{ij} = \begin{cases} \frac{1}{s_j^2 - s_i^2} & \text{if } i \ne j, \\ 0 & \text{if } i = j, \end{cases} \tag{K.3}$$

where s is each entry of the diagonal of singular values matrix $\mathbf{S}$. Also, $\circ$ denotes the Hadamard product. This formula works with truncated SVD and also with only real-valued inputs and outputs. Wan and Zhang [60] proposed the following RAD formula

$$\begin{aligned}\overline{\mathbf{A}} = \frac{1}{2}\Bigg(& 2\mathbf{U}\overline{\mathbf{S}}\mathbf{V}^{*} + \mathbf{U}(\mathbf{J}+\mathbf{J}^{*})\mathbf{S}\mathbf{V}^{*} + \mathbf{U}\mathbf{S}(\mathbf{K}+\mathbf{K}^{*})\mathbf{V}^{*} + \frac{1}{2}\mathbf{U}\mathbf{S}^{-1}(\mathbf{L}^{*}-\mathbf{L})\mathbf{V}^{*} \\ &+2(\mathbf{I}-\mathbf{U}\mathbf{U}^{*})\overline{\mathbf{U}}\mathbf{S}^{-1}\mathbf{V}^{*} + 2\mathbf{U}\mathbf{S}^{-1}\overline{\mathbf{V}}^{*}(\mathbf{I}-\mathbf{V}\mathbf{V}^{*})\Bigg),\end{aligned} \tag{K.4}$$

where $\mathbf{J} = \mathbf{F}\circ(2\mathbf{U}^{*}\overline{\mathbf{U}})$, $\mathbf{K} = \mathbf{F}\circ(2\mathbf{V}^{*}\overline{\mathbf{V}})$, $\mathbf{L} = \mathbf{I}\circ(2\mathbf{V}^{*}\overline{\mathbf{V}})$, $\mathbf{F}$ is same as in Eq. (K.3), $\mathbf{I}$ is the identity matrix. The formula proposed works with imaginary valued inputs and gives one single derivative value as opposed to the imaginary components shown in te current manuscript (such as $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$). The details on this Wiritinger derivative are shown in Appendix L.

Seeger et al. [59] proposed the following RAD formula for singular variables derivative

$$\begin{aligned}\overline{\mathbf{A}} &= \mathbf{U}^{\mathsf{T}}\left(\mathbf{G}_2\mathbf{V} + \Lambda^{-1}\overline{\mathbf{V}}\right), \\ \mathbf{G}_2 &= \mathbf{G}_1\circ\mathbf{E}\Lambda - \left(\Lambda^{-1}\overline{\mathbf{V}}\mathbf{V}^{\mathsf{T}}\circ\mathbf{I}\right), \\ \mathbf{G}_1 &= \overline{\mathbf{U}}\mathbf{U}^{\mathsf{T}} + \Lambda^{-1}\overline{\mathbf{V}}\mathbf{V}^{\mathsf{T}}\mathbf{A}, \\ \mathbf{E}_{i,j} &= \begin{cases} \frac{1}{\lambda_j - \lambda_i}, & i \ne j, \\ 0, & i = j, \end{cases} \qquad h(t) = \max(|t|,\epsilon)\,\mathrm{sgn}(t),\end{aligned} \tag{K.5}$$

where the singular value decomposition (SVD) in Seeger et al. [59] is defined as

$$\mathbf{A} = \mathbf{U}^{\mathsf{T}}\Lambda\mathbf{V}, \quad \Lambda = \mathrm{diag}(\lambda) \in \mathbb{R}^{m\times m}, \quad \mathbf{U} \in \mathbb{R}^{m\times m}, \quad \mathbf{V} \in \mathbb{R}^{m\times n}, \tag{K.6}$$

and further details on the computation of thin SVD can be found in Seeger et al. [59]. The formula in Eq. (K.5) works with real valued inputs.

