



ELSEVIER

Comput. Methods Appl. Mech. Engrg. 190 (2001) 3121–3146

**Computer methods  
in applied  
mechanics and  
engineering**

www.elsevier.com/locate/cma

# A linearized method for the frequency analysis of three-dimensional fluid/structure interaction problems in all flow regimes

Michel Lesoinne<sup>a</sup>, Marcus Sarkis<sup>b</sup>, Ulrich Hetmaniuk<sup>a,c</sup>, Charbel Farhat<sup>a,\*</sup>

<sup>a</sup> Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado, Campus Box 429, Boulder, CO 80309-0429, USA

<sup>b</sup> Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609-2280, USA

<sup>c</sup> Ecole Nationale des Ponts et Chaussées, 6 and 8 avenue Pascal, 77455 Champs-sur-Marne, France

Received 15 February 1999

---

## Abstract

We present a computational fluid dynamics (CFD)-based linearized method for the frequency analysis of three-dimensional fluid/structure interaction problems. This method is valid in the subsonic, transonic, and supersonic flow regimes, and is insensitive to the frequency or damping level of the sought-after coupled eigenmodes. It is based on the solution by an orthogonal iteration procedure of a complex eigenvalue problem derived from the linearization of a three-field fluid/structure/moving mesh formulation. The key computational features of the proposed method include the reuse of existing unsteady flow solvers, a second-order approximation of the flux Jacobian matrix, and a parallel domain decomposition-based iterative solver for the solution of large-scale systems of discretized fluid/structure equations. While the frequency analysis method proposed here is primarily targeted at the extraction of the eigenpairs of a wet structure, we validate it with the flutter analysis of the AGARD Wing 445.6, for which experimental data is available. © 2001 Elsevier Science B.V. All rights reserved.

*Keywords:* Fluid/structure interaction; Aeroelasticity; Eigen analysis; Second-order Jacobian

---

## 1. Introduction

Numerical methods that rely on a simplified linear flow theory for computing the mode shapes, natural frequencies and damping levels of a “wet” structure are often restricted to *critical* subsonic and supersonic flow speeds [5,18,24]. They can neither predict the negative/positive damping of an aeroelastic system away from the flutter speed or in the transonic regime, nor compute non-flutter mode shapes that may be required for design and/or control purposes. Some of these methods have been extended to produce a more complete set of solutions [10]. However, the range of applications of these extensions usually excludes the transonic regime where nonlinear effects can be significant. In this paper, we describe a computational fluid dynamics (CFD)-based numerical method for computing a few complex eigensolutions of a given three-dimensional fluid/structure interaction problem that is valid in all flow regimes. This method is based on the linearization around an equilibrium point of the three-way coupled formulation of nonlinear transient aeroelastic problems first presented in [14,21,23], and the solution by an inverse orthogonal iteration

---

\* Corresponding author. Tel.: +1-303-492-3992; fax: +1-303-492-4990.

*E-mail addresses:* michel@vette.colorado.edu (M. Lesoinne), msarkis@wpi.edu (M. Sarkis), hetmaniu@vette.colorado.edu (U. Hetmaniuk), charbel@alexandra.colorado.edu (C. Farhat).

algorithm [16] of the corresponding generalized complex eigenvalue problem. Unlike classical approaches, it does not assume a harmonic motion, and therefore is applicable independently from the frequency content or damping level of the target aeroelastic modes.

In [22], the authors have presented a preliminary version of the method proposed herein, which emphasizes the reuse of existing unsteady flow solvers for solving coupled fluid/structure eigenvalue problems. In this paper, we re-examine the key discretization and algorithmic issues of this method in order to improve its computational efficiency. More specifically, we derive an explicit expression of a second-order approximation of the flux Jacobian matrix rather than using the Adifor software package for this purpose. We also tailor the restricted additive Schwarz (RAS) domain decomposition method [3,27] to the solution of the systems of equations that arise during the inverse orthogonal iterations. We show that both modifications improve significantly the overall computational performance of the method proposed for the frequency analysis of fluid/structure interaction problems. Finally, we validate this method with the flutter analysis of the AGARD Wing 445.6 in the subsonic, transonic, and early supersonic regimes – for which experimental data is available.

