This is a preprint of the following article, which is available at: http://mdolab.engin.umich.edu
S. He, Y. Shi, E. Jonsson, J. R. R. A. Martins. Eigenvalue problem derivatives computation for a complex matrix using the adjoint method. *mechanical systems and signal processing*, (In press).
The original article may differ from this preprint and is available at:
https://www.sciencedirect.com/science/article/abs/pii/S0888327022007920.

Eigenvalue problem derivatives computation for a complex matrix using the adjoint method

Sicheng He[1], Yayun Shi[2], Eirikur Jonsson[3], and Joaquim R. R. A. Martins[4]
University of Michigan, Department of Aerospace Engineering, Ann Arbor, MI 48109

Abstract

Derivatives of eigenvalues and eigenvectors with respect to design variables are required for gradient-based optimization in many engineering design problems. However, for the generalized and standard eigenvalue problems with general complex and non-Hermitian coefficient matrices, no method can accurately compute the eigenvalue and eigenvector derivatives while remaining efficient for large numbers of design variables. In this paper, we develop an adjoint method to compute complex eigenvalue and eigenvector derivatives with machine precision. For the special case when only the eigenvalue derivative is required, we propose a reverse algorithmic differentiation (RAD) formula using a newly developed dot product identity for complex functions. We verify the proposed method against the finite differences (FD) for a simple algebraic example with a 3-by-3 complex non-Hermitian matrix and a plane Poiseulle flow stability problem that is modeled as a generalized eigenvalue problem. The adjoint method is demonstrated to scale well with the number of design variables, matching the FD reference to about 5 to 7 digits.

1 Introduction

Eigenvalue and eigenvectors are essential metrics that can be used for dynamic system behavior characterization. They are widely used in engineering applications, such as structural dynamics with mode superposition [1], aeroelastic simulation [2, 3, 4, 5], laminar-turbulence transition prediction [6, 7, 8, 9, 10], buffet-onset prediction [11, 12], reacting flow instability analysis [13], turbine blade mistuning prediction [14, 15, 16, 17], and dynamic system identification [18, 19, 20]. There are different types of eigenvalue problems encountered in practice. In this research, we consider two types of eigenvalue problems that are frequently encountered: (1) Eigenvalue problems of a general complex matrix and (2) generalized eigenvalue problems with complex matrices.

For example, buffet onset [11, 12] and dynamic system identification [18, 19, 20] belong to the first category; structural mode superposition [1] and laminar-turbulence transition [6, 7, 8, 9, 10] belong to the second category. Other types of eigenvalue problems, such as quadratic eigenvalue problems [21] and other nonlinear eigenvalue problems [22], are out of the scope of the paper. In some cases, the coefficient matrices are real and symmetric, e.g., structural mode superposition [1], but in more general cases, the matrices are complex and not Hermitian, e.g., laminar-turbulence transition [6, 7, 8, 9, 10]. Problems whose solution involved repeated eigenvalues are beyond the scope of this paper.

Eigenvalues and eigenvectors derivative with respect to design variables are important information required for gradient-based optimization in many aircraft design related field, e.g., flutter suppression [23, 24], aerodynamic drag reduction optimization with a laminar-turbulent transition model [10], and structural optimization [25, 26, 27, 28, 29, 30]. Thus, it is crucial to compute the derivatives of eigenvalues and eigenvectors accurately and efficiently. For a more extensive review of the field, we refer the reader to a recent review paper by Lin et al. [31].

Several methods exist to compute derivatives, such as finite differences (FD), complex step (CS), algorithmic differentiation (AD), direct method, and adjoint method (see Martins and Ning [32, Chapter 6]), but they differ in the level of accuracy and efficiency. In terms of efficiency, methods either scale well with the number of outputs (functions of interest to be differentiated, eigenvalues and eigenvectors in this case) or with the number of inputs (design variables), but unlikely both [33],[32, Chapter 6].

While FD is prone to truncation and subtraction cancellation errors, CS does not suffer from these limitations (assuming small enough step-size) and can compute the derivative to machine precision [34]. Both FD and CS require little effort to implement, due to their black-box-like application. However, their computational cost scales proportional to the number of inputs, with CS being more costly due to the complex arithmetic. Thus, they are not feasible for many high-fidelity applications with a large number of design variables. For the comparison of the FD and the adjoint method see [32, Chapter 6] (Fig. 6.43 from that chapter) and [35].

Alternatively, we can use AD to compute the derivative. AD is a well-known approach to differentiate a program based on a systematic application of chain rule [32, Chapter 6]. AD can be implemented by transforming the source code line-by-line [36], or, for some fundamental matrix operations, such as matrix products, inversion, and eigenvalue and eigenvector computation, analytic AD formulas can be conveniently derived Dwyer and Macphail [37], Giles [38]. The AD based on the analytic formula has the advantage that the derivatives can then be computed using the optimized libraries without differentiating the underlying library source code. AD can also be classified into (1) forward algorithmic differentiation (FAD) and (2) reverse algorithmic differentiation (RAD) based on the order in which the chain rule is applied. FAD computes the derivatives by applying the chain rule in a forward sequence of operations propagating from the inputs to the outputs; RAD computes the derivatives by applying the chain rule backward, starting with the outputs and ending with the inputs. The computational cost of FAD is proportional to the number of inputs, while the computational cost of RAD is proportional to the number of outputs.

Finally, besides the explicit analytic methods such as AD, we can also use implicit analytic methods. There are two approaches in the implicit analytic methods category: direct and adjoint [32, Chapter 6]. The efficiency of the direct and adjoint approaches depends on the number of inputs and outputs. When the number of inputs is less than the number of outputs, the direct method is preferable. On the other hand, the adjoint method is more efficient when the number of inputs is greater than the number of outputs.

Any of the derivative computation methods mentioned previously can be applied to eigenvalue and eigenvector problems. Previous developments have focused on methods that scale well with respect to the number of outputs, for example, the Nelson method [28] can be categorized as a direct method and [25] is a FAD-based method. However, in many practical design problems, there are many more design variables (usually $\mathcal{O}(100-1000)$) than functions of interest (usually $\mathcal{O}(10)$) [39, 40, 41]. Using the direct method to compute derivatives for these PDE-constrained optimization problems can be prohibitively expensive. Thus, it is crucial to develop methods that can compute derivatives accurately and scale with the number of design variables.

As discussed before, there are mainly two methods that scale well with the number of inputs: (1) the adjoint method proposed by Lee [42], and (2) the RAD method proposed by Giles [38], He et al. [43], Jonsson et al. [23]. The RAD methods can be further decomposed into two categories according to whether it applies an iterative or a projection-based scheme. For the projection-based method, we can classify approaches according to whether a full basis is applied or not. Thus, we have three variations of the RAD method, i.e., (2.a) the RAD with a full basis proposed by Giles [38], (2.b) the RAD with an incomplete basis (the modal-based method) proposed by He et al. [43], and (3.a) the RAD with an iterative method proposed by Jonsson et al. [23]. Among all the approaches, the adjoint method and the RAD with a full basis can achieve machine precision. Lee [42] developed an adjoint derivative formulation for the generalized eigenvalue problem with real and symmetric matrices. Using this formulation, they computed the structural mode shape derivatives with machine precision. Kim and Cho [44] proposed to compute the partial derivatives in the adjoint equation using CS to reduce development effort. Lewandowski and Łasecka-Plura [45] applied the method to solve viscoelastic problems. Yoon et al. [46] developed an efficient adjoint-based method to compute repeated eigenvalue derivatives building on the previous work by Lee [42]. van der Veen et al. [47] studied a control problem with reduced order models where the mode derivatives were computed using the adjoint method. However, the coefficient matrices encountered in Lee [42] are complex and non-Hermitian in many applications. Different adjoint methods have been proposed for the general complex matrices by several authors [48, 49, 26, 50]. However, the adjoint method was named after the *adjugate matrix* (or the *classic adjoint matrix*) and is indeed a modal-based method, as discussed next (see [50]). To obtain machine precision results, it requires high computational cost because it requires the complete knowledge of the eigenvalues and eigenvectors. In this paper, we extend the adjoint method proposed by Lee [42] to eigenvalue problems and generalized eigenvalue problems with general complex and non-Hermitian matrices.

Giles [38] derived an analytic RAD formula for eigenvalue problems using the dot product identity, and as mentioned before, this method can achieve machine precision accuracy. Walter [51] applied the method proposed by Giles [38] to compute higher order derivatives, and Seeger et al. [52] applied them in a deep learning application. Peltzer et al. [53] used the RAD formulas developed by Giles [38] to improve efficiency of a linear algebra package. However, the formula requires the complete knowledge of the eigenvectors, and for problems with large dimensions, the computational cost of computing all eigenvalue and eigenvectors is prohibitive. On the other hand, if only the eigenvalue derivative is computed, the method can compute the derivative accurately if the corresponding eigenvectors are known. Following Giles [38], Roberts and Roberts [54] extended the dot product identity to complex functions (see C for more details). In this paper, we propose a more succinct expression that can be used to derive the identity proposed by Roberts and Roberts [54]. Furthermore, we compare and relate the adjoint method and the RAD formula in eigenvalue derivative computation.

Besides the accurate method developed by Giles [38], we can also approximate the derivatives of eigenvectors using a small set of known eigenvectors as proposed by Fox and Kapoor [25]. This method is known as the modal-based method. We can develop RAD formulas based on the modal-based method [43]. To remedy the truncation error Lim et al. [29], Wang [30] proposed a correction that approximates the higher-order terms based on spectral decomposition. Leveraging this correction method, He et al. [43] proposed RAD formulas to compute eigenvalue and eigenvector derivative that scales favorably with the number of design variables for the generalized eigenvalue problem with positive definite coefficient matrices. They demonstrated that with about six basis vectors, the relative error of the derivatives is about 10^{-6}. However, when more basis vectors were added, the relative error reduction plateaued somewhere between 10^{-6} to 10^{-7}.

The RAD method can also be used with an iterative eigenvalue problem solver, such as a Lanczos method [55] based solver [23]. However, the implementation of this method requires the knowledge of AD tools, such as Tapenade [56]. While transforming highly optimized linear algebra libraries (e.g., LAPACK) is possible, it is tedious and requires significant implementation effort. Its success depends on the transformation tool used and the source code programming paradigm. Furthermore, it is likely that the transformed code performance is sub-optimal compared to the original routine both in terms of speed and memory usage.

Our contribution in this paper is summarized as follows: (1) we develop an adjoint equation for the eigenvalue problem with a general complex matrix, (2) we develop an adjoint equation for the generalized eigenvalue problem with complex matrices, (3) we develop a succinct dot product identity for complex variables, (4) using the dot product identity for complex functions, we find a new formula for eigenvalue derivative computation based on RAD, and (5) we discuss the relationship between the RAD and adjoint methods. The proposed methods can compute the derivative to machine precision, can be implemented easily, scale favorably with the number of design variables, and can be used in gradient-based optimization.

The paper is organized as follows. In Section 2, we present the governing equation for the eigenvalue problems involving complex eigenvectors and the proposed adjoint method. The adjoint method is then extended to the generalized eigenvalue problems in Section 3. In Section 3, we develop a RAD formula when the function of interest is only the eigenvalue, using our newly proposed dot product identity for complex functions. Then, in Section 4, we present two test cases to verify the formulas we obtained in Section 2 and Section 3. The test cases include a simple algebraic problem with a complex 3-by-3 coefficient matrix and a plane Poiseulle flow stability problem modeled as a generalized eigenvalue problem. Finally, we present our conclusions in Section 5.

2 Eigenvalue problem

In this section, we discuss the eigenvalue problem with a complex coefficient matrix and the proposed adjoint method to compute eigenvalue or eigenvector derivatives. This is a special case of the generalized eigenvalue problem we present in Section 3. In Section 2.1, we introduce the eigenvalue residual form, followed by the derivation of adjoint method in Section 2.2 presenting. Finally, Section 2.3 presents the RAD formula for derivatives of eigenvalues only.

2.1 Governing equation

The eigenvalue problem is given by

$$\mathbf{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}, \tag{1}$$

where the coefficient matrix is in general a complex matrix, $\mathbf{A} \in \mathbb{C}^{n\times n}$, the eigenvector is a complex vector, $\boldsymbol{\phi} \in \mathbb{C}^n$, the eigenvalue is a complex scalar, $\lambda \in \mathbb{C}$, and n is the dimension of the coefficient matrix. We assume that all eigenvalues are distinct, that is, there are no repeated eigenvalues. In practice, the repeated eigenvalues are usually due to some spatial symmetry [31]. Thus, it is less common compared with the case that all eigenvalues are distinct. Given these definitions and the assumption, we can prove that the eigenvalue is analytic as a function of matrix entries (see A for the proof).

However, the eigenvalue problem given in Eq. (1) cannot determine one unique eigenvector given an eigenvalue. This is because the eigenvector remains an eigenvector after scaling and rotating in the complex space. Suppose that $\lambda, \boldsymbol{\phi}$ is a complex eigenpair of a matrix $\mathbf{A}$. By applying stretching and rotation about the origin in the complex plane, we obtain the following equation

$$\mathbf{A}\left(\alpha\boldsymbol{\phi}e^{i\theta}\right) = \lambda\left(\alpha\boldsymbol{\phi}e^{i\theta}\right), \tag{2}$$

where $\alpha \in \mathbb{R}$ is the scaling factor, and $\theta \in \mathbb{R}$ is the rotation angle. Here, $\alpha\boldsymbol{\phi}e^{i\theta}$ is also an eigenvector. An illustration of scaling and rotating of a complex eigenvector is shown in Fig. 1.