Appendix L. Wiritinger derivative

In the current manuscript just as in He et al. [44], the derivatives in the imaginary domain are expressed in terms of their real and imaginary components as $\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$ and such. It may therefore be useful to the readers to see one single derivative $\mathrm{d}f/\mathrm{d}\mathbf{A}$ for the imaginary valued case. This can be achieved by using the Wiritinger derivative. Information on this derivative can be found in the book by Hjorunges [76]. The derivative is

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{A}} = \frac{1}{2}\left(\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_r} + i\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_r} - i\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_i} + \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_i}\right), \tag{L.1}$$

where i is iota, the imaginary number $\sqrt{-1}$.

For instance, take the singular value derivative as shown in Eqs. (F.7) and (F.9). These go into the Wiritinger derivative as shown below

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}} = \frac{1}{2}\left[\frac{\partial\sigma_r}{\partial\mathbf{A}_r} - i\frac{\partial\sigma_r}{\partial\mathbf{A}_i}\right]. \tag{L.2}$$

The imaginary terms for the singular value have been removed in Eq. (L.2) because the singular value is purely real. Before we proceed, consider the following equation:

$$\mathbf{u}\mathbf{v}^* = (\mathbf{u}_r + i\mathbf{u}_i)(\mathbf{v}_r^{\mathrm{T}} - i\mathbf{v}_i^{\mathrm{T}}) = (\mathbf{u}_r\mathbf{v}_r^{\mathrm{T}} + \mathbf{u}_i\mathbf{v}_i^{\mathrm{T}}) + i(\mathbf{u}_i\mathbf{v}_r^{\mathrm{T}} - \mathbf{u}_r\mathbf{v}_i^{\mathrm{T}}). \tag{L.3}$$

We will then use Eq. (F.7) and Eq. (F.9) to expand Eq. (L.2). After doing so, we obtain:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}} = \frac{1}{2}\left[(\mathbf{u}_r\mathbf{v}_r^{\mathrm{T}} + \mathbf{u}_i\mathbf{v}_i^{\mathrm{T}}) - i(\mathbf{u}_i\mathbf{v}_r^{\mathrm{T}} - \mathbf{u}_r\mathbf{v}_i^{\mathrm{T}})\right]. \tag{L.4}$$

Upon careful observation, one can notice that the term square in brackets of Eq. (L.4) is simply the imaginary conjugate of Eq. (L.3). Thus, we have

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\mathbf{A}} = \frac{1}{2}(\mathbf{u}\mathbf{v}^*)^c, \tag{L.5}$$

where c denotes imaginary conjugate operation. In this way, we can compute the derivative of any one particular singular value with respect to its imaginary matrix $\mathbf{A}$. The formula however must not be confused with the formula in the real case. This is because the Wiritinger derivative is only applied in the imaginary plane treating the imaginary variable and its imaginary conjugate variable as two independent variables [77]. Similarly, we can compute the total derivative from the imaginary parts in Eqs. (20), (25) and (28).

Appendix M. Truss and finite element equations

The governing equations for static response design sensitivity analysis is

$$\mathbf{K}\mathbf{d} = \mathbf{f}, \tag{M.1}$$

where $\mathbf{K} \in \mathbb{R}^{n_b \times n_b}$ is the global stiffness matrix, $\mathbf{f} \in \mathbb{R}^{n_b}$ is the vector of applied forces, $\mathbf{d}$ is the vector of displacements in the static condition and n_b is the number of elements/bars in the truss.

Next, we must discuss the finite element equations for Eq. (M.1). Consider a 3D truss element with nodes i and j, having coordinates (x_i, y_i, z_i) and (x_j, y_j, z_j) respectively. The element has cross-sectional area b, Young's modulus E and length L.

The direction cosines of the element are

$$l = \frac{x_j - x_i}{L}, \quad m = \frac{y_j - y_i}{L}, \quad n = \frac{z_j - z_i}{L}. \tag{M.2}$$

The local stiffness matrix in global coordinates (6 × 6) is

$$\mathbf{K}_{\text{local}}^{(e)} = \frac{bE}{L}\begin{bmatrix} l^2 & lm & ln & -l^2 & -lm & -ln \\ lm & m^2 & mn & -lm & -m^2 & -mn \\ ln & mn & n^2 & -ln & -mn & -n^2 \\ -l^2 & -lm & -ln & l^2 & lm & ln \\ -lm & -m^2 & -mn & lm & m^2 & mn \\ -ln & -mn & -n^2 & ln & mn & n^2 \end{bmatrix}, \tag{M.3}$$

where bE/L is basically the stiffness of the bar. The global stiffness matrix $\mathbf{K}$ is assembled by adding contributions of all element stiffness matrices $\mathbf{K}_{\text{local}}^{(e)}$ to their appropriate global degrees of freedom (DOFs).