## 2. The three-field formulation

As stated earlier, our starting point is the three-field formulation developed in [14,21,23]. In order to keep this paper as self-contained as possible, we begin with an overview of this nonlinear formulation of fluid/structure interaction problems.

### 2.1. Governing equations

In many aeroelastic applications, some of the fluid domain boundaries undergo a motion with a large amplitude. In such problems, it is necessary to solve the flow equations on a moving and possibly deforming grid. Such a grid is commonly referred to in the literature as a dynamic mesh. Several approaches are available for solving fluid/structure interaction problems on dynamic meshes, among which we note the two closely related arbitrary Lagrangian–Eulerian (ALE) [8,14] and dynamic mesh [2] methods. These methods can be used to formulate the fluid/structure problem of interest as a three-field problem [21]: the fluid, the structure, and the dynamic mesh that is often represented by a pseudo-structural system. For example, in the case of the ALE method, a fluid/structure interaction problem can be described by the following coupled partial differential equations:

$$\begin{aligned} \frac{\partial(Jw)}{\partial t} \Big|_{\xi} + J\nabla_x \cdot \left( F(w) - \frac{\partial x}{\partial t} w \right) &= J\nabla_x \cdot R(w), \\ \rho_s \frac{\partial^2 u_s}{\partial t^2} - \operatorname{div}(\sigma_s(\epsilon_s(u_s))) &= b, \\ \tilde{\rho} \frac{\partial^2 x}{\partial t^2} - \operatorname{div}(\tilde{E} : \tilde{\epsilon}(x)) &= 0. \end{aligned} \tag{1}$$

The first of Eq. (1) is the ALE non-dimensional conservative form of the Navier–Stokes equations. Here,  $t$  denotes time,  $x(t)$  denotes the time-dependent position or displacement of a fluid grid point (depending on the context of the sentence and the equation),  $\xi$  its position in a reference configuration,  $J = \det(dx/d\xi)$ ,  $w$  is the fluid state vector using the conservative variables, and  $F$  and  $R$  denote, respectively, the convective and diffusive fluxes. The second of Eq. (1) is the elastodynamic equation where  $u_s$  denotes the displacement field of the structure and  $\rho_s$  its density,  $\sigma_s$  and  $\epsilon_s$  denote, respectively, the stress and strain tensors, and  $b$  represents the body forces acting on the given structure. The third of Eq. (1) governs the dynamics of the fluid moving grid. It is similar to the elastodynamic equation because the dynamic mesh is viewed here as a pseudo-structural system. A tilde notation is used to indicate that  $\tilde{\rho}$  is a fictitious density, and  $\tilde{E}$  is a fictitious tensor of elasticities [11]. The various Dirichlet and Neumann boundary conditions intrinsic to each of the fluid and structure problems are omitted here for simplicity.

The first and third of Eq. (1) are directly coupled. If  $u_F$  denotes the ALE displacement field of the fluid and  $p$  its pressure field,  $\sigma_S$  and  $\sigma_F$  the structure stress tensor and the fluid viscous stress tensor,  $\Gamma$  the fluid/structure interface boundary (wet boundary of the structure), and  $n$  is the normal at a point to  $\Gamma$ , the fluid and structure equations are coupled by the interface conditions

$$\begin{aligned} \sigma_S \cdot n &= -pn + \sigma_F \cdot n \quad \text{on } \Gamma, \\ \frac{\partial u_S}{\partial t} &= \frac{\partial u_F}{\partial t} \quad \text{on } \Gamma. \end{aligned} \tag{2}$$