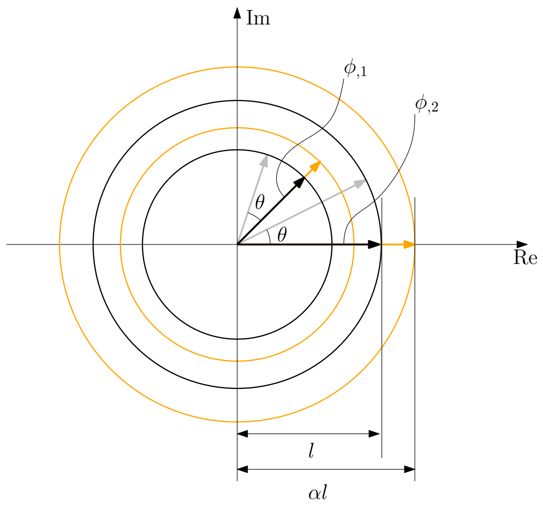

Figure 1: Scaling and rotation of a complex eigenvector. The black arrows indicate the original eigenvector $\boldsymbol{\phi} = (\phi_{,1}, \phi_{,2})$. The orange lines show $\boldsymbol{\phi}$ after scaling by α. The gray lines show $\boldsymbol{\phi}$ after a rotation by θ. l is the norm of $\phi_{j,2}$.

Thus, to obtain one unique solution we need to constrain the eigenvector to a certain length and angle. The norm is constrained by

$$\boldsymbol{\phi}^*\boldsymbol{\phi} = 1, \tag{3}$$

where $(\cdot)^*$ is a conjugate transpose operator. There are many ways to constrain the angle of a complex eigenvector. One approach is to make the entry with the maximum norm be a positive real number. We can express these conditions as

$$\begin{aligned} \mathrm{Im}\left(\boldsymbol{\phi}_{,k}\right) &= 0 \\ \mathrm{Re}\left(\boldsymbol{\phi}_{,k}\right) &> 0 \\ k &= \mathrm{argmax}_j ||\boldsymbol{\phi}_{,j}||_2 \end{aligned}, \tag{4}$$

where j is the entry index of the eigenvector.

To summarize, solving the following equation gives a unique complex eigenvector

$$\begin{aligned} \mathbf{A}\boldsymbol{\phi} &= \lambda\boldsymbol{\phi} \\ \boldsymbol{\phi}^*\boldsymbol{\phi} &= 1 \\ \mathrm{Im}\left(\boldsymbol{\phi}_{,k}\right) &= 0 \\ \mathrm{Re}\left(\boldsymbol{\phi}_{,k}\right) &> 0 \\ k &= \mathrm{argmax}_j ||\boldsymbol{\phi}_{,j}||_2 \end{aligned}. \tag{5}$$

If multiple entries have the same norm, but they are not equal with each other, we set the value of k to the smallest entry index.

Equation (5) is written using complex numbers. However, we can expand and split the complex equation into two real equations, namely the real and imaginary components of the original equation. The resulting system of equations can then be written in terms of real numbers only as

$$\mathbf{r}(\mathbf{w}) = 0, \tag{6}$$

where $\mathbf{r}(\mathbf{w})$ and $\mathbf{w}$ are defined as

$$\mathbf{r}(\mathbf{w}) = \begin{bmatrix} \mathbf{r}_{\text{main},r} \\ \mathbf{r}_{\text{main},i} \\ \mathbf{r}_m \\ \mathbf{r}_p \end{bmatrix} = \begin{bmatrix} \mathbf{A}_r\boldsymbol{\phi}_r - \mathbf{A}_i\boldsymbol{\phi}_i - \lambda_r\boldsymbol{\phi}_r + \lambda_i\boldsymbol{\phi}_i \\ \mathbf{A}_i\boldsymbol{\phi}_r + \mathbf{A}_r\boldsymbol{\phi}_i - \lambda_i\boldsymbol{\phi}_r - \lambda_r\boldsymbol{\phi}_i \\ \boldsymbol{\phi}_r^\intercal\boldsymbol{\phi}_r + \boldsymbol{\phi}_i^\intercal\boldsymbol{\phi}_i - 1 \\ \mathbf{e}_k^\intercal\boldsymbol{\phi}_i \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \boldsymbol{\phi}_r \\ \boldsymbol{\phi}_i \\ \lambda_r \\ \lambda_i \end{bmatrix}, \tag{7}$$

where the subscript “main” distinguishes the eigenvalue equations from the additional phase and magnitude equations, the subscription r and i represents real and imaginary parts, respectively, the subscription m and p represents the magnitude and the phase, respectively, and the state variable $\mathbf{w}$ is obtained by stacking the eigenvector and the eigenvalue together.

2.2 Adjoint method

Now we compute the derivative of a real function $f(\boldsymbol{\phi}, \lambda)$ with respect to the design variables $\mathbf{x}$ where the matrix $\mathbf{A}(\mathbf{x})$ is directly dependent on $\mathbf{x}$. If the function f is otherwise complex, we can compute its real and imaginary component derivatives separately following a similar routine. Using the notation of [32, Sec. 6.7], we formulate the total derivative of the adjoint method as

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \boldsymbol{\psi}^\intercal \frac{\partial \mathbf{r}}{\partial \mathbf{x}}, \tag{8}$$

where the partial derivatives are provided and the vector of adjoint variables $\boldsymbol{\psi}$ is obtained by solving the adjoint equation

$$\frac{\partial \mathbf{r}}{\partial \mathbf{w}}^\intercal \boldsymbol{\psi} = \frac{\partial f}{\partial \mathbf{w}}^\intercal, \tag{9}$$

which is a linear system. In our case, this linear system can be expanded as

$$\begin{aligned} & \frac{\partial \mathbf{r}}{\partial \mathbf{w}}^\intercal \boldsymbol{\psi} = \frac{\partial f}{\partial \mathbf{w}}^\intercal \\ \Leftrightarrow & \begin{bmatrix} \mathbf{A}_r - \lambda_r\mathbf{I} & -\mathbf{A}_i + \lambda_i\mathbf{I} & -\boldsymbol{\phi}_r & \boldsymbol{\phi}_i \\ \mathbf{A}_i - \lambda_i\mathbf{I} & \mathbf{A}_r - \lambda_r\mathbf{I} & -\boldsymbol{\phi}_i & -\boldsymbol{\phi}_r \\ 2\boldsymbol{\phi}_r^\intercal & 2\boldsymbol{\phi}_i^\intercal & 0 & 0 \\ 0 & \mathbf{e}_k^\intercal & 0 & 0 \end{bmatrix}^\intercal \begin{bmatrix} \boldsymbol{\psi}_{\text{main,r}} \\ \boldsymbol{\psi}_{\text{main,i}} \\ \boldsymbol{\psi}_m \\ \boldsymbol{\psi}_p \end{bmatrix} = \frac{\partial f}{\partial \mathbf{w}}^\intercal. \end{aligned} \tag{10}$$

After solving these adjoint equations (10), we evaluate $(\partial \mathbf{r}/\partial \mathbf{x})^\intercal \boldsymbol{\psi}$, which can be further decomposed using the chain rule as

$$\frac{\partial \mathbf{r}}{\partial \mathbf{x}}^\intercal \boldsymbol{\psi} = \frac{\partial \mathbf{A}_r}{\partial \mathbf{x}}^\intercal \frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^\intercal \boldsymbol{\psi} + \frac{\partial \mathbf{A}_i}{\partial \mathbf{x}}^\intercal \frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^\intercal \boldsymbol{\psi}, \tag{11}$$

where $\partial \mathbf{A}_r / \partial \mathbf{x}$ and $\partial \mathbf{A}_i / \partial \mathbf{x}$ are problem-specific in the sense that they depend on the design variables of the problem and usually straight-forward to evaluate; while $(\partial \mathbf{r}/\partial \mathbf{A}_r)^\intercal \boldsymbol{\psi}$ and $(\partial \mathbf{r}/\partial \mathbf{A}_i)^\intercal \boldsymbol{\psi}$ are general. The derivative expression here involves matrices, e.g., $(\partial \mathbf{r}/\partial \mathbf{A}_r)^\intercal \boldsymbol{\psi}$ that involves tensor-vector product and may cause confusion. To avoid that, we assume that the derivative is computed after the matrices are flattened as vectors, and in the final result, the vectors are mapped back to the original matrix sizes. This operation is defined in B. We can expand these two terms as

$$\begin{aligned}
\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^\intercal \boldsymbol{\psi} &= \boldsymbol{\psi}_{\text{main},r} \boldsymbol{\phi}_r^\intercal + \boldsymbol{\psi}_{\text{main},i} \boldsymbol{\phi}_i^\intercal \\
\frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^\intercal \boldsymbol{\psi} &= -\boldsymbol{\psi}_{\text{main},r} \boldsymbol{\phi}_i^\intercal + \boldsymbol{\psi}_{\text{main},i} \boldsymbol{\phi}_r^\intercal,
\end{aligned} \tag{12}$$

where the adjoint vector can be a complex vector. These equations are derived in G.

Finally, if we want to compute the derivative of the eigenvalue and eigenvector with respect to the entries of the coefficient matrix $\mathbf{A}$, we can use the total derivative equation (8) with $\mathbf{A}$ in place of $\mathbf{x}$.

2.3 RAD formula to computing derivatives of eigenvalues

Before we proceed with the analytic formula, we define the forward ($\dot{\square}$) and reverse seeds ($\overline{\square}$). Consider a computation with one input, s_I, and one output, s_O. Suppose matrix $\mathbf{A}$ is some intermediate variable within the computation, then $\dot{\mathbf{A}}$ denotes the derivative of $\mathbf{A}$ with respect to s_I and $\overline{\mathbf{A}}$ denotes the derivative of s_O with respect to elements of $\overline{\mathbf{A}}$.

When we only need the eigenvalue derivatives, a more efficent method can be applied. The FAD form is given by Magnus [57] as

$$\tilde{\boldsymbol{\phi}}^* \dot{\mathbf{A}} \boldsymbol{\phi} = \dot{\lambda} \tilde{\boldsymbol{\phi}}^* \boldsymbol{\phi}. \tag{13}$$

where $\tilde{\boldsymbol{\phi}}$ is a left eigenvector corresponding with the complex conjugate eigenvalue λ^*. The left eigenvector satisfies the following equation:

$$\mathbf{A}^* \tilde{\boldsymbol{\phi}} = \lambda^* \tilde{\boldsymbol{\phi}}. \tag{14}$$

We derive the RAD formula using proposed complex dot product identity presented in D. The detailed derivation is included in E. The results can be summarized as

$$\begin{aligned}
\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r} &= \operatorname{Re}\left(\frac{\tilde{\boldsymbol{\phi}} \boldsymbol{\phi}^*}{\boldsymbol{\phi}^* \tilde{\boldsymbol{\phi}}}\right), \\
\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i} &= \operatorname{Im}\left(\frac{\tilde{\boldsymbol{\phi}} \boldsymbol{\phi}^*}{\boldsymbol{\phi}^* \tilde{\boldsymbol{\phi}}}\right), \\
\frac{\mathrm{d}\lambda_i}{\mathrm{d}\mathbf{A}_r} &= -\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i}, \\
\frac{\mathrm{d}\lambda_i}{\mathrm{d}\mathbf{A}_i} &= \frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r},
\end{aligned} \tag{15}$$

The last two equations are due to the Cauchy–Riemann conditions (61) for an analytic function. In F we explore the relation between Eq. (12) and Eq. (15). When an eigenvalue derivative is sought, Eq. (15) is cheaper to evaluate instead of using Eq. (12). Thus, we recommend using Eq. (15).

3 Generalized eigenvalue problem

The generalized eigenvalue problems are frequently encountered in engineering applications. In this section, we extend the adjoint method to this class of problems.

3.1 Governing equation

The generalized eigenvalue problem is defined as follows:

$$\mathbf{K}\boldsymbol{\phi} = \lambda\mathbf{M}\boldsymbol{\phi} \tag{16}$$

where $\mathbf{K}$ and $\mathbf{M}$ are complex matrices. The governing equation for the generalized eigenvalue problem is as follows:

$$\begin{aligned} \mathbf{K}\boldsymbol{\phi} &= \lambda\mathbf{M}\boldsymbol{\phi} \\ \boldsymbol{\phi}^*\boldsymbol{\phi} &= 1 \\ \operatorname{Im}\left(\boldsymbol{\phi}_{,k}\right) &= 0 \\ \operatorname{Re}\left(\boldsymbol{\phi}_{,k}\right) &> 0 \\ k &= \operatorname{argmax}_j ||\boldsymbol{\phi}_{,j}||_2 \end{aligned} \quad , \tag{17}$$

There are other ways to normalize the eigenvectors. For example, a commonly used normalization functions yields

$$\boldsymbol{\phi}^\intercal\mathbf{M}\boldsymbol{\phi} = 1. \tag{18}$$

Different normalization conditions can be taken into account by replacing the normalization condition in Eq. (17).