For each element, we map the local DOFs to global ones. Suppose an element connects nodes i and j, the global DOF vector is:

$$\mathbf{u}^{(e)} = [u_{i_x}, u_{i_y}, u_{i_z}, u_{j_x}, u_{j_y}, u_{j_z}]^T. \tag{M.4}$$

Next, we assemble each element's stiffness matrix into the global matrix using these DOF indices

$$\mathbf{K} = \sum_{e=1}^{n_{\text{elem}}} \mathbf{A}_e^{\mathrm{T}}\mathbf{K}_{\text{local}}^{(e)}\mathbf{A}_e, \tag{M.5}$$

where $\mathbf{A}_e$ is a Boolean assembly matrix that maps local to global DOFs. In the finite element method (FEM), a "Boolean assembly matrix" is used to map local DOFs from individual elements to the global system. It is called "Boolean" because its entries are either 0 or 1, indicating whether a particular local DOF contributes to a global DOF.

Consider a system with a global DOF and an element with b local DOFs. When assembling the global stiffness matrix $\mathbf{K}$ from local element matrices $\mathbf{K}_{\text{local}}^{(e)}$, the Boolean matrix is used as in Eq. (M.5) where each row of $\mathbf{A}_e$ has a single entry of 1 corresponding to the global index of the respective local DOF. This facilitates correct placement of local element contributions into the global system matrix or vector.

Application of boundary conditions involves modifying $\mathbf{K}$ and $\mathbf{f}$ to enforce prescribed displacements, zeroing rows and columns of fixed DOFs and inserting 1 along the diagonal, and setting corresponding entries in $\mathbf{f}$ to the prescribed values. The reduction for $\mathbf{f}$ is done by identifying an index set of free DOFs and extracting the subvector $\mathbf{f}_{\text{red}}$, which is the reduced vector of $\mathbf{f}$.

Finally, we solve the modified system

$$\mathbf{K}_{\text{red}}\mathbf{d}_{\text{red}} = \mathbf{f}_{\text{red}}, \tag{M.6}$$

where $\mathbf{K}_{\text{red}}$ is the reduced stiffness matrix, $\mathbf{d}_{\text{red}}$ is the vector of displacements that with only the nodes we solve for and $\mathbf{f}_{\text{red}}$ is the reduced vector of forces applied to only the nodes under consideration. We refer the readers to Chandrupatla et al. [[78],Chapter 4] which throws light on the three-dimensional finite element formulation for trusses.

It is noted at this stage that we perform SVD on the inverse this reduced stiffness matrix as

$$\mathbf{G} = \mathbf{K}_{\text{red}}^{-1}, \tag{M.7}$$

where $\mathbf{G}$ is the transformation matrix.

Once $\mathbf{d}_{\text{mod}}$ is obtained, element forces can be calculated using:

$$f^{(e)} = \frac{bE}{L}\begin{bmatrix}-l & -m & -n & l & m & n\end{bmatrix}\mathbf{d}_{\text{red}}^{(e)}, \tag{M.8}$$

where f is the internal force for the element e.

For the response of the system, SVD of the matrix $\mathbf{G}$ is sought as shown in Eq. (M.7) which gives the singular variables subject to design conditions in $\mathbf{b}$. In the study of Ersoy and Mugan [29], the design parameter b_1 was found to be the most effective parameter in order to reduce displacements, or in other words, increase the stiffness of the structure. In the following part of this section, we formulate RAD-based equations which includes the RAD form for the singular value sensitivity from Eq. (31).