The first of these two transmission conditions states that the tractions on the wet surface of the structure are in equilibrium with those on the fluid side of  $\Gamma$ . The second of Eq. (2) expresses the compatibility between the velocity fields of the structure and the fluid at the fluid/structure interface. For inviscid flows, this second equation is replaced by the slip wall boundary condition

$$\frac{\partial u_F}{\partial t} \cdot n = \frac{\partial u_S}{\partial t} \cdot n \quad \text{on } \Gamma. \tag{3}$$

The equations governing the structure and dynamic mesh motions are coupled by the continuity conditions

$$\begin{aligned} x &= u_S \quad \text{on } \Gamma, \\ \frac{\partial x}{\partial t} &= \frac{\partial u_S}{\partial t} \quad \text{on } \Gamma. \end{aligned} \tag{4}$$

In this paper, we consider only inviscid flows.

### 2.2. Semi-discretization

The spatial approximation of the ALE non-dimensional conservative form of the Euler equations by finite element or finite volume schemes leads to semi-discrete equations that can be written as

$$(\mathbf{A}(\hat{\mathbf{x}})\dot{\mathbf{w}}) + \mathbf{F}(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = 0, \tag{5}$$

where a bold font designates the discrete counterpart of a field variable, and a dot represents a time derivative. In the case of a finite volume semi-discretization,  $\mathbf{A}$  is the diagonal matrix of the cell volumes and  $\mathbf{F}$  is the numerical flux approximating the integral of the physical flux function over the cell interfaces.

The semi-discretization of the structural equations of dynamic equilibrium by finite elements leads to <sup>1</sup>

$$\mathbf{M}\ddot{\mathbf{u}}_S + \mathbf{f}^{\text{int}}(\mathbf{u}_S, \dot{\mathbf{u}}_S) = \mathbf{f}^{\text{ext}}(\mathbf{u}_S, \mathbf{w}), \tag{6}$$

where  $\mathbf{M}$  is the finite element mass matrix,  $\mathbf{u}_S$  the displacement vector,  $\mathbf{f}^{\text{int}}$  the vector of internal forces, and  $\mathbf{f}^{\text{ext}}$  is the vector of external forces induced by the fluid on the structure.

If the subscript  $i$  is used to designate the grid points located in the interior of a computational domain (structure or dynamic mesh), and the subscript  $b$  is used to designate those located at the fluid/structure interface  $\Gamma$ , and if the dynamic mesh is assimilated with a *quasi-static* pseudo-structural model, the semi-discrete equation governing the evolution of the dynamic mesh can be written as

$$\tilde{\mathbf{K}}_{ii}\mathbf{x}_i = -\tilde{\mathbf{K}}_{ib}\mathbf{x}_b, \quad \mathbf{x}_b = \mathbf{U}\mathbf{u}_{S_b}, \tag{7}$$

where  $\tilde{\mathbf{K}}$  is the fictitious stiffness matrix resulting from the finite element semi-discretization of the third of Eq. (1) [11], and  $\mathbf{U}$  is a transfer matrix. If the fluid and structure meshes have compatible interfaces,  $\mathbf{U} = \mathbf{I}$ . If not,  $\mathbf{U}$  is given by the finite element discretization of the second of the interface conditions (2) [13].

<sup>1</sup> This specific expression assumes that the rotational inertia forces are negligible.