Expanding Eq. (17) to separate real and imaginary components, we obtain

$$\begin{aligned} \mathbf{r}(\mathbf{w}) &= \begin{bmatrix} \mathbf{r}_{\text{main},r} \\ \mathbf{r}_{\text{main},i} \\ \mathbf{r}_m \\ \mathbf{r}_p \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{K}_r - \lambda_r\mathbf{M}_r + \lambda_i\mathbf{M}_i)\,\boldsymbol{\phi}_r + (-\mathbf{K}_i + \lambda_i\mathbf{M}_r + \lambda_r\mathbf{M}_i)\,\boldsymbol{\phi}_i \\ (\mathbf{K}_i - \lambda_i\mathbf{M}_r - \lambda_r\mathbf{M}_i)\,\boldsymbol{\phi}_r + (\mathbf{K}_r - \lambda_r\mathbf{M}_r + \lambda_i\mathbf{M}_i)\,\boldsymbol{\phi}_i \\ \boldsymbol{\phi}_r^\intercal\boldsymbol{\phi}_r + \boldsymbol{\phi}_i^\intercal\boldsymbol{\phi}_i - 1 \\ \mathbf{e}_k^\intercal\boldsymbol{\phi}_i \end{bmatrix}, \end{aligned} \tag{19}$$

where

$$\mathbf{w} = \begin{bmatrix} \boldsymbol{\phi}_r \\ \boldsymbol{\phi}_i \\ \lambda_r \\ \lambda_i \end{bmatrix}. \tag{20}$$

3.2 Adjoint method

The total derivative equation (8) still holds. However, the adjoint equation for the generalized eigenvalue problem is different and is as follows:

$$\begin{aligned}
&\frac{\partial \mathbf{r}}{\partial \mathbf{w}}^{\intercal} \boldsymbol{\psi} = \frac{\partial f}{\partial \mathbf{w}}^{\intercal} \\
\Leftrightarrow &\begin{bmatrix} \mathbf{K}_r - \lambda_r \mathbf{M}_r + \lambda_i \mathbf{M}_i & -\mathbf{K}_i + \lambda_i \mathbf{M}_r + \lambda_r \mathbf{M}_i & -\mathbf{M}_r \boldsymbol{\phi}_r + \mathbf{M}_i \boldsymbol{\phi}_i & \mathbf{M}_i \boldsymbol{\phi}_r + \mathbf{M}_r \boldsymbol{\phi}_i \\ \mathbf{K}_i - \lambda_i \mathbf{M}_r - \lambda_r \mathbf{M}_i & \mathbf{K}_r - \lambda_r \mathbf{M}_r + \lambda_i \mathbf{M}_i & -\mathbf{M}_r \boldsymbol{\phi}_i - \mathbf{M}_i \boldsymbol{\phi}_r & \mathbf{M}_i \boldsymbol{\phi}_i - \mathbf{M}_r \boldsymbol{\phi}_r \\ 2\boldsymbol{\phi}_r^{\intercal} & 2\boldsymbol{\phi}_i^{\intercal} & 0 & 0 \\ 0 & \mathbf{e}_k^{\intercal} & 0 & 0 \end{bmatrix}^{\intercal} \\
&\cdot \begin{bmatrix} \boldsymbol{\psi}_{\text{main,r}} \\ \boldsymbol{\psi}_{\text{main,i}} \\ \boldsymbol{\psi}_m \\ \boldsymbol{\psi}_p \end{bmatrix} = \frac{\partial f}{\partial \mathbf{w}}^{\intercal}.
\end{aligned} \tag{21}$$

Also, the $(\partial \mathbf{r}^{\intercal}/\partial \mathbf{x})\,\boldsymbol{\psi}$ is different from that of Eq. (11). The $(\partial \mathbf{r}^{\intercal}/\partial \mathbf{x})\,\boldsymbol{\psi}$ term is given by

$$\frac{\partial \mathbf{r}}{\partial \mathbf{x}}^{\intercal} \boldsymbol{\psi} = \frac{\partial \mathbf{M}_r}{\partial \mathbf{x}}^{\intercal} \frac{\partial \mathbf{r}}{\partial \mathbf{M}_r}^{\intercal} \boldsymbol{\psi} + \frac{\partial \mathbf{M}_i}{\partial \mathbf{x}}^{\intercal} \frac{\partial \mathbf{r}}{\partial \mathbf{M}_i}^{\intercal} \boldsymbol{\psi} + \frac{\partial \mathbf{K}_r}{\partial \mathbf{x}}^{\intercal} \frac{\partial \mathbf{r}}{\partial \mathbf{K}_r}^{\intercal} \boldsymbol{\psi} + \frac{\partial \mathbf{K}_i}{\partial \mathbf{x}}^{\intercal} \frac{\partial \mathbf{r}}{\partial \mathbf{K}_i}^{\intercal} \boldsymbol{\psi}, \tag{22}$$

where $\partial \mathbf{K}_r/\partial \mathbf{x}$, $\partial \mathbf{K}_i/\partial \mathbf{x}$, $\partial \mathbf{M}_r/\partial \mathbf{x}$, and $\partial \mathbf{M}_i/\partial \mathbf{x}$ are problem-specific and usually straight-forward to evaluate; while $(\partial \mathbf{r}/\partial \mathbf{K}_r)^{\intercal}\,\boldsymbol{\psi}$, $(\partial \mathbf{r}/\partial \mathbf{K}_i)^{\intercal}\,\boldsymbol{\psi}$, $(\partial \mathbf{r}/\partial \mathbf{M}_r)^{\intercal}\,\boldsymbol{\psi}$, and $(\partial \mathbf{r}/\partial \mathbf{M}_i)^{\intercal}\,\boldsymbol{\psi}$ are general. As before for the standard eigenvalue problem, the derivative expressions involve matrices and are treated as defined in B. The final expressions are

$$\begin{aligned}
\frac{\partial \mathbf{r}}{\partial \mathbf{K}_r}^{\intercal} \boldsymbol{\psi} &= \boldsymbol{\psi}_{\text{main},r} \boldsymbol{\phi}_r^{\intercal} + \boldsymbol{\psi}_{\text{main},i} \boldsymbol{\phi}_i^{\intercal}, \\
\frac{\partial \mathbf{r}}{\partial \mathbf{K}_i}^{\intercal} \boldsymbol{\psi} &= -\boldsymbol{\psi}_{\text{main},r} \boldsymbol{\phi}_i^{\intercal} + \boldsymbol{\psi}_{\text{main},i} \boldsymbol{\phi}_r^{\intercal}, \\
\frac{\partial \mathbf{r}}{\partial \mathbf{M}_r}^{\intercal} \boldsymbol{\psi} &= \boldsymbol{\psi}_{\text{main},r} \left(-\lambda_r \boldsymbol{\phi}_r + \lambda_i \boldsymbol{\phi}_i\right)^{\intercal} + \boldsymbol{\psi}_{\text{main},i} \left(-\lambda_i \boldsymbol{\phi}_r - \lambda_r \boldsymbol{\phi}_i\right)^{\intercal}, \\
\frac{\partial \mathbf{r}}{\partial \mathbf{M}_i}^{\intercal} \boldsymbol{\psi} &= \boldsymbol{\psi}_{\text{main},r} \left(\lambda_i \boldsymbol{\phi}_r + \lambda_r \boldsymbol{\phi}_i\right)^{\intercal} + \boldsymbol{\psi}_{\text{main},i} \left(-\lambda_r \boldsymbol{\phi}_r + \lambda_i \boldsymbol{\phi}_i\right)^{\intercal}.
\end{aligned} \tag{23}$$

The detailed derivation is similar to Eq. (12), which is presented in G.

4 Numerical results

In this section, we verify the proposed adjoint methods with FD using two test cases. The first case is a simple algebraic problem with a 3×3 matrix, where we demonstrate and verify the adjoint and the RAD expressions. The second case involves the more complicated Poiseuille flow modeled with Orr–Sommerfeld and Squire's equation, which we used to verify the adjoint expressions for generalized eigenvalue problems.

4.1 Eigenvalue problem test case: the eigenvalue problem adjoint method verification

Consider the 3×3 complex matrix,

$$\mathbf{A} = \mathbf{A}_r + i\mathbf{A}_i, = \begin{bmatrix} -1.01 & 0.86 & -4.60 \\ 3.98 & 0.53 & -7.04 \\ 3.30 & 8.26 & -3.89 \end{bmatrix} + i \begin{bmatrix} 0.30 & 0.79 & 5.47 \\ 7.21 & 1.90 & 0.58 \\ 3.42 & 8.97 & 0.30 \end{bmatrix}. \tag{24}$$

The matrix values are arbitrarily chosen by generating random numbers in the range of $(-10, 10)$. The first eigenpair $\boldsymbol{\phi}, \lambda$ of this system is

$$\begin{aligned} \boldsymbol{\phi} = \boldsymbol{\phi}_r + i\boldsymbol{\phi}_i &= \begin{bmatrix} 0.378298320174238 \\ 0.448628978890548 \\ 0.703251318380440 \end{bmatrix} + i \begin{bmatrix} 0.211732867893793 \\ 0.340924032744271 \\ 0 \end{bmatrix}, \\ \lambda = \lambda_r + i\lambda_i &= -2.22367558699108 + i12.859852984709278. \end{aligned} \tag{25}$$

Moreover, the full set of eigenvalues is

$$\boldsymbol{\Lambda} = \begin{bmatrix} -2.22367558699108 + i12.85985298470927 \\ -5.81588878300751 - i2.05104471148432 \\ 3.66956436999860 - i8.30880827322497 \end{bmatrix}, \tag{26}$$

where $\boldsymbol{\Lambda}$ is a vector contains all eigenvalues. No repeated eigenvalues show up in this case.

The function of interest f is defined as

$$f = f_r + if_i. \tag{27}$$

We choose a linear function of interest, involving the first eigenpair $(\boldsymbol{\phi}, \lambda)$,

$$f = \mathbf{c}_1^\intercal \boldsymbol{\phi} + c_2\lambda, \tag{28}$$

where the constants $\mathbf{c}_1$ and c_2 are defined as

$$\begin{aligned} \mathbf{c}_1 = \mathbf{c}_{1r} + i\mathbf{c}_{1i} &= \begin{bmatrix} 0.16 \\ 0.53 \\ 0.11 \end{bmatrix} + i \begin{bmatrix} 0.78 \\ 0.11 \\ 0.77 \end{bmatrix}, \\ c_2 = c_{2r} + ic_{2i} &= 1.0 + i0.5, \end{aligned} \tag{29}$$

Similar to the coefficient matrices, the value of each entry is arbitrarily chosen by generating random numbers in the range of $(-10, 10)$. Expanding the function of interest terms of real and imaginary components we have

$$\begin{aligned} f_r &= \mathbf{c}_{1r}^\intercal \boldsymbol{\phi}_r - \mathbf{c}_{1i}^\intercal \boldsymbol{\phi}_i + c_{2r}\lambda_r - c_{2i}\lambda_i, \\ f_i &= \mathbf{c}_{1r}^\intercal \boldsymbol{\phi}_i + \mathbf{c}_{1i}^\intercal \boldsymbol{\phi}_r + c_{2r}\lambda_i + c_{2i}\lambda_r. \end{aligned} \tag{30}$$

The goal is to compute $\mathrm{d}f/\,\mathrm{d}\mathbf{A}$, in particular $\mathrm{d}f_r/\,\mathrm{d}\mathbf{A}_r$, $\mathrm{d}f_r/\,\mathrm{d}\mathbf{A}_i$, $\mathrm{d}f_i/\,\mathrm{d}\mathbf{A}_r$, and $\mathrm{d}f_i/\,\mathrm{d}\mathbf{A}_i$.

We compute the derivative using the proposed adjoint method. First, we form the RHS of the adjoint equation (10) and obtain,

$$\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{w}} = \begin{bmatrix} \mathbf{c}_{1r} \\ -\mathbf{c}_{1i} \\ c_{2r} \\ -c_{2i} \end{bmatrix}, \quad \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{w}} = \begin{bmatrix} \mathbf{c}_{1i} \\ \mathbf{c}_{1r} \\ c_{2i} \\ c_{2r} \end{bmatrix}. \tag{31}$$

Then, solving for the adjoint variables using Eq. (10) and we apply the following equations to compute the total derivatives

$$\begin{aligned} \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^{\intercal} \boldsymbol{\psi}_r, \\ \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^{\intercal} \boldsymbol{\psi}_r, \\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_r}^{\intercal} \boldsymbol{\psi}_i, \\ \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{A}_i}^{\intercal} \boldsymbol{\psi}_i, \end{aligned} \tag{32}$$

where $\boldsymbol{\psi}_r$ and $\boldsymbol{\psi}_i$ are adjoint vectors for f_r and f_i, respectively. Finally, applying the first row from Eq. (8), Eq. (11) and Eq. (12), we can compute the derivatives. We tabulated the first row of the derivative matrix in Table 1. We note that the most computational expensive steps are the adjoint equation solutions for $\boldsymbol{\psi}_r$ and $\boldsymbol{\psi}_i$.