The path of perturbations is first studied. The design variables affect the stiffness matrix $\mathbf{K}_{\text{red}}$. $\mathbf{K}_{\text{red}}$ in turn affects $\mathbf{G}$, the transfer function matrix. The SVD of $\mathbf{G}$ gives the first singular value σ_1, whose derivative was found to be crucial in minimization of the total static response magnitude [29].

The derivative of the first singular value with respect to the matrix $\mathbf{G}$ is

$$\frac{\mathrm{d}\sigma_1}{\mathrm{d}\mathbf{G}} = \mathbf{u}_1\mathbf{v}_1^{\mathsf{T}}, \tag{M.9}$$

where $\mathbf{u}, \mathbf{v}$ is the first pair of singular vectors corresponding to σ_1. This was derived earlier in Appendix F in Eq. (F.10). The derivative is given in Eq. (31). Next, the RAD form for Eq. (M.7) is given by

$$\overline{\mathbf{K}}_{\text{red}} = \mathbf{K}_{\text{red}}^{-\mathsf{T}}\overline{\mathbf{G}}\mathbf{K}_{\text{red}}^{-\mathsf{T}}, \tag{M.10}$$

in which upon seeding σ_1 as a reverse seed, we get

$$\frac{\mathrm{d}\sigma_1}{\mathrm{d}\mathbf{K}_{\text{red}}} = \mathbf{K}_{\text{red}}^{-\mathsf{T}}\mathbf{u}_1\mathbf{v}_1^{\mathsf{T}}\mathbf{K}_{\text{red}}^{-\mathsf{T}}, \tag{M.11}$$

which is the derivative we require before we flatten the derivative matrix in LHS into a vector. For the derivation of the RAD form for inverse of a real-valued matrix, we refer the readers to Giles [75].

The final step is to obtain $\mathrm{d}\mathbf{K}_{\text{red}}/\mathrm{d}\mathbf{b}$. It is evident from Eq. (M.3) that the derivative of $\mathrm{d}\mathbf{K}_{\text{red}}/\mathrm{d}\mathbf{b}$ is a tensor. $\mathrm{d}\mathbf{K}_{\text{red}}/\mathrm{d}b_1$ for instance is simply

$$\frac{\mathrm{d}\mathbf{K}_{\text{red}}^{(e)}}{\mathrm{d}b_1} = \frac{E}{L}\begin{bmatrix} l^2 & lm & ln & -l^2 & -lm & -ln \\ lm & m^2 & mn & -lm & -m^2 & -mn \\ ln & mn & n^2 & -ln & -mn & -n^2 \\ -l^2 & -lm & -ln & l^2 & lm & ln \\ -lm & -m^2 & -mn & lm & m^2 & mn \\ -ln & -mn & -n^2 & ln & mn & n^2 \end{bmatrix}, \tag{M.12}$$

which is simply a matrix of all constants with $e = 1$. This matrix does not change for the other areas in $\mathbf{b}$. Each matrix in this tensor is then flattened into a vector. This tensor is then simply plugged into Eq. (50) to complete the derivative formula for $\mathrm{d}\sigma_1/\mathrm{d}\mathbf{b}$.

Data availability

Data will be made available on request.

References

[1] G. Strang, Introduction to Linear Algebra, sixth ed., Wellesley-Cambridge Press, Philadelphia, PA, 2022, http://dx.doi.org/10.1137/1.9781733146678.

[2] C. Eckart, G. Young, A principal axis transformation for non-Hermitian matrices, 1939.

[3] E. Candès, B. Recht, Exact matrix completion via convex optimization, Commun. ACM 55 (6) (2012) 111–119, http://dx.doi.org/10.1145/2184319.2184343.

[4] K.-W. Gwak, G.Y. Masada, Structural analysis and optimization of nonlinear control systems using singular value decomposition, J. Dyn. Syst. Meas. Control. 127 (1) (2004) 105–113, http://dx.doi.org/10.1115/1.1876495.