### 3. Linearization of the semi-discrete equations

Given a structure and a set of flow conditions, we wish to compute the mode shapes, frequencies, and natural damping of the wet structure. A numerical method that addresses this problem is useful because: (a) it can be used to determine whether a fluid/structure system is stable or unstable in a given configuration, (b) it can be used as the building block of a sweeping strategy for determining the flutter envelope of an aeroelastic system, and (c) it provides the essential ingredients for designing an efficient active controller. An important characteristic of fluid/structure stability problems is that they can often be analyzed by linearizing the governing equations around an equilibrium point, and investigating the response of the target aeroelastic system to a small perturbation around that point. For this reason, we linearize the governing semi-discrete equations with respect to the variables  $\mathbf{w}$ ,  $\dot{\mathbf{w}}$ ,  $\mathbf{x}$ ,  $\dot{\mathbf{x}}$ ,  $\ddot{\mathbf{x}}$ ,  $\mathbf{u}_S$ ,  $\dot{\mathbf{u}}_S$ , and  $\ddot{\mathbf{u}}_S$  around an equilibrium configuration  $(\mathbf{w}_0, \dot{\mathbf{w}}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0, \ddot{\mathbf{x}}_0, \mathbf{u}_{S_0}, \dot{\mathbf{u}}_{S_0}, \ddot{\mathbf{u}}_{S_0})$

$$\begin{aligned} \mathbf{w} &= \mathbf{w}_0 + \delta\mathbf{w}, & \dot{\mathbf{w}} &= \dot{\mathbf{w}}_0 + \delta\dot{\mathbf{w}}, & \mathbf{x} &= \mathbf{x}_0 + \delta\mathbf{x}, & \dot{\mathbf{x}} &= \dot{\mathbf{x}}_0 + \delta\dot{\mathbf{x}}, & \ddot{\mathbf{x}} &= \ddot{\mathbf{x}}_0 + \delta\ddot{\mathbf{x}}, & \mathbf{u}_S &= \mathbf{u}_{S_0} + \delta\mathbf{u}_S, \\ \dot{\mathbf{u}}_S &= \dot{\mathbf{u}}_{S_0} + \delta\dot{\mathbf{u}}_S, & \ddot{\mathbf{u}}_S &= \ddot{\mathbf{u}}_{S_0} + \delta\ddot{\mathbf{u}}_S. \end{aligned}$$

The first term in Eq. (5) can be expanded as follows:

$$A(\hat{\mathbf{x}})\dot{\mathbf{w}} = A(\mathbf{x})\dot{\mathbf{w}} + (\hat{A}(\hat{\mathbf{x}}) - A(\mathbf{x}))\dot{\mathbf{w}} = A(\mathbf{x})\dot{\mathbf{w}} + \dot{x}_k \frac{\partial A}{\partial x_k} \mathbf{w} = A(\mathbf{x})\dot{\mathbf{w}} + \dot{x}_k \frac{\partial a_{ij}}{\partial x_k} w_j = A(\mathbf{x})\dot{\mathbf{w}} + \mathbf{E}(\mathbf{w})\dot{\mathbf{x}}, \quad (8)$$

where

$$e_{ik} = \frac{\partial a_{ij}}{\partial x_k} w_j. \quad (9)$$

Hence, its linearization is given by the following expression:

$$A(\mathbf{x}_0)\dot{\mathbf{w}}_0 + \left( \frac{\partial A}{\partial \mathbf{x}} \delta\mathbf{x} \right) \dot{\mathbf{w}}_0 + A(\mathbf{x}_0)\delta\dot{\mathbf{w}} + \mathbf{E}(\mathbf{w}_0)\dot{\mathbf{x}}_0 + \left( \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \delta\mathbf{w} \right) \dot{\mathbf{x}}_0 + \mathbf{E}(\mathbf{w}_0)\delta\dot{\mathbf{x}},$$

where  $(\partial \mathbf{E} / \partial \mathbf{w}) \delta\mathbf{w}$  is a matrix with coefficients

$$\left( \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \delta\mathbf{w} \right)_{ik} = \frac{\partial a_{ij}}{\partial x_k} \delta w_j. \quad (10)$$

Let

$$\begin{aligned} \mathbf{A}_0 &= A(\mathbf{x}_0), & \mathbf{E}_0 &= \mathbf{E}(\mathbf{w}_0), & \mathbf{F}_0 &= \mathbf{F}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0), & \mathbf{H}_0 &= \frac{\partial \mathbf{F}}{\partial \mathbf{w}}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0), \\ \mathbf{G}_0 &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0), & \mathbf{C}_0 &= \frac{\partial \mathbf{F}}{\partial \dot{\mathbf{x}}}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0). \end{aligned} \quad (11)$$