Table 1: Verification of the adjoint derivatives with FD

Type	Index	Adjoint	FD
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,1)$	0.315879228919551	0.3158792$\underline{14340953}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,2)$	0.408674084806795	0.4086740$\underline{75913495}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,3)$	0.437931392924164	0.437931392$\underline{482938}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,1)$	−0.011401200176911	−0.0114012$\underline{10642248}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,2)$	0.039666848554661	0.0396668$\underline{58242526}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,3)$	−0.254254544722728	−0.2542545$\underline{33871290}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,1)$	0.011625919400695	0.011625$\underline{894913436}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,2)$	−0.042150227409544	−0.0421502$\underline{37078431}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,3)$	0.266721810617628	0.266721$\underline{789543567}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,1)$	0.300544793153509	0.300544$\underline{812148473}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,2)$	0.388896314270688	0.38889631$\underline{9591936}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,3)$	0.416402795592258	0.416402$\underline{807346117}$

Now we compute the derivative using the FD. We compute the derivative $\mathrm{d}f/d\mathbf{A}$

using FD with step size $\epsilon = 10^{-6}$ using the following formulas:

$$
\begin{aligned}
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_{r,pq}} &= \mathrm{Re}\left(\frac{f(\mathbf{A}+\epsilon\mathbf{E}_{pq})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{A}_{i,pq}} &= \mathrm{Im}\left(\frac{f(\mathbf{A}+i\epsilon\mathbf{E}_{pq})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_{r,pq}} &= \mathrm{Re}\left(\frac{f(\mathbf{A}+\epsilon\mathbf{E}_{pq})-f(\mathbf{A})}{\epsilon}\right),\\
\frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{A}_{i,pq}} &= \mathrm{Im}\left(\frac{f(\mathbf{A}+i\epsilon\mathbf{E}_{pq})-f(\mathbf{A})}{\epsilon}\right),
\end{aligned}
\tag{33}
$$

where $\mathbf{E}_{pq}$ is a matrix with a single entry set to one (indexed as (p, q)) and all other entries are zero. The underlining in Table 1 indicates digits that differ from those computed with the adjoint method. Overall, 5 to 7 digits match between the FD and the adjoint method. Thus, it demonstrates that the adjoint method can be used to compute the eigenvalue and eigenvector derivatives accurately. However, in comparison with the adjoint method, for each entry (p, q), the eigenpair needs to be computed again. So in total, this requires solving an eigenvalue problem for $N \times N$ times, where $N \times N$ is the matrix dimension. Compared with the adjoint equation solutions, the FD requires many more operations.

4.2 Eigenvalue problem test case: a simple algebraic problem for the RAD formula verification

In this section, we verify the proposed RAD formula (15) for eigenvalue derivative computation. We reuse the coefficient matrix $\mathbf{A}$ defined in Eq. (24) and compute the eigenvalue derivative using the RAD and the adjoint method.

Using the RAD formula defined by Eq. (15), we computed the derivatives listed in Table 2. We also computed the derivative using the adjoint method (8), (10), where the differences are highlighted by underlines. Overall, the results match to machine precision. This is consistent with the fact that the RAD and the adjoint method are equivalent. Both are accurate and are not subject to the errors encountered when using FD approximations. Finally, we compute the derivatives using FD. Similar to the adjoint method results, the differing digits are highlighted with underlines. Overall, 6 to 8 digits match between the FD and the adjoint method. As discussed before, the differences are caused by the errors involved FD approximations. Moreover, as we discussed before, the Cauchy–Riemann condition holds here because the eigenvalues are distinct. This is verified in the table.

4.3 Eigenvalue problem test case: comparison of the adjoint method and the RAD method derivative computation efficiency

In this section, we compare the computational speed and scalability of the two proposed eigenvalue derivative computation methods, the adjoint method (see Eq. (8)) and the RAD formula (see Eq. (15)). In this study, we use a sparse $n \times n$ matrix with tridiagonal

Table 2: Verification of the RAD derivatives with the adjoint method and FD

Type	Index	RAD	Adjoint	FD
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,1)$	0.229556432543903	0.229556432543903	$0.2295564\underline{18318566}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,2)$	0.319488240798766	$0.31948824079876\underline{7}$	$0.3194882\underline{38531562}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_r$	$(1,3)$	0.219681767737122	$0.21968176773712\underline{3}$	$0.2196817\underline{74993319}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,1)$	0.132865295735042	0.132865295735042	$0.1328652\underline{88549056}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,2)$	0.129506466458660	0.129506466458660	$0.1295064\underline{53561135}$
$\mathrm{d}f_r/\mathrm{d}\mathbf{A}_i$	$(1,3)$	0.369950215573716	$0.36995021557371\underline{7}$	$0.369950\underline{194922808}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,1)$	-0.132865295735042	-0.132865295735042	$-0.132865295\underline{654483}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,2)$	-0.129506466458660	-0.129506466458660	$-0.1295064\underline{53561135}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_r$	$(1,3)$	-0.369950215573716	$-0.36995021557371\underline{7}$	$-0.3699502\underline{05580949}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,1)$	0.229556432543903	0.229556432543903	$0.2295564\underline{62727487}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,2)$	0.319488240798766	$0.31948824079876\underline{7}$	$0.3194882\underline{50966060}$
$\mathrm{d}f_i/\mathrm{d}\mathbf{A}_i$	$(1,3)$	0.219681767737122	0.219681767737122	$0.2196817\underline{94533244}$

entries statistically generated satisfying the standard normal distribution and other entries are all zero. Each entry in the matrix is independent and identically distributed. The choice of the sparse matrix is motivated by the fact that in many engineering applications, such as computational fluid dynamics (CFD), the Jacobian matrix is sparse. The dominant eigenvalue is considered in this test.

We compare the adjoint equation solution time with the left eigenvalue solution time required by the RAD formula both solved to the tolerance of 10^{-8}. The adjoint equation is solved using the generalized minimal residual iteration (GMRES) method [58], implemented in SciPy [59], using the function *scipy.sparse.linalg.gmres* The eigenvalue problem and the left eigenvalue problem are solved using the implicitly restarted Arnoldi method, using the *scipy.sparse.linalg.eigs* function. In addition, thanks to the knowledge of the eigenvalue from solving the original eigenvalue problem, for the RAD equation, a shift-and-invert spectral transform is used to improved the convergence rate. For more details about shift-and-invert spectral transform, see [60]. A row-based sparse matrix format is applied in this study.

The computational wall time versus the number of rows for the square matrix is plotted in Fig. 2. We consider 10 different randomly generated cases, with the mean and variance of the wall time also shown.We also include the primal eigenvalue problem solution time as a reference. From Fig. 2, we find that the variance of all methods is small compared with their means. Also, the solution time scales linearly with a similar trend was observed for problems with few off-diagonal terms [61]. The computational cost of the RAD formula and the adjoint method is smaller compared with the eigenvalue solution. For the test conducted here only a single set of solvers from SciPy was used, with no preconditioners applied, but it's possible that with improved solvers and tailored preconditioners that the adjoint method performance might improve relative to RAD method. To develop efficient preconditioners for the adjoint equation is itself a research topic worth exploring and is beyond the scope of this paper. However, we recommend the use of the RAD formula if the function of interest is only dependent on the eigenvalue and not on the eigenvector as this approach offers an easier implementation with consistent scaling and performance.

Note that we only record the time for the adjoint equation solution and the left eigenvalue problem solution for the RAD formula. We do not record the time to form the total derivative matrix because in practice the total derivative matrix is low rank (see Eq. (15) and Eq. (12)) but dense. Thus, it is better to store it in the matrix outer product format.

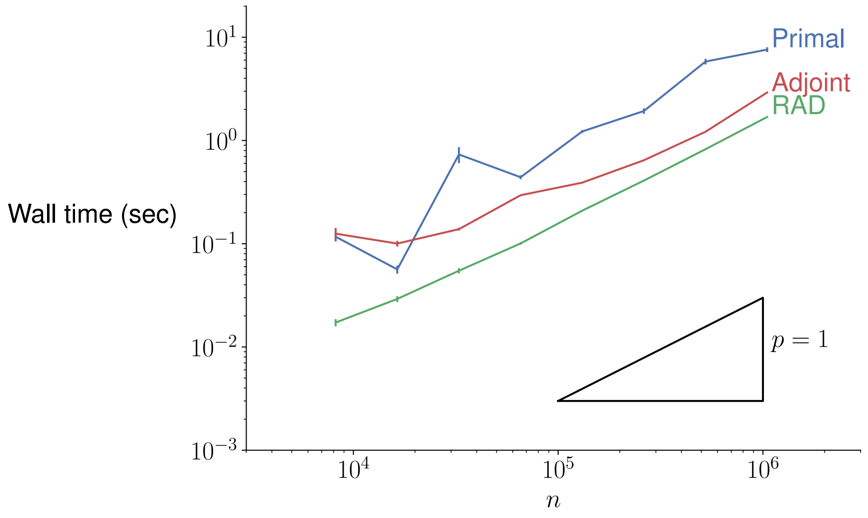

Figure 2: Scalability of the adjoint and the RAD methods for eigenvalue derivative computation. For the adjoint method, only the adjoint equation solution time is shown. For the RAD formula, only the left eigenvalue problem solution time is shown. The x-axis, n, is the row (column) number of the matrix. The y-axis, T, is the wall time in second.

To summarize, we recommend using the proposed RAD method for the eigenvalue derivative computation due to its slightly better efficiency and the fact that it requires less implementation effort. However, this needs be tested for a more general set of problems to make sure the conclusion is universal.

4.4 Eigenvalue problem test case: comparison of the adjoint method and the adjugate method derivative computation efficiency

In this section, we compare the performance of the adjoint method and the adjugate method for eigenvector derivative computation. We apply the same method to solve the adjoint equation as stated in Section 4.3. Rogers [26] proposed the following formula to compute the eigenvector derivatives

$$\frac{\mathrm{d}\boldsymbol{\phi}^{(i)}}{\mathrm{d}\mathbf{x}_k} = \sum_{j=1}^{n} c_{ijk}\boldsymbol{\phi}^{(j)}, \tag{34}$$

where $\boldsymbol{\phi}^{(i)}$ denotes the ith eigenvector, $\mathbf{x}_k$ is the kth design variable, and the coefficient c_{ijk} is defined as follows

$$c_{ijk} = \frac{\tilde{\boldsymbol{\phi}}^{(j)\intercal} \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}\mathbf{x}_k} \boldsymbol{\phi}^{(i)}}{(\lambda^{(i)} - \lambda^{(j)}) \tilde{\boldsymbol{\phi}}^{(j)\intercal} \boldsymbol{\phi}^{(j)}}. \tag{35}$$

Here $\tilde{\boldsymbol{\phi}}^{(j)}$ is the jth left eigenvector and $\lambda^{(i)}$ denotes the ith eigenvalue. Thus, it is evident that to compute the eigenvector derivative for any design variable we have to know the full set of eigenvalues and eigenvectors for both the left and the right eigenvalue problem. This is prohibitively expensive for most applications. For the Rogers' method, we apply the Schur factorization based method implemented in the function of *numpy.linalg.eig* in the NumPy package [62]. To use *numpy.linalg.eig*, the matrix has to be in the dense format, which may add more computational cost. However, this is the only function that implements the eigendecomposition method for general complex matrix in NumPy or SciPy packages.

We compare the method proposed by Rogers [26] with our proposed adjoint method. We consider the same matrix structure as discussed in Section 4.3. In addition, the function of interest is defined as follows

$$f = \mathbf{c}_{1r}^\intercal \boldsymbol{\phi}_r^{(1)} + \mathbf{c}_{1i}^\intercal \boldsymbol{\phi}_i^{(1)}, \tag{36}$$

where $\mathbf{c}_{1r}, \mathbf{c}_{1i}$ are two randomly generated arrays with each entry satisfying the standard normal distribution. Each entry in the vector is independent and identically distributed.

After running 10 randomly generated cases, we obtain the results shown in Fig. 3. We found that the Rogers method quickly becomes computationally infeasible, demonstrating a quadratic scaling for our problem. The proposed adjoint method is several orders more efficient.

Notice we only record the time for the adjoint equation solution and the eigendecomposition for the Rogers method. We do not record the time to form the total derivative matrix because in practice this matrix shall never be formed explicitly (see Section 4.3). For the Rogers method, it is required to iterate through all the design variables (see Eq. (34)) that can also be computationally expensive when there are numerous design variables.

To summarize, we recommend using the proposed adjoint method for the scalar function derivative computation. However, the fact that the current solver requires using a dense matrix format for the eigendecomposition method may contribute to the large time use of the Rogers method.

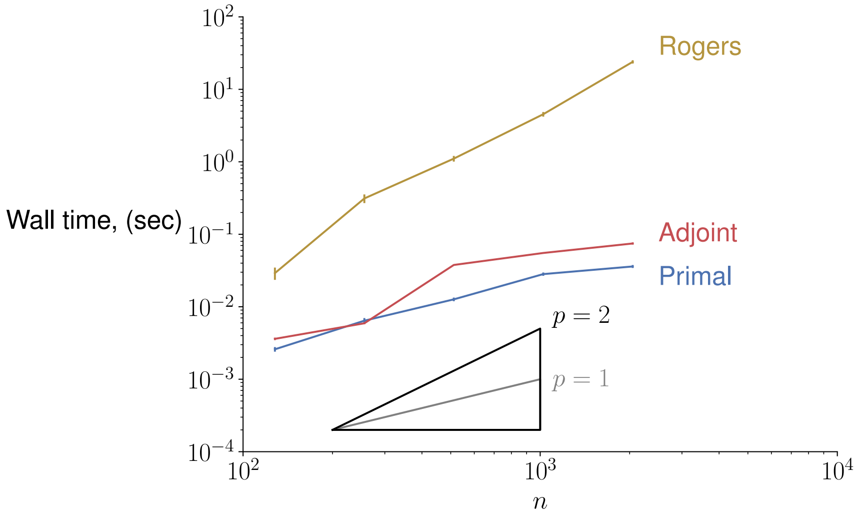

Figure 3: Scalability of the adjoint and the Rogers methods for eigenvector derivative computation. For the Rogers method, only the eigendecomposition time is shown. For the adjoint method, only the adjoint equation solution time is shown. The x-axis, n, is the row (column) number of the matrix. The y-axis, T, is the wall time in second.