[5] M. Khalil, S. Adhikari, A. Sarkar, Linear system identification using proper orthogonal decomposition, Mech. Syst. Signal Process. 21 (8) (2007) 3123–3145, http://dx.doi.org/10.1016/j.ymssp.2007.03.007.
[6] S. Sarkar, A. Dong, J.S. Gero, Design optimization problem Reformulation using singular value decomposition, J. Mech. Des. 131 (8) (2009) http://dx.doi.org/10.1115/1.3179148.
[7] S.d. Lucas Bodas, An Optimization Method for Conceptual Design in Engineering Systems Based on High Order Singular Value Decomposition (Ph.D. thesis), Universidad Politecnica de Madrid - University Library, 2012, http://dx.doi.org/10.20868/upm.thesis.13202.
[8] X. Zhao, B. Ye, Singular value decomposition packet and its application to extraction of weak fault feature, Mech. Syst. Signal Process. 70–71 (2016) 73–86, http://dx.doi.org/10.1016/j.ymssp.2015.08.033.
[9] J. Li, S. He, J.R.R.A. Martins, M. Zhang, B.C. Khoo, Efficient data-driven off-design constraint modeling for practical Aerodynamic shape optimization, AIAA J. 61 (7) (2023) 2854–2866, http://dx.doi.org/10.2514/1.J062629.
[10] P. Díaz-Morales, A. Corrochano, M. López-Martín, S. Le Clainche, Deep learning combined with singular value decomposition to reconstruct databases in fluid dynamics, Expert Syst. Appl. 238 (2024) 121924, http://dx.doi.org/10.1016/j.eswa.2023.121924.
[11] K. Shin, S. Feraday, C. Harris, M. Brennan, J.-E. Oh, Optimal autoregressive modelling of a measured noisy deterministic signal using singular-value decomposition, Mech. Syst. Signal Process. 17 (2) (2003) 423–432, http://dx.doi.org/10.1006/mssp.2002.1510.
[12] H. Pan, Y. Yang, J. Zheng, J. Cheng, A noise reduction method of symplectic singular mode decomposition based on Lagrange multiplier, Mech. Syst. Signal Process. 133 (2019) 106283, http://dx.doi.org/10.1016/j.ymssp.2019.106283.
[13] J. Li, Z. Chen, S. Li, Selection of the number of effective singular values for noise reduction, Mech. Syst. Signal Process. 191 (2023) 110175, http://dx.doi.org/10.1016/j.ymssp.2023.110175.
[14] R.A. Sadek, SVD based image processing applications: State of the art, contributions and research challenges, 2012, http://dx.doi.org/10.48550/ARXIV.1211.7102, arXiv.
[15] D. Wang, Adjustable robust singular value decomposition: Design, analysis and application to finance, Data 2 (3) (2017) 29, http://dx.doi.org/10.3390/data2030029.
[16] S.d. Lucas, J.M. Vega, A. Velazquez, Aeronautic conceptual design optimization method based on high-order singular value decomposition, AIAA J. 49 (12) (2011) 2713–2725, http://dx.doi.org/10.2514/1.j051133.
[17] Y. Choi, G. Boncoraglio, S. Anderson, D. Amsallem, C. Farhat, Gradient-based constrained optimization using a database of linear reduced-order models, J. Comput. Phys. 423 (2020) 109787, http://dx.doi.org/10.1016/j.jcp.2020.109787.
[18] D.J. Poole, C.B. Allen, T. Rendall, Efficient aero-structural wing optimization using compact Aerofoil decomposition, in: AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics, 2019, http://dx.doi.org/10.2514/6.2019-1701.
[19] J. Li, M.A. Bouhlel, J.R.R.A. Martins, A data-based approach for fast airfoil analysis and optimization, in: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 2018, http://dx.doi.org/10.2514/6.2018-1383.
[20] S. Li, J. Trevelyan, Z. Wu, H. Lian, D. Wang, W. Zhang, An adaptive SVD–krylov reduced order model for surrogate based structural shape optimization through isogeometric boundary element method, Comput. Methods Appl. Mech. Engrg. 349 (2019) 312–338, http://dx.doi.org/10.1016/j.cma.2019.02.023.
[21] W. Yao, S. Marques, T. Robinson, C. Armstrong, L. Sun, A reduced-order model for gradient-based aerodynamic shape optimisation, Aerosp. Sci. Technol. 106 (2020) 106120, http://dx.doi.org/10.1016/j.ast.2020.106120.
[22] J. Li, M. Zhang, Data-based approach for wing shape design optimization, Aerosp. Sci. Technol. 112 (2021) 106639, http://dx.doi.org/10.1016/j.ast.2021.106639.