The linearization of the second term in Eq. (5) around the equilibrium position  $(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0)$  can be written as <sup>2</sup>

$$\mathbf{F}(\mathbf{w}, \mathbf{x}, \dot{\mathbf{x}}) = \mathbf{F}_0 + \mathbf{H}_0 \delta\mathbf{w} + \mathbf{G}_0 \delta\mathbf{x} + \mathbf{C}_0 \delta\dot{\mathbf{x}}. \quad (12)$$

Therefore, the complete linearized fluid equation is

$$\mathbf{A}_0(\dot{\mathbf{w}}_0 + \delta\dot{\mathbf{w}}) + \mathbf{E}_0(\dot{\mathbf{x}}_0 + \delta\dot{\mathbf{x}}) + \left( \frac{\partial A}{\partial \mathbf{x}} \delta\mathbf{x} \right)_0 \dot{\mathbf{w}}_0 + \left( \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \delta\mathbf{w} \right)_0 \dot{\mathbf{x}}_0 + \mathbf{F}_0 + \mathbf{H}_0 \delta\mathbf{w} + \mathbf{G}_0 \delta\mathbf{x} + \mathbf{C}_0 \delta\dot{\mathbf{x}} = 0. \quad (13)$$

<sup>2</sup> This linearization assumes that the flux function is differentiable at  $(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0)$ , which is not true for some flux functions used in CFD.

If  $\mathbf{K}_0$  and  $\mathbf{D}_0$  denote, respectively, the finite element stiffness and damping matrices of the structure at the equilibrium point, the linearization on the left-hand side of the structural equation of motion (6) can be expressed as

$$\mathbf{M}\ddot{\mathbf{u}}_{S_0} + \mathbf{M}\delta\ddot{\mathbf{u}}_S + \mathbf{f}^{\text{int}}(\mathbf{u}_{S_0}, \dot{\mathbf{u}}_{S_0}) + \mathbf{D}_0\delta\dot{\mathbf{u}}_S + \mathbf{K}_0\delta\mathbf{u}_S. \tag{14}$$

The linearization of the external forces  $\mathbf{f}^{\text{ext}}$  is given by

$$\mathbf{f}^{\text{ext}}(\mathbf{u}_S, \mathbf{w}) = \mathbf{f}^{\text{ext}}(\mathbf{u}_{S_0}, \mathbf{w}_0) + \mathbf{K}_{f_0}\delta\mathbf{u}_S + \mathbf{P}_0\delta\mathbf{w}, \tag{15}$$

where

$$\mathbf{K}_{f_0} = \frac{\partial \mathbf{f}^{\text{ext}}}{\partial \mathbf{u}_S}(\mathbf{u}_{S_0}, \mathbf{w}_0), \quad \mathbf{P}_0 = \frac{\partial \mathbf{f}^{\text{ext}}}{\partial \mathbf{w}}(\mathbf{u}_{S_0}, \mathbf{w}_0). \tag{16}$$

Hence, the complete linearized structural equation of motion can be written as

$$\mathbf{M}\delta\ddot{\mathbf{u}}_S + \mathbf{D}_0\delta\dot{\mathbf{u}}_S + (\mathbf{K}_0 - \mathbf{K}_{f_0})\delta\mathbf{u}_S = \mathbf{f}_0^{\text{ext}} - \mathbf{M}\ddot{\mathbf{u}}_{S_0} - \mathbf{f}_0^{\text{int}} + \mathbf{P}_0\delta\mathbf{w}. \tag{17}$$