4.5 Generalized eigenvalue test case: the plane Poiseuille flow problem

This example is derived from an example in the textbook by Schmid and Henningson [6]. We analyze the eigenvalue and eigenvector derivative for the stability analysis of the plane Poiseuille flow. The linear stability analysis is the foundation of the e^n-based laminar-turbulence transition prediction method proposed by [8]. For the plane Poiseuille flow, the mainstream velocity component U is found to

$$U(y) = U_0(1 - y^2), \tag{37}$$

where U_0 is the dimensionless flow speed at the midpoint of two infinite planes, y is the dimensionless coordinate perpendicular to the flow direction. The velocity profile of the flow is shown in Fig. 4.

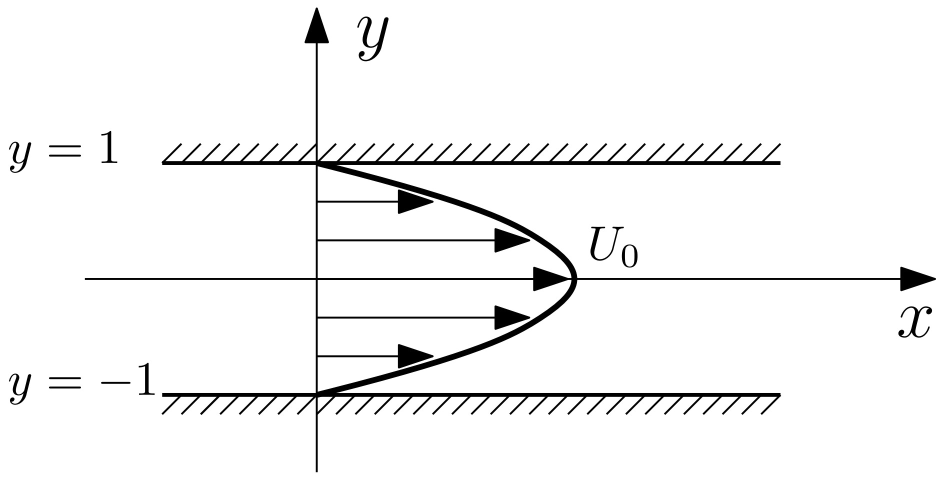

Figure 4: Plane Poiseuille flow

The stability of the flow is captured using Orr–Sommerfeld and Squire's equations. They are defined as

$$\begin{bmatrix} -iL_{\mathrm{OS}} & 0 \\ \beta U' & -iL_{\mathrm{SQ}} \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\eta} \end{bmatrix} = \lambda \begin{bmatrix} k^2 - D^2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\eta} \end{bmatrix}, \tag{38}$$

where the eigenvalue is λ, the eigenvector is composed of vertical velocity perturbation component $\tilde{v}$ and vortex perturbation component $\tilde{\eta}$, D is a differentiation operator for $\partial(\cdot)/\partial y$, $k = \sqrt{\alpha^2 + \beta^2}$ is the wave number, and α and β are wave numbers in x and z directions, respectively. The operators L_{OS} and L_{SQ} are coefficients of the Orr–Sommerfeld and Squire's equations. They are defined as

$$\begin{aligned} L_{\mathrm{OS}} &= i\alpha U(k^2 - D^2) + i\alpha U'' + \frac{1}{\mathrm{Re}} \left(k^2 - D^2\right)^2, \\ L_{\mathrm{SQ}} &= i\alpha U + \frac{1}{\mathrm{Re}} \left(k^2 - D^2\right), \end{aligned} \tag{39}$$

where Re is the Renoylds number. For more details about the equation definition, see [6, Chapter 3].

Following Schmid and Henningson [6, Chapter 3], we discretize Eq. (38) using a spectral collocation method based on Chebyshev polynomials. Then, we can write the eigenvalue problem in the following form

$$\mathbf{K}\boldsymbol{\phi} = \lambda \mathbf{M}\boldsymbol{\phi}, \tag{40}$$

where $\mathbf{K}$ is related with the coefficient matrix the left-hand side of Eq. (38), $\mathbf{M}$ is related with the right-hand side, $\boldsymbol{\phi}$ is the first eigenvector, and λ is the first eigenvalue. For the test case with Re = 10000, $\alpha = 1$, and $\beta = 0$, the eigenvalue distribution of this problem is as plotted in Fig. 5. We want to compute the following derivatives:

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{K}}, \frac{\mathrm{d}f}{\mathrm{d}\mathbf{M}} \tag{41}$$

where we define

$$f = \mathbf{c}_1^\intercal \boldsymbol{\phi} + c_2 \lambda. \tag{42}$$

The constants $\mathbf{c}_1$ and c_2 are set to

$$\mathbf{c}_1 = \begin{bmatrix} 1+i \\ \vdots \\ 1+i \end{bmatrix}, \quad c_2 = 1. \tag{43}$$

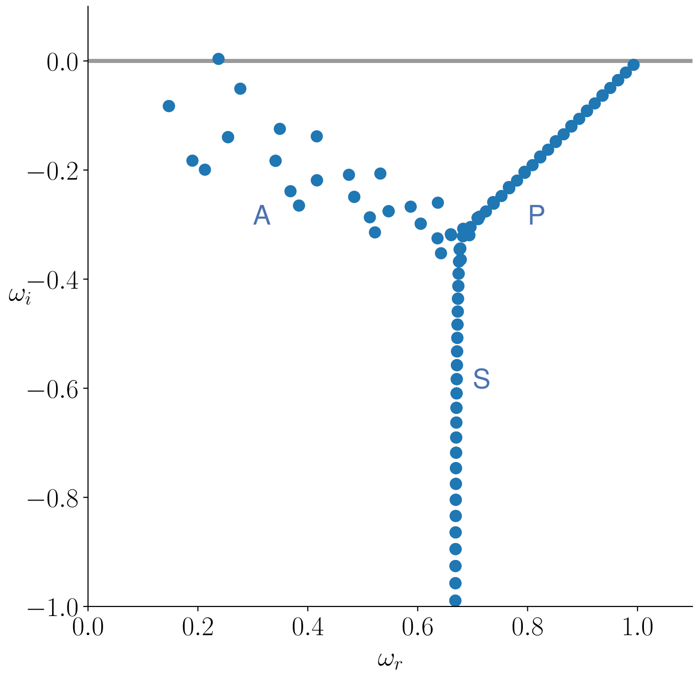

Figure 5: Orr–Sommerfeld spectrum of plane Poiseuille flow for Re $= 10000, \alpha = 1$, and $\beta = 0$. We have $\omega = i\lambda$. The mode at $\omega_r \approx 0.2375$ is slightly unstable. A, P, and S are three branches of the eigenvalues.

Solving for the adjoint variables and computing the total derivative,

$$\begin{aligned}
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{K}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{K}_r}^\intercal \boldsymbol{\psi}_r, & \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{K}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{K}_i}^\intercal \boldsymbol{\psi}_r, & \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{K}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{K}_r}^\intercal \boldsymbol{\psi}_i, & \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{K}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{K}_i}^\intercal \boldsymbol{\psi}_i, \\
\frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{M}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{M}_r}^\intercal \boldsymbol{\psi}_r, & \frac{\mathrm{d}f_r}{\mathrm{d}\mathbf{M}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{M}_i}^\intercal \boldsymbol{\psi}_r, & \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{M}_r} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{M}_r}^\intercal \boldsymbol{\psi}_i, & \frac{\mathrm{d}f_i}{\mathrm{d}\mathbf{M}_i} &= -\frac{\partial \mathbf{r}}{\partial \mathbf{M}_i}^\intercal \boldsymbol{\psi}_i.
\end{aligned} \tag{44}$$

Using the results given in Eq. (23), we compute the total derivatives using the adjoint method.

We compare our adjoint results with FD results for the selected entries of the matrices $\mathbf{K}$ and $\mathbf{M}$ in Table 3. Most of the adjoint results agree with FD results by 5 to 6 digits except for the derivatives close to zero. This verifies our proposed adjoint formulas.

Table 3: Verification of the generalized eigenvalue problem adjoint method for the plane Poiseuille flow problem

Type	Index	Adjoint	FD
$\mathrm{d}f/\mathrm{d}\mathbf{K}_r$	$(1,1)$	$-3.093467\underline{892317424} \times 10^{-03} + 3.15767\underline{5085911619} \times 10^{-03}j$	$-3.093467\underline{371151348} \times 10^{-03} + 3.15768\underline{3238150244} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_r$	$(1,2)$	$\underline{1.747872043767352} \times 10^{-17} + \underline{1.050054351752967} \times 10^{-17}j$	$\underline{3.219646771412954} \times 10^{-09} + \underline{2.116362640691704} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_r$	$(1,3)$	$3.93486\underline{5464062442} \times 10^{-03} - 4.01859\underline{6155412830} \times 10^{-03}j$	$3.93486\underline{8042509976} \times 10^{-03} - 4.01858\underline{7464096196} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_r$	$(1,4)$	$-\underline{2.028157890193728} \times 10^{-17} - \underline{1.482465678767763} \times 10^{-17}j$	$\underline{3.386180225106727} \times 10^{-09} - \underline{3.695394684699593} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_r$	$(1,5)$	$-5.7080\underline{53695355289} \times 10^{-04} + 5.9329\underline{25258108770} \times 10^{-04}j$	$-5.7080\underline{98438361731} \times 10^{-04} + 5.9329\underline{50138121118} \times 10^{-04}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_i$	$(1,1)$	$-3.1576\underline{75085911617} \times 10^{-03} - 3.0934\underline{67892317431} \times 10^{-03}j$	$-3.1576\underline{87333832371} \times 10^{-03} - 3.0934\underline{23122258604} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_i$	$(1,2)$	$-\underline{1.050054351752963} \times 10^{-17} + \underline{1.747872043767353} \times 10^{-17}j$	$\underline{3.441691376337985} \times 10^{-09} + \underline{5.948800479993466} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_i$	$(1,3)$	$4.018596\underline{155412829} \times 10^{-03} + 3.93486\underline{5464062451} \times 10^{-03}j$	$4.018597\underline{954402736} \times 10^{-03} + 3.93487\underline{7967730344} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_i$	$(1,4)$	$\underline{1.482465678767760} \times 10^{-17} - \underline{2.028157890193729} \times 10^{-17}j$	$\underline{3.552713678800501} \times 10^{-09} + \underline{1.875669758399923} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{K}_i$	$(1,5)$	$-5.9329\underline{25258108768} \times 10^{-04} - 5.70\underline{8053695355301} \times 10^{-04}j$	$-5.9329\underline{3192352462} \times 10^{-04} - 5.70\underline{7966022582001} \times 10^{-04}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_r$	$(1,1)$	$1.210393\underline{515416074} \times 10^{-03} - 1.24939\underline{5427944639} \times 10^{-03}j$	$1.210393\underline{618666927} \times 10^{-03} - 1.24939\underline{0839917763} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_r$	$(1,2)$	$-\underline{6.901243089506205} \times 10^{-18} - \underline{4.093735634953344} \times 10^{-18}j$	$\underline{4.551914400963142} \times 10^{-09} + \underline{1.485660552913437} \times 10^{-8}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_r$	$(1,3)$	$-1.5396\underline{05972225761} \times 10^{-03} + \underline{1.5900}31123181217 \times 10^{-03}j$	$-1.5396\underline{10106071621} \times 10^{-03} + \underline{1.5900}65942195379 \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_r$	$(1,4)$	$\underline{8.013716233205633} \times 10^{-18} + \underline{5.789188448385005} \times 10^{-18}j$	$\underline{6.106226635438361} \times 10^{-10} + \underline{3.210539473164076} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_r$	$(1,5)$	$2.2331\underline{79146721826} \times 10^{-04} - 2.347\underline{247990953623} \times 10^{-04}j$	$2.2331\underline{60878439833} \times 10^{-04} - 2.347\underline{071109513876} \times 10^{-04}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_i$	$(1,1)$	$1.24939\underline{5427944638} \times 10^{-03} + 1.21039\underline{3515416076} \times 10^{-03}j$	$1.24939\underline{8362546117} \times 10^{-03} + 1.21039\underline{1674041911} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_i$	$(1,2)$	$\underline{4.093735634953333} \times 10^{-18} - \underline{6.901243089506210} \times 10^{-18}j$	$-\underline{1.665334536937735} \times 10^{-10} - \underline{3.721415536839245} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_i$	$(1,3)$	$-1.5900\underline{31123181217} \times 10^{-03} - 1.53960\underline{5972225765} \times 10^{-03}j$	$-1.5900\underline{56253331085} \times 10^{-03} - 1.53960\underline{6692135820} \times 10^{-03}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_i$	$(1,4)$	$-\underline{5.789188448384990} \times 10^{-18} + \underline{8.013716233205639} \times 10^{-18}j$	$\underline{2.220446049250313} \times 10^{-09} + \underline{8.922550198686707} \times 10^{-09}j$
$\mathrm{d}f/\mathrm{d}\mathbf{M}_i$	$(1,5)$	$2.34724\underline{7990953622} \times 10^{-04} + 2.233\underline{179146721831} \times 10^{-04}j$	$2.34724\underline{6286227289} \times 10^{-04} + 2.233\underline{274962529230} \times 10^{-04}j$

5 Conclusion

In this paper, we developed an adjoint method for complex standard eigenvalue and generalized eigenvalue problems and a RAD formula for the eigenvalue derivative. The proposed adjoint method requires fewer function evaluations than the forward methods, such as FD and the direct methods, for problems with more design variables than functions of interest. This is a critical advantage for PDE-constrained gradient-based optimization with the eigenvalue or the eigenvector, where there are usually more design variables than functions of interest. One potential application is the aerodynamic shape optimization with transition modeled using the e^N method, which requires differentiating a generalized eigenvalue problem for derivative computation. We compared the proposed derivative methods with FD approximations. We achieved a 5 to 6 digit match of these two methods for both eigenvalue and generalized eigenvalue problems.