[23] N. Wu, C.A. Mader, J.R.R.A. Martins, Sensitivity-based geometric parametrization and automatic scaling for Aerodynamic Shape optimization, AIAA J. 62 (1) (2023) 231–246, http://dx.doi.org/10.2514/1.J062661.
[24] W. Chen, J. Kou, W. Yang, S. Pan, Dynamic-mode-decomposition-based gradient prediction for adjoint-based aerodynamic shape optimization, Aerosp. Sci. Technol. 150 (2024) 109175, http://dx.doi.org/10.1016/j.ast.2024.109175.
[25] B. Liu, H. Li, F. Liu, W. Huang, G. Wang, H. Wang, W. Wang, Y. Tang, H. Liu, Beat vibration analysis model of overflow dam pier under discharge excitation based on adaptive VMD and automatic operational modal analysis, Mech. Syst. Signal Process. 211 (2024) 111201, http://dx.doi.org/10.1016/j.ymssp.2024.111201.
[26] H. Li, B. Liu, W. Huang, H. Liu, G. Wang, Vibration load identification in the time-domain of high arch dam under discharge excitation based on hybrid LSQR algorithm, Mech. Syst. Signal Process. 177 (2022) 109193, http://dx.doi.org/10.1016/j.ymssp.2022.109193.
[27] K. Liu, Application of SVD in optimization of structural modal test, Comput. Struct. 63 (1) (1997) 51–59, http://dx.doi.org/10.1016/s0045-7949(96)00329-x.
[28] T.H. Lee, An adjoint variable method for structural design sensitivity analysis of a distinct eigenvalue problem, KSME Int. J. 13 (6) (1999) 470–476, http://dx.doi.org/10.1007/bf02947716.
[29] H. Ersoy, A. Muğan, Design sensitivity analysis of structures based upon the singular value decomposition, Comput. Methods Appl. Mech. Engrg. 191 (32) (2002) 3459–3476, http://dx.doi.org/10.1016/s0045-7825(02)00259-1.
[30] H. Ersoy, Optimum laminate design by using singular value decomposition, Compos. Part B: Eng. 52 (2013) 144–154, http://dx.doi.org/10.1016/j.compositesb.2013.04.005.
[31] L.N. Trefethen, A.E. Trefethen, S.C. Reddy, T.A. Driscoll, Hydrodynamic stability without eigenvalues, Science 261 (5121) (1993) 578–584, http://dx.doi.org/10.1126/science.261.5121.578.
[32] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28, http://dx.doi.org/10.1017/s0022112010001217.
[33] K. Taira, M.S. Hemati, S.L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S.T.M. Dawson, C.-A. Yeh, Modal analysis of fluid flows: Applications and outlook, AIAA J. (2019) 1–25, http://dx.doi.org/10.2514/1.j058462.
[34] K. Taira, M.S. Hemati, S.L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S.T.M. Dawson, C.-A. Yeh, Modal analysis of fluid flows: Applications and outlook, AIAA J. 58 (3) (2020) 998–1022, http://dx.doi.org/10.2514/1.j058462.
[35] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[36] L.N. Trefethen, D. Bau III, Numerical Linear Algebra, SIAM, Society for Industrial and Applied Mathematics, 1997.
[37] J. Demmel, K. Veselić, Jacobi's method is more accurate than QR, SIAM J. Matrix Anal. Appl. 13 (4) (1992) 1204–1245, http://dx.doi.org/10.1137/0613074.
[38] Y. Saad, Numerical Methods for Large Eigenvalue Problems: Revised Edition, Society for Industrial and Applied Mathematics, 2011, http://dx.doi.org/10.1137/1.9781611970739.
[39] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288, http://dx.doi.org/10.1137/090771806.
[40] Z. Zhang, The singular value decomposition, applications and beyond, 2015, http://dx.doi.org/10.48550/ARXIV.1510.08532.
[41] A.J. Salgado, S.M. Wise, Classical Numerical Analysis: A Comprehensive Course, Cambridge University Press, 2022, http://dx.doi.org/10.1017/9781108942607.
[42] H. Schwerdtfeger, Introduction to linear algebra and the theory of matrices, 1961.
[43] O. Makkonen, C. Hollanti, Secure distributed Gram Matrix multiplication, in: 2023 IEEE Information Theory Workshop, ITW, IEEE, 2023, http://dx.doi.org/10.1109/itw55543.2023.10161614.
[44] S. He, Y. Shi, E. Jonsson, J.R.R.A. Martins, Eigenvalue problem derivatives computation for a complex matrix using the adjoint method, Mech. Syst. Signal Process. 185 (2023) 109717, http://dx.doi.org/10.1016/j.ymssp.2022.109717.