By definition, the equilibrium point  $(\mathbf{w}_0, \dot{\mathbf{w}}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0, \ddot{\mathbf{x}}_0, \mathbf{u}_{S_0}, \dot{\mathbf{u}}_{S_0}, \ddot{\mathbf{u}}_{S_0})$  satisfies the governing Eqs. (5)–(7), as well as the following steady-state conditions:

$$\dot{\mathbf{w}}_0 = 0, \quad \mathbf{F}_0 = 0, \quad \dot{\mathbf{x}}_0 = 0, \quad \ddot{\mathbf{x}}_0 = 0, \quad \tilde{\mathbf{K}}_{ii}\mathbf{x}_{i_0} = -\tilde{\mathbf{K}}_{ib}\mathbf{U}\mathbf{u}_{S_{b_0}}, \quad \ddot{\mathbf{u}}_{S_0} = 0, \quad \dot{\mathbf{x}}_{0k} \left( \frac{\partial a_{ij}}{\partial x_k} \right)_0 = 0. \tag{18}$$

It follows that

$$\begin{aligned} \mathbf{A}_0\dot{\mathbf{w}}_0 + \mathbf{E}_0\dot{\mathbf{x}}_0 + \mathbf{F}_0 = 0, \quad \left( \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \delta \dot{\mathbf{x}} \right)_0 \dot{\mathbf{w}}_0 = 0, \quad \dot{\mathbf{x}}_{0k} \left( \frac{\partial a_{ij}}{\partial x_k} \right)_0 \delta w_j = 0, \\ \left( \frac{\partial \mathbf{E}}{\partial \mathbf{w}} \delta \mathbf{w} \right)_0 \dot{\mathbf{x}}_0 = 0, \quad \mathbf{M}\ddot{\mathbf{u}}_{S_0} + \mathbf{f}_0^{\text{int}} = \mathbf{f}_0^{\text{ext}}. \end{aligned}$$

Therefore, the linearized flow equation can be re-written as

$$\mathbf{A}_0\delta\dot{\mathbf{w}} + \mathbf{H}_0\delta\mathbf{w} + \mathbf{G}_0\delta\mathbf{x} + (\mathbf{E}_0 + \mathbf{C}_0)\delta\dot{\mathbf{x}} = 0 \tag{19}$$

and the linearized structural and mesh motion equations simplify to

$$\mathbf{M}\delta\ddot{\mathbf{u}}_S + \mathbf{D}_0\delta\dot{\mathbf{u}}_S + \mathbf{K}_{s_0}\delta\mathbf{u}_S = \mathbf{P}_0\delta\mathbf{w}, \tag{20}$$

where

$$\mathbf{K}_{s_0} = \mathbf{K}_0 - \mathbf{K}_{f_0}$$

and

$$\tilde{\mathbf{K}}_{ii}\delta\mathbf{x}_i = -\tilde{\mathbf{K}}_{ib}\mathbf{U}\delta\mathbf{u}_{S_b}. \tag{21}$$

We note that the adjusted structural stiffness matrix  $\mathbf{K}_{s_0}$  is not guaranteed to be positive definite. When it is not, the aeroelastic system exhibits divergence.

In the sequel, we drop the subscript 0 and the prefix  $\delta$  to simplify the notation. This should not cause any confusion because in our subsequent analysis, all matrices are evaluated at the equilibrium point, and all vectors are perturbation quantities.

Finally, we group Eqs. (19)–(21) to obtain the following representation of the linearized equations associated with the three-field formulation of fluid/structure interaction problems:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \mathbf{M} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{w}} \\ \mathbf{x} \\ \mathbf{u}_S \end{pmatrix} + \begin{pmatrix} \mathbf{A} & (\mathbf{E} + \mathbf{C}) & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \mathbf{D} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{w}} \\ \mathbf{x} \\ \mathbf{u}_S \end{pmatrix} + \begin{pmatrix} \mathbf{H} & \mathbf{G} & 0 \\ 0 & \tilde{\mathbf{K}}_{ii}^* & -\tilde{\mathbf{K}}_{ib}^* \\ -\mathbf{P} & 0 & \mathbf