6 Acknowledgment

The first author wants to thank Huajie Qian for helpful discussions on the proof in A.

References

[1] K.-J. Bathe, Finite element procedures, Klaus-Jurgen Bathe, 2006.

[2] W. Su, C. E. S. Cesnik, Strain-based analysis for geometrically nonlinear beams: A modal approach, Journal of Aircraft 51 (2014) 890–903. doi:10.2514/1.c032477.

[3] P. R. Sudalagunta, C. Sultan, R. K. Kapania, L. T. Watson, P. Raj, Accurate computing of higher vibration modes of thin flexible structures, AIAA Journal 54 (2016) 1704–1718. doi:10.2514/1.j054428.

[4] E. Jonsson, C. Riso, C. A. Lupp, C. E. S. Cesnik, J. R. R. A. Martins, B. I. Epureanu, Flutter and post-flutter constraints in aircraft design optimization, Progress in Aerospace Sciences 109 (2019) 100537. doi:10.1016/j.paerosci.2019.04.001.

[5] A. Drachinsky, D. E. Raveh, Modal rotations: A modal-based method for large structural deformations of slender bodies, AIAA Journal (2020) 1–15. doi:10.2514/1.j058899.

[6] P. J. Schmid, D. S. Henningson, Stability and Transition in Shear Flows, Springer New York, 2001. doi:10.1007/978-1-4613-0185-1.

[7] J. Driver, D. W. Zingg, Numerical aerodynamic optimization incorporating laminar-turbulent transition prediction, AIAA Journal 45 (2007) 1810–1818. doi:10.2514/1.23569.

[8] J. L. van Ingen, A Suggested Semi-Empirical Method for the Calculation of the Boundary Layer Transition Region, Univ. Delft Report VTH-74, University of Delft, Delft, The Nederlands, 1956.

[9] Y. Shi, R. Gross, C. A. Mader, J. R. R. A. Martins, Transition prediction based on linear stability theory with the RANS solver for three-dimensional configurations, in: Proceedings of the AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, Kissimmee, FL, 2018. doi:10.2514/6.2018-0819.

[10] Y. Shi, C. A. Mader, S. He, G. L. O. Halila, J. R. R. A. Martins, Natural laminar-flow airfoil optimization design using a discrete adjoint approach, AIAA Journal 58 (2020) 4702–4722. doi:10.2514/1.J058944.

[11] S. Xu, S. Timme, K. J. Badcock, Enabling off-design linearised aerodynamics analysis using Krylov subspace recycling technique, Computers & Fluids 140 (2016) 385–396. doi:10.1016/j.compfluid.2016.10.018.

[12] S. Timme, Global instability of wing shock-buffet onset, Journal of Fluid Mechanics 885 (2020) A37. doi:10.1017/jfm.2019.1001.

[13] B. Emerson, T. Lieuwen, M. P. Juniper, Local stability analysis and eigenvalue sensitivity of reacting bluff-body wakes, Journal of Fluid Mechanics 788 (2016) 549–575. doi:10.1017/jfm.2015.724.

[14] A. C. Madden, M. P. Castanier, B. I. Epureanu, Mistuning identification of blisks at higher frequencies, AIAA Journal 49 (2011) 1299–1302. doi:10.2514/1.j050427.

[15] W. Tang, S. Baek, B. I. Epureanu, Reduced-order models for blisks with small and large mistuning and friction dampers, Journal of Engineering for Gas Turbines and Power 139 (2016). doi:10.1115/1.4034212.

[16] J. A. Beck, J. M. Brown, O. E. Scott-Emuakpor, E. B. Carper, A. A. Kaszynski, Modal expansion method for eigensensitivity calculations of cyclically symmetric bladed disks, AIAA Journal 56 (2018) 4112–4120. doi:10.2514/1.j057322.

[17] R. M. Lin, T. Y. Ng, Prediction of mistuning effect of bladed disks using eigensensitivity analysis, Engineering Structures 212 (2020) 110416. doi:10.1016/j.engstruct.2020.110416.

[18] J.-J. E. Slotine, W. Li, Applied nonlinear control, Prentice Hall, Englewood Cliffs, N.J, 1991.

[19] R. M. Lin, M. K. Lim, Complex eigensensitivity-based characterization of structures with viscoelastic damping, The Journal of the Acoustical Society of America 100 (1996) 3182–3191. doi:10.1121/1.417202.

[20] R. M. Lin, T. Y. Ng, Frequency response functions and modal analysis of general nonviscously damped dynamic systems with and without repeated modes, Mechanical Systems and Signal Processing 120 (2019) 744–764. doi:10.1016/j.ymssp.2018.10.032.

[21] F. Tisseur, K. Meerbergen, The quadratic eigenvalue problem, SIAM Review 43 (2001) 235–286. doi:10.1137/s0036144500381988.

[22] S. Güttel, F. Tisseur, The nonlinear eigenvalue problem, Acta Numerica 26 (2017) 1–94. doi:10.1017/s0962492917000034.

[23] E. Jonsson, G. K. W. Kenway, G. J. Kennedy, J. R. R. A. Martins, Development of flutter constraints for high-fidelity aerostructural optimization, in: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Denver, CO, 2017. AIAA 2017-4455.

[24] E. Jonsson, C. A. Mader, G. J. Kennedy, J. R. R. A. Martins, Computational modeling of flutter constraint for high-fidelity aerostructural optimization, in: 2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, San Diego, CA, 2019. doi:10.2514/6.2019-2354.

[25] R. L. Fox, M. P. Kapoor, Rates of change of eigenvalues and eigenvectors., AIAA Journal 6 (1968) 2426–2429. doi:10.2514/3.5008.

[26] L. C. Rogers, Derivatives of eigenvalues and eigenvectors, AIAA Journal 8 (1970) 943–944. doi:10.2514/3.5795.

[27] C. S. Rudisill, Y.-Y. Chu, Numerical methods for evaluating the derivatives of eigenvalues and eigenvectors, AIAA Journal 13 (1975) 834–837. doi:10.2514/3.60449.

[28] R. B. Nelson, Simplified calculation of eigenvector derivatives, AIAA Journal 14 (1976) 1201–1205. doi:10.2514/3.7211.

[29] K. B. Lim, J. L. Junkins, B. P. Wang, Re-examination of eigenvector derivatives, Journal of Guidance, Control, and Dynamics 10 (1987) 581–587. doi:10.2514/3.20259.

[30] B. P. Wang, Improved approximate methods for computing eigenvector derivatives in structural dynamics, AIAA Journal 29 (1991) 1018–1020. doi:10.2514/3.59945.

[31] 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, Mechanical Systems and Signal Processing 138 (2020) 106536. doi:10.1016/j.ymssp.2019.106536.

[32] J. R. R. A. Martins, A. Ning, Engineering Design Optimization, Cambridge University Press, 2021. URL: https://mdobook.github.io. doi:10.1017/9781108980647.

[33] J. R. R. A. Martins, J. T. Hwang, Review and unification of methods for computing derivatives of multidisciplinary computational models, AIAA Journal 51 (2013) 2582–2599. doi:10.2514/1.J052184.

[34] J. R. R. A. Martins, P. Sturdza, J. J. Alonso, The complex-step derivative approximation, ACM Transactions on Mathematical Software 29 (2003) 245–262. doi:10.1145/838250.838251.

[35] 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.

[36] L. Hascoët, V. Pascual, TAPENADE 2.1 User's Guide, Technical report 300, INRIA, 2004. URL: https://hal.inria.fr/inria-00069880/document.

[37] P. S. Dwyer, M. S. Macphail, Symbolic matrix derivatives, The Annals of Mathematical Statistics 19 (1948) 517–534.

[38] M. Giles, An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation, 2008. URL: https://people.maths.ox.ac.uk/gilesm/files/NA-08-01.pdf.

[39] N. P. Bons, X. He, C. A. Mader, J. R. R. A. Martins, Multimodality in aerodynamic wing design optimization, AIAA Journal 57 (2019) 1004–1018. doi:10.2514/1.J057294.

[40] M. Mangano, S. He, Y. Liao, D.-G. Caprace, J. R. R. A. Martins, Towards passive aeroelastic tailoring of large wind turbines using high-fidelity multidisciplinary design optimization, in: AIAA SciTech Forum, 2022. doi:10.2514/6.2022-1289.

[41] Y. Liao, J. R. R. A. Martins, Y. L. Young, 3-D high-fidelity hydrostructural optimization of cavitation-free composite lifting surfaces, Composite Structures 268 (2021) 113937. doi:10.1016/j.compstruct.2021.113937.

[42] T. H. Lee, Adjoint method for design sensitivity analysis of multiple eigenvalues and associated eigenvectors, AIAA Journal 45 (2007) 1998–2004. doi:10.2514/1.25347.

[43] S. He, E. Jonsson, J. R. R. A. Martins, Derivatives for eigenvalues and eigenvectors via analytic reverse algorithmic differentiation, AIAA Journal (2022). doi:10.2514/1.J060726, (In press).

[44] H. Kim, M. Cho, Study on the design sensitivity analysis based on complex variable in eigenvalue problem, Finite Elements in Analysis and Design 45 (2009) 892–900. doi:10.1016/j.finel.2009.07.002.

[45] R. Lewandowski, M. Łasecka-Plura, Design sensitivity analysis of structures with viscoelastic dampers, Computers & Structures 164 (2016) 95–107. doi:10.1016/j.compstruc.2015.11.011.

[46] G. H. Yoon, A. Donoso, J. C. Bellido, D. Ruiz, Highly efficient general method for sensitivity analysis of eigenvectors with repeated eigenvalues without passing through adjacent eigenvectors, International Journal for Numerical Methods in Engineering (2020). doi:10.1002/nme.6442.

[47] G. van der Veen, M. Langelaar, S. van der Meulen, D. Laro, W. Aangenent, F. van Keulen, Integrating topology optimization in precision motion system design for optimal closed-loop control performance, Mechatronics 47 (2017) 1–13. doi:10.1016/j.mechatronics.2017.06.003.

[48] B. Morgan, Computational procedure for the sensitivity of an eigenvalue, Electronics Letters 2 (1966) 197. doi:10.1049/el:19660166.

[49] D. Reddy, Evaluation of the sensitivity coefficient of an eigenvalue, IEEE Transactions on Automatic Control 12 (1967) 792–792. doi:10.1109/tac.1967.1098773.

[50] D. V. Murthy, R. T. Haftka, Derivatives of eigenvalues and eigenvectors of a general complex matrix, International Journal for Numerical Methods in Engineering 26 (1988) 293–311. doi:10.1002/nme.1620260202.

[51] S. F. Walter, Efficient higher order derivatives of objective functions composed of matrix operations, 2009. doi:10.48550/ARXIV.0911.4940.

[52] M. Seeger, A. Hetzel, Z. Dai, E. Meissner, N. D. Lawrence, Auto-differentiating linear algebra, 2017. doi:10.48550/ARXIV.1710.08717.

[53] P. Peltzer, J. Lotz, U. Naumann, Eigen-AD: Algorithmic differentiation of the eigen library, in: Lecture Notes in Computer Science, Springer International Publishing, 2020, pp. 690–704. doi:10.1007/978-3-030-50371-0_51.

[54] D. A. O. 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. doi:10.48550/ARXIV.2009.10071.

[55] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, Journal of Research of the National Bureau of Standards 45 (1950) 255. doi:10.6028/jres.045.026.

[56] L. Hascoet, V. Pascual, The Tapenade automatic differentiation tool: Principles, model, and specification, ACM Transactions on Mathematical Software 39 (2013) 20:1–20:43. doi:10.1145/2450153.2450158.

[57] J. R. Magnus, On differentiating eigenvalues and eigenvectors, Econometric Theory 1 (1985) 179–191. doi:10.1017/S0266466600011129.

[58] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869. doi:10.1137/0907058.

[59] 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. J. 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 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[60] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Society for Industrial and Applied Mathematics, 2011. doi:10.1137/1.9781611970739.

[61] H. V. der Vorst, C. Vuik, The superlinear convergence behaviour of GMRES, Journal of Computational and Applied Mathematics 48 (1993) 327–341. doi:10.1016/0377-0427(93)90028-a.

[62] S. van der Walt, S. C. Colbert, G. Varoquaux, The NumPy array: A structure for efficient numerical computation, Computing in Science & Engineering 13 (2011) 22–30. doi:10.1109/mcse.2011.37.

[63] D. R. Brillinger, The analyticity of the roots of a polynomial as functions of the coefficients, Mathematics Magazine 39 (1966) 145–147. doi:10.1080/0025570x.1966.11975702.

[64] T. P. Minka, Old and new matrix algebra useful for statistics (2000).