[45] S. Ragnarsson, C.F. Van Loan, Block tensors and symmetric embeddings, Linear Algebra Appl. 438 (2) (2013) 853–874, http://dx.doi.org/10.1016/j.laa.2011.04.014.
[46] P.C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion, Society for Industrial and Applied Mathematics, 1998, http://dx.doi.org/10.1137/1.9780898719697.
[47] G.W. Stewart, Perturbation theory for the singular value decomposition, 1998, Citeseer.
[48] V. Angelova, P. Petkov, Componentwise perturbation analysis of the singular value decomposition of a Matrix, Appl. Sci. 14 (4) (2024) 1417, http://dx.doi.org/10.3390/app14041417.
[49] P.I. Davies, M.I. Smith, Updating the singular value decomposition, J. Comput. Appl. Math. 170 (1) (2004) 145–167, http://dx.doi.org/10.1016/j.cam.2003.12.039.
[50] R.M. Lin, J.E. Mottershead, T.Y. Ng, A state-of-the-art review on theory and engineering applications of eigenvalue and eigenvector derivatives, Mech. Syst. Signal Process. 138 (2020) 106536, http://dx.doi.org/10.1016/j.ymssp.2019.106536.
[51] X. García Santiago, Numerical Methods for Shape Optimization of Photonic Nanostructures, Karlsruher Institut für Technologie (KIT), 2021, http://dx.doi.org/10.5445/IR/1000131006.
[52] C.S. Skene, P.J. Schmid, Adjoint-based parametric sensitivity analysis for swirling M-flames, J. Fluid Mech. 859 (2018) 516–542, http://dx.doi.org/10.1017/jfm.2018.793.
[53] J.R.R.A. Martins, A. Ning, Engineering Design Optimization, Cambridge University Press, Cambridge, UK, 2022, http://dx.doi.org/10.1017/9781108980647, URL https://mdobook.github.io.
[54] J.R.R.A. Martins, J.T. Hwang, Review and unification of methods for computing derivatives of multidisciplinary computational models, AIAA J. 51 (11) (2013) 2582–2599, http://dx.doi.org/10.2514/1.J052184.
[55] J.R.R.A. Martins, P. Sturdza, J.J. Alonso, The complex-step derivative approximation, ACM Trans. Math. Software 29 (3) (2003) 245–262, http://dx.doi.org/10.1145/838250.838251.
[56] Z. Lyu, Z. Xu, J.R.R.A. Martins, Benchmarking optimization algorithms for wing Aerodynamic design optimization, in: Proceedings of the 8th International Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China, 2014, ICCFD8-2014-0203.
[57] M.B. Giles, N.A. Pierce, An introduction to the adjoint approach to design, Flow, Turbul. Combust. 65 (2000) 393–415, http://dx.doi.org/10.1023/A:1011430410075.
[58] J. Townsend, Differentiating the Singular Value Decomposition, Technical Report 2016, 2016.
[59] M. Seeger, A. Hetzel, Z. Dai, E. Meissner, N.D. Lawrence, Auto-differentiating linear Algebra, 2017, http://dx.doi.org/10.48550/ARXIV.1710.08717, arXiv.
[60] Z.-Q. Wan, S.-X. Zhang, Automatic differentiation for complex valued SVD, 2019, http://dx.doi.org/10.48550/ARXIV.1909.02659.
[61] S. He, E. Jonsson, J.R.R.A. Martins, Derivatives for eigenvalues and Eigenvectors via analytic reverse algorithmic differentiation, AIAA J. 60 (4) (2022) 2654–2667, http://dx.doi.org/10.2514/1.J060726.