A Proof of that the eigenvalue as a function of the matrix is analytic

Theorem 1 *For complex matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ with distinctive eigenvalues, $\lambda_i \neq \lambda_j, \forall i, j, i \neq j$, we have $\lambda_i, \forall i$, is an analytic function of $\mathbf{A}$.*

Proof: First, we construct the characteristic polynomial, $p(x)$, of a matrix, $\mathbf{A}$,

$$p(x) = x^n + c_{n-1}(\mathbf{A})x^{n-1} + \cdots + c_0(\mathbf{A}), \tag{45}$$

where $c_i(\mathbf{A}) \in \mathbb{C}$, $i = 1, \ldots, n-1$, are coefficients dependent on the matrix $\mathbf{A}$. The eigenvalues are the roots of the characteristic polynomial, $p(x) = 0$.

It can also be shown that the coefficients, $c_i(\mathbf{A}) \in \mathbb{C}$, $i = 1, \ldots, n-1$, are analytic function of the coefficient matrix $\mathbf{A}$. Thus, by the composition property of the analytic function, we only need to show that any root is an analytic function of the coefficients,

c_i, $i = 1, \dots, n-1$, and then, we know that any root is indeed an analytic function of the matrix $\mathbf{A}$.

We apply the following lemma.

Lemma 2 *(Brillinger [63]) The distinct roots of an n-th degree complex polynomial are analytic functions of the coefficients in the region where the roots retain their various multiplicities.*

As long as we show that the roots retain their multiplicities, then we know it is analytic.

Now we show the roots retain their multiplicities. It can be shown that the roots are continuous function of the coefficients, c_i, $i = 1, \dots, n-1$. Because c_i, $i = 1, \dots, n-1$, is also continuous with respect to $\mathbf{A}$. It follows that the roots are continuous with respect to $\mathbf{A}$. We measure the minimum distance between the distinct roots

$$\epsilon_0 = \min_{i,j,i\neq j} |\lambda_i - \lambda_j|, \tag{46}$$

where $\epsilon_0 > 0$ because the assumption of all distintive eigenvalues. Then, due to continuity, with $\epsilon = \epsilon_0/4$, we can pick a positive scalar, δ, such that, $\forall \hat{\mathbf{A}} \in \mathbb{C}^{n\times n}$, $|\hat{\mathbf{A}} - \mathbf{A}| < \delta$, we have

$$|\hat{\lambda}_i - \lambda_i| \leq \epsilon, i = 1, \dots, n, \tag{47}$$

where $\hat{\mathbf{A}}$ is a perturbed coefficient matrix, and $\hat{\lambda}_i$ are the corresponding perturbed eigenvalues. By construction, we have

$$\hat{\lambda}_i \neq \hat{\lambda}_j, \forall i, j, i \neq j. \tag{48}$$

This is because if there is indeed a pair, $i \neq j$, such that $\hat{\lambda}_i = \hat{\lambda}_j$, we then have

$$\begin{aligned} & |\lambda_i - \lambda_j| \\ =& |(\lambda_i - \hat{\lambda}_i) - (\lambda_j - \hat{\lambda}_j) + (\hat{\lambda}_i - \hat{\lambda}_j)| \\ =& |(\lambda_i - \hat{\lambda}_i) - (\lambda_j - \hat{\lambda}_j)| \\ \leq& |(\lambda_i - \hat{\lambda}_i)| + |(\lambda_j - \hat{\lambda}_j)| \\ \leq& \frac{\epsilon_0}{4} + \frac{\epsilon_0}{4} = \frac{\epsilon_0}{2} < \epsilon_0. \end{aligned} \tag{49}$$

This is a contradiction

$$|\lambda_i - \lambda_j| < \epsilon_0 = \min_{i,j,i\neq j} |\lambda_i - \lambda_j|. \tag{50}$$

Thus, in a small neighborhood of $\mathbf{A}$ specified by δ, the eigenvalues retain their multiplicities, in this case, one for all the eigenvalues. This finishes the proof that any eigenvalue is an analytic function of the coefficient matrix, $\mathbf{A}$. □

B Notation conventions

The vectorization operator vec $(\cdot)$ is defined as follows,

$$(\text{vec}\,(\mathbf{A}))_{i\times(n_2-1)+j} = \mathbf{A}_{ij}, \quad i = 1, \dots, n_1, \quad j = 1, \dots, n_2, \tag{51}$$

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

$$\text{vec}^{-1}\,(\text{vec}\,(\mathbf{A})) = \mathbf{A}, \tag{52}$$

for arbitrary matrix $\mathbf{A}$. As a special case, when the vectorization operator or the inverse vectorization operator is operating on a vector, we obtain the vector itself, i.e.,

$$\begin{aligned} \text{vec}\,(\mathbf{a}) &= \mathbf{a}, \\ \text{vec}^{-1}\,(\mathbf{a}) &= \mathbf{a}, \end{aligned} \tag{53}$$

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

The following convention is used when writing a derivative involving matrices in this paper. For $(\partial \mathbf{A}/\partial \mathbf{B})^\intercal \, \overline{\mathbf{A}}$, 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, $(\partial \mathbf{A}/\partial \mathbf{B})^\intercal \, \overline{\mathbf{A}}$ is a simplified notation of the following operation,

$$\text{vec}^{-1}\left(\left(\frac{\partial \text{vec}\,(\mathbf{A})}{\partial \text{vec}\,(\mathbf{B})}\right)^\intercal \text{vec}\,\left(\overline{\mathbf{A}}\right)\right) \in \mathbb{R}^{n_2 \times n_3}. \tag{54}$$

C Trace identities

The following matrix trace identities are used in the remainder of the appendices [38, 64]:

$$\begin{aligned} \text{Tr}(\mathbf{AB}) &= \text{Tr}(\mathbf{BA}) \\ \text{Tr}(\mathbf{A} + \mathbf{B}) &= \text{Tr}(\mathbf{A}) + \text{Tr}(\mathbf{B}). \end{aligned} \tag{55}$$

D Dot product identity for an analytic complex function

It is well-known for a real function that the following dot product identity holds for function $\mathbf{f}(\mathbf{w})$ [38]

$$\text{Tr}(\bar{\mathbf{f}}^\intercal \dot{\mathbf{f}}) = \text{Tr}(\overline{\mathbf{w}}^\intercal \dot{\mathbf{w}}). \tag{56}$$

We find that, for a complex analytic function, i.e., $\mathbf{f}(\mathbf{w})$ is analytic, $\mathbf{w} \in \mathbb{C}^{m_\mathbf{w}}, \mathbf{f} \in \mathbb{C}^{m_\mathbf{f}}$, where $m_\mathbf{w}, m_\mathbf{f}$ are the dimensions of the vectors, similar results hold

$$\text{Tr}(\bar{\mathbf{f}}^* \dot{\mathbf{f}}) = \text{Tr}(\overline{\mathbf{w}}^* \dot{\mathbf{w}}). \tag{57}$$

In this case, we apply a conjugate transpose instead of a transpose on the reverse seed. We will prove this result.

We first expand the seeds of $\mathbf{f}$ and $\mathbf{w}$ into real and imaginary parts. Here, the "$\dot{\square}$" specifies the accumulated derivative in a FAD mode, and "$\overline{\square}$" specify it in a RAD mode. For the conventions of AD, such as seeds, we refer the reader to [38, 32, 43].

$$\begin{aligned} \dot{\mathbf{f}} &= \dot{\mathbf{f}}_r + i\dot{\mathbf{f}}_i, \quad \dot{\mathbf{w}} = \dot{\mathbf{w}}_r + i\dot{\mathbf{w}}_i, \\ \bar{\mathbf{f}} &= \bar{\mathbf{f}}_r + i\bar{\mathbf{f}}_i, \quad \overline{\mathbf{w}} = \overline{\mathbf{w}}_r + i\overline{\mathbf{w}}_i. \end{aligned} \tag{58}$$

Then, the LHS of Eq. (57) can be written as

$$\begin{aligned}
\text{(LHS)}\ &\mathrm{Tr}\left(\bar{\mathbf{f}}_r^\intercal\dot{\mathbf{f}}_r+\bar{\mathbf{f}}_i^\intercal\dot{\mathbf{f}}_i\right)+i\mathrm{Tr}\left(-\bar{\mathbf{f}}_i^\intercal\dot{\mathbf{f}}_r+\bar{\mathbf{f}}_r^\intercal\dot{\mathbf{f}}_i\right)\\
=&\mathrm{Tr}\left(\bar{\mathbf{f}}_r^\intercal\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r+\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_i}\dot{\mathbf{w}}_i\right)+\bar{\mathbf{f}}_i^\intercal\left(\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_i}\dot{\mathbf{w}}_i\right)\right)\\
&+i\mathrm{Tr}\left(-\bar{\mathbf{f}}_i^\intercal\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r+\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_i}\dot{\mathbf{w}}_i\right)+\bar{\mathbf{f}}_r^\intercal\left(\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_i}\dot{\mathbf{w}}_i\right)\right).
\end{aligned}\tag{59}$$

Going from the first equation to the second, we use the definition of the forward seeds.

Then, for the RHS, similarly, we have

$$\begin{aligned}
\text{(RHS)}\ &\mathrm{Tr}\left(\overline{\mathbf{w}}_r^\intercal\dot{\mathbf{w}}_r+\overline{\mathbf{w}}_i^\intercal\dot{\mathbf{w}}_i\right)+i\mathrm{Tr}\left(-\overline{\mathbf{w}}_i^\intercal\dot{\mathbf{w}}_r+\overline{\mathbf{w}}_r^\intercal\dot{\mathbf{w}}_i\right)\\
=&\mathrm{Tr}\left(\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_r+\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_i}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_i}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_i\right)\\
&+i\mathrm{Tr}\left(-\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_i}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_i}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_r+\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_i\right).
\end{aligned}\tag{60}$$

Now, we show that the LHS and the RHS are actually equal to each other, which proves the theorem. The real parts of LHS and RHS are apparently equal. For the imaginary parts, we need to apply the Cauchy–Riemann condition that is satisfied because the function is analytic,

$$\begin{aligned}
\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}&=\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_i},\\
\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_i}&=-\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}.
\end{aligned}\tag{61}$$

Using the Cauchy–Riemann condition, we can convert all the partial derivatives in the imaginary parts for LHS and RHS.

$$\begin{aligned}
(\mathrm{Im}\,(\mathrm{LHS}))\ &\mathrm{Tr}\left(-\bar{\mathbf{f}}_i^\intercal\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r-\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_i\right)+\bar{\mathbf{f}}_r^\intercal\left(\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_r+\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}\dot{\mathbf{w}}_i\right)\right)\\
(\mathrm{Im}\,(\mathrm{RHS}))\ &\mathrm{Tr}\left(-\left(-\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_r+\left(\frac{\partial\mathbf{f}_r}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_r+\frac{\partial\mathbf{f}_i}{\partial\mathbf{w}_r}^\intercal\overline{\mathbf{f}}_i\right)^\intercal\dot{\mathbf{w}}_i\right).
\end{aligned}\tag{62}$$

They are equal. Thus, we conclude that Eq. (57) holds.

Roberts and Roberts [54] proposed the following identity for complex functions:

$$\mathrm{Tr}(\bar{\mathbf{f}}^*\dot{\mathbf{f}}+(\bar{\mathbf{f}}^*\dot{\mathbf{f}})^*)=\mathrm{Tr}(\overline{\mathbf{w}}^*\dot{\mathbf{w}}+(\overline{\mathbf{w}}^*\dot{\mathbf{w}})^*).\tag{63}$$

We can derive this identity using Eq. (57) such that,

$$\begin{aligned}
\mathrm{Tr}(\bar{\mathbf{f}}^*\dot{\mathbf{f}})&=\mathrm{Tr}(\overline{\mathbf{w}}^*\dot{\mathbf{w}}),\\
\mathrm{Tr}((\bar{\mathbf{f}}^*\dot{\mathbf{f}})^*)&=\mathrm{Tr}((\overline{\mathbf{w}}^*\dot{\mathbf{w}})^*).
\end{aligned}\tag{64}$$

Then, by adding up these two equations, we obtain the identity Eq. (63) proposed by Roberts and Roberts [54].