[62] L. Jin, T.A. Zaki, From streaks to spots and on to turbulence: exploring the dynamics of boundary layer transition, Flow, Turbul. Combust. 91 (3) (2013) 451–473, http://dx.doi.org/10.7281/T17S7KX8.
[63] M.B. Giles, Collected matrix derivative results for forward and reverse mode algorithmic differentiation, in: Advances in Automatic Differentiation, Springer, 2008, pp. 35–44, http://dx.doi.org/10.1007/978-3-540-68942-3_4.
[64] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, 2018.
[65] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362, http://dx.doi.org/10.1038/s41586-020-2649-2.
[66] J. Hunt, Vorticity and vortex dynamics in complex turbulent flows, Trans. Can. Soc. Mech. Eng. 11 (1) (1987) 21–35, http://dx.doi.org/10.1139/tcsme-1987-0004.
[67] P.J. Schmid, D.S. Henningson, Stability and Transition in Shear Flows, Springer, New York, 2001, http://dx.doi.org/10.1007/978-1-4613-0185-1.
[68] R.J. Adrian, Hairpin vortex organization in wall turbulence, Phys. Fluids 19 (4) (2007) http://dx.doi.org/10.1063/1.2717527.
[69] J. Lee, T.A. Zaki, Detection algorithm for turbulent interfaces and large-scale structures in intermittent flows, 175 (2018) http://dx.doi.org/10.1016/j.compfluid.2018.08.015.
[70] T.R. Brooks, G.K.W. Kenway, J.R.R.A. Martins, Benchmark aerostructural models for the study of transonic Aircraft Wings, AIAA J. 56 (7) (2018) 2840–2855, http://dx.doi.org/10.2514/1.J056603.
[71] M. Shahabsafa, A. Mohammad-Nezhad, T. Terlaky, L. Zuluaga, S. He, J.T. Hwang, J.R.R.A. Martins, A novel approach to discrete truss design problems using mixed integer neighborhood search, Struct. Multidiscip. Optim. 58 (2018) 2411–2429, http://dx.doi.org/10.1007/s00158-018-2099-8.
[72] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, b. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 contributorsSciPy 1.0: Fundamental algorithms for scientific computing in Python, Nature Methods 17 (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.
[73] A.R. Conn, N.I.M. Gould, P.L. Toint, Trust Region Methods, SIAM, Philadelphia, PA, 2000.
[74] D.A. Roberts, L.R. Roberts, QR and LQ decomposition matrix backpropagation algorithms for square, wide, and Deep–Real or complex–matrices and their Software implementation, 2020, http://dx.doi.org/10.48550/ARXIV.2009.10071, arXiv preprint arXiv:2009.10071.
[75] M. Giles, An Extended Collection of Matrix Derivative Results for Forward and Reverse Mode Algorithmic Differentiation, Oxford University Computing Laboratory, Oxford, UK, 2008.
[76] A. Hjørungnes, Complex-Valued Matrix Derivatives: With Applications in Signal Processing and Communications, Cambridge University Press, 2011, http://dx.doi.org/10.1017/cbo9780511921490.
[77] A. Hjorungnes, D. Gesbert, Complex-valued Matrix differentiation: Techniques and key results, IEEE Trans. Signal Process. 55 (6) (2007) 2740–2746, http://dx.doi.org/10.1109/tsp.2007.893762.
[78] T. Chandrupatla, A. Belegundu, Introduction to Finite Elements in Engineering, Cambridge University Press, 2021, http://dx.doi.org/10.1017/9781108882293.