E Derivation of Eq. (15)

First, we present the derivation of the FAD formula. The classic complex eigenvalue derivative found by Magnus [57], as a direct extension of its real counterpart [25], can be derived as follows

$$\dot{\mathbf{A}}\boldsymbol{\phi} + \mathbf{A}\dot{\boldsymbol{\phi}} = \dot{\lambda}\boldsymbol{\phi} + \lambda\dot{\boldsymbol{\phi}}. \tag{65}$$

For more details of the seed definition in the context of AD, we refer the reader to [32, 43]. Then, we premultiply Eq. (65) with its corresponding conjugate transpose left eigenvector $\tilde{\boldsymbol{\phi}}$ defined by Eq. (14). We have

$$\begin{aligned} \tilde{\boldsymbol{\phi}}^*\dot{\mathbf{A}}\boldsymbol{\phi} + \tilde{\boldsymbol{\phi}}^*\mathbf{A}\dot{\boldsymbol{\phi}} &= \dot{\lambda}\tilde{\boldsymbol{\phi}}^*\boldsymbol{\phi} + \lambda\tilde{\boldsymbol{\phi}}^*\dot{\boldsymbol{\phi}}, \\ \Rightarrow \tilde{\boldsymbol{\phi}}^*\dot{\mathbf{A}}\boldsymbol{\phi} + \lambda\tilde{\boldsymbol{\phi}}^*\dot{\boldsymbol{\phi}} &= \dot{\lambda}\tilde{\boldsymbol{\phi}}^*\boldsymbol{\phi} + \lambda\tilde{\boldsymbol{\phi}}^*\dot{\boldsymbol{\phi}}, \end{aligned} \tag{66}$$

where going from the first to the second equation, we apply Eq. (14). Canceling identical terms from both sides, we have

$$\dot{\lambda} = \frac{\tilde{\boldsymbol{\phi}}^*\dot{\mathbf{A}}\boldsymbol{\phi}}{\tilde{\boldsymbol{\phi}}^*\boldsymbol{\phi}}. \tag{67}$$

Next, we derive the RAD formula using the proposed complex dot product identity shown in C. Using Eq. (57), we have

$$\operatorname{Tr}\left(\overline{\mathbf{A}}^*\dot{\mathbf{A}}\right) = \operatorname{Tr}\left(\bar{\lambda}^*\dot{\lambda}\right). \tag{68}$$

Inserting Eq. (67) into Eq. (68), and using the second trace identity from Eq. (55) we have,

$$\operatorname{Tr}\left(\overline{\mathbf{A}}^*\dot{\mathbf{A}}\right) = \operatorname{Tr}\left(\bar{\lambda}^*\frac{\tilde{\boldsymbol{\phi}}^*\dot{\mathbf{A}}\boldsymbol{\phi}}{\tilde{\boldsymbol{\phi}}^*\boldsymbol{\phi}}\right) = \operatorname{Tr}\left(\frac{\bar{\lambda}^*}{\tilde{\boldsymbol{\phi}}^*\boldsymbol{\phi}}\boldsymbol{\phi}\tilde{\boldsymbol{\phi}}^*\dot{\mathbf{A}}\right). \tag{69}$$

Since the equation mush hold for arbitrary $\dot{\mathbf{A}}$ we have,

$$\overline{\mathbf{A}} = \frac{\bar{\lambda}}{\boldsymbol{\phi}^*\tilde{\boldsymbol{\phi}}}\tilde{\boldsymbol{\phi}}\boldsymbol{\phi}^*. \tag{70}$$

Thus, to obtain the derivative of the real part of λ, i.e., $\mathrm{d}\lambda_r/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}\lambda_r/\mathrm{d}\mathbf{A}_i$, we simply seed $\bar{\lambda} = 1$. We then have the following results

$$\begin{aligned} \frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r} &= \operatorname{Re}\left(\frac{\tilde{\boldsymbol{\phi}}\boldsymbol{\phi}^*}{\boldsymbol{\phi}^*\tilde{\boldsymbol{\phi}}}\right), \\ \frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i} &= \operatorname{Im}\left(\frac{\tilde{\boldsymbol{\phi}}\boldsymbol{\phi}^*}{\boldsymbol{\phi}^*\tilde{\boldsymbol{\phi}}}\right), \end{aligned} \tag{71}$$

The remaining two partial derivatives of the imaginary parts, $\mathrm{d}\lambda_i/\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}\lambda_i/\mathrm{d}\mathbf{A}_i$, can be obtained using the Cauchy–Riemann condition (see Eq. (61))

$$\begin{aligned} \frac{\mathrm{d}\lambda_i}{\mathrm{d}\mathbf{A}_r} &= -\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i}, \\ \frac{\mathrm{d}\lambda_i}{\mathrm{d}\mathbf{A}_i} &= \frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r}. \end{aligned} \tag{72}$$

Equation (71) then fully determines the derivatives.

Notice that the expression is independent of normalization condition of both $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{\phi}}$. For example, if some other normalization condition is applied, the right and left eigenvectors are scaled and rotated to the following new eigenvectors

$$\begin{aligned}\boldsymbol{\phi} &= \boldsymbol{\phi}_0 \alpha_r e^{i\theta_r},\\ \tilde{\boldsymbol{\phi}} &= \tilde{\boldsymbol{\phi}}_0 \alpha_l e^{i\theta_l}.\end{aligned} \tag{73}$$

Here, $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_0$ are the new and original right eigenvectors, respectively, $\tilde{\boldsymbol{\phi}}$ and $\tilde{\boldsymbol{\phi}}_0$ are the new and original left eigenvectors, respectively, α_r and α_l are scaling factors for the right and left eigenvector, respectively, and θ_r and θ_l are rotation angles for the right and left eigenvector, respectively. Inserting Eq. (73) into Eq. (71), we have

$$\begin{aligned}\frac{\mathrm{d}\lambda}{\mathrm{d}\mathbf{A}} &= \frac{\tilde{\boldsymbol{\phi}}\boldsymbol{\phi}^*}{\boldsymbol{\phi}^*\tilde{\boldsymbol{\phi}}}\\ &= \frac{\alpha_l\alpha_r e^{i\theta_l} e^{-i\theta_r} \tilde{\boldsymbol{\phi}}_0 \boldsymbol{\phi}_0^*}{\alpha_r\alpha_l e^{-i\theta_r} e^{i\theta_l} \boldsymbol{\phi}_0^* \tilde{\boldsymbol{\phi}}_0}\\ &= \frac{\tilde{\boldsymbol{\phi}}_0 \boldsymbol{\phi}_0^*}{\boldsymbol{\phi}_0^* \tilde{\boldsymbol{\phi}}_0}.\end{aligned} \tag{74}$$

Thus, the result is independent of the normalization condition.

F Relation between Eq. (10) and Eq. (15)

The derivative of $\mathrm{d}\lambda_r/\,\mathrm{d}\mathbf{A}_r$ and $\mathrm{d}\lambda_r/\,\mathrm{d}\mathbf{A}_i$ can be obtained by setting

$$f = \lambda_r, \tag{75}$$

and solve Eq. (10). The solution of the adjoint equation is found to be

$$\boldsymbol{\psi} = \begin{bmatrix}\boldsymbol{\psi}_{\mathrm{main},r}\\ \boldsymbol{\psi}_{\mathrm{main},i}\\ \boldsymbol{\psi}_m\\ \boldsymbol{\psi}_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}_r\\ \mathbf{u}_i\\ 0\\ 0\end{bmatrix}, \tag{76}$$

where $\mathbf{u} = \mathbf{u}_r + i\mathbf{u}_i$ is a left eigenvector and satisfies the following normalization condition

$$\boldsymbol{\phi}^*\mathbf{u} = -1. \tag{77}$$

By applying the adjoint solution $\boldsymbol{\psi}$ in Eq. (12) and the respective result in Eq. (8) we obtain,

$$\begin{aligned}\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r} &= -\mathbf{u}_r\boldsymbol{\phi}_r^\intercal - \mathbf{u}_i\boldsymbol{\phi}_i^\intercal\\ \frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i} &= \mathbf{u}_r\boldsymbol{\phi}_i^\intercal - \mathbf{u}_i\boldsymbol{\phi}_r^\intercal.\end{aligned} \tag{78}$$

Now we show that Eq. (78) is indeed equal to Eq. (71). Since for Eq. (71), we show that we can pick arbitrary normalization condition. Thus, we set

$$\tilde{\boldsymbol{\phi}} = \mathbf{u}. \tag{79}$$

Due to Eq. (71), we have

$$\begin{aligned}
\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_r} &= \mathrm{Re}\left(\frac{\mathbf{u}\boldsymbol{\phi}^*}{\boldsymbol{\phi}^*\mathbf{u}}\right) = \mathrm{Re}\left(\frac{\mathbf{u}\boldsymbol{\phi}^*}{-1}\right) = -\mathbf{u}_r\boldsymbol{\phi}_r^\intercal - \mathbf{u}_i\boldsymbol{\phi}_i^\intercal, \\
\frac{\mathrm{d}\lambda_r}{\mathrm{d}\mathbf{A}_i} &= \mathrm{Im}\left(\frac{\mathbf{u}\boldsymbol{\phi}^*}{\boldsymbol{\phi}^*\mathbf{u}}\right) = \mathrm{Im}\left(\frac{\mathbf{u}\boldsymbol{\phi}^*}{-1}\right) = \mathbf{u}_r\boldsymbol{\phi}_i^\intercal - \mathbf{u}_i\boldsymbol{\phi}_r^\intercal,
\end{aligned} \tag{80}$$

Thus, we conclude that the eigenvalue derivative formula is a special case of the more general adjoint-based formula.

G Derivation of Eq. (12)

In this section, we provide the derivation for $(\partial\mathbf{r}/\partial\mathbf{A}_r)^\intercal\,\boldsymbol{\psi}$. The derivation for $(\partial\mathbf{r}/\partial\mathbf{A}_i)^\intercal\,\boldsymbol{\psi}$ is similar and is therefore omitted. As mentioned in the main text of the paper, we evaluate this product using RAD. Here, $\boldsymbol{\psi}$ can be taken as a seed for $\mathbf{r}$, i.e., $\boldsymbol{\psi}$ is an instance of $\bar{\mathbf{r}}$. We use the following identity for the derivation,

$$\mathrm{Tr}(\bar{\mathbf{r}}^\intercal\dot{\mathbf{r}}) = \mathrm{Tr}(\overline{\mathbf{A}}_r^\intercal\dot{\mathbf{A}}_r). \tag{81}$$

Before proceeding with deriving the RAD formulation, we derive the FAD expressions. We differentiate Eq. (7) to obtain the partial derivative of $\mathbf{r}$ with respect to $\mathbf{A}_r$. The FAD formula is given as

$$\dot{\mathbf{r}} = \begin{bmatrix}\dot{\mathbf{A}}_r\boldsymbol{\phi}_r\\ \dot{\mathbf{A}}_r\boldsymbol{\phi}_i\\ 0\\ 0\end{bmatrix}. \tag{82}$$

Now we derive $\bar{\mathbf{r}}$. By taking $\boldsymbol{\psi}$ as a reverse seed, we obtain

$$\mathrm{Tr}(\bar{\mathbf{r}}^\intercal\dot{\mathbf{r}}) = \mathrm{Tr}(\boldsymbol{\psi}^\intercal\dot{\mathbf{r}}). \tag{83}$$

We now substitute in the FAD formula Eq. (82), followed by expanding the $\boldsymbol{\psi} = \begin{bmatrix}\boldsymbol{\psi}_{\mathrm{main},r}^\intercal & \boldsymbol{\psi}_{\mathrm{main},i}^\intercal & \boldsymbol{\psi}_m^\intercal & \boldsymbol{\psi}_p^\intercal\end{bmatrix}^\intercal$, we obtain,

$$\mathrm{Tr}(\boldsymbol{\psi}^\intercal\dot{\mathbf{r}}) = \mathrm{Tr}\left(\boldsymbol{\psi}^\intercal\begin{bmatrix}\dot{\mathbf{A}}_r\boldsymbol{\phi}_r\\ \dot{\mathbf{A}}_r\boldsymbol{\phi}_i\\ 0\\ 0\end{bmatrix}\right) = \mathrm{Tr}\left(\boldsymbol{\psi}_{\mathrm{main},r}^\intercal\dot{\mathbf{A}}_r\boldsymbol{\phi}_r + \boldsymbol{\psi}_{\mathrm{main},i}^\intercal\dot{\mathbf{A}}_r\boldsymbol{\phi}_i\right). \tag{84}$$

Now, using the first identity from Eq. (55), and then factoring out similar terms, we obtain

$$\begin{aligned}
\mathrm{Tr}\left(\boldsymbol{\psi}_{\mathrm{main},r}^\intercal\dot{\mathbf{A}}_r\boldsymbol{\phi}_r + \boldsymbol{\psi}_{\mathrm{main},i}^\intercal\dot{\mathbf{A}}_r\boldsymbol{\phi}_i\right) &= \mathrm{Tr}\left(\boldsymbol{\phi}_r\boldsymbol{\psi}_{\mathrm{main},r}^\intercal\dot{\mathbf{A}}_r + \boldsymbol{\phi}_i\boldsymbol{\psi}_{\mathrm{main},i}^\intercal\dot{\mathbf{A}}_r\right) &(85)\\
&= \mathrm{Tr}\left(\left(\boldsymbol{\phi}_r\boldsymbol{\psi}_{\mathrm{main},r}^\intercal + \boldsymbol{\phi}_i\boldsymbol{\psi}_{\mathrm{main},i}^\intercal\right)\dot{\mathbf{A}}_r\right). &(86)
\end{aligned}$$

By Eq. (81), we can then write

$$\mathrm{Tr}\left(\left(\boldsymbol{\phi}_r\boldsymbol{\psi}_{\mathrm{main},r}^{\intercal} + \boldsymbol{\phi}_i\boldsymbol{\psi}_{\mathrm{main},i}^{\intercal}\right)\dot{\mathbf{A}}_r\right) = \mathrm{Tr}(\overline{\mathbf{A}}_r^{\intercal}\dot{\mathbf{A}}_r). \tag{87}$$

Since the equation holds for arbitrary $\dot{\mathbf{A}}_r$, comparing and matching the LHS and RHS we conclude that

$$\overline{\mathbf{A}}_r = \boldsymbol{\psi}_{\mathrm{main},r}\boldsymbol{\phi}_r^{\intercal} + \boldsymbol{\psi}_{\mathrm{main},i}\boldsymbol{\phi}_i^{\intercal}, \tag{88}$$

This finishes our derivation of Eq. (12).

View publication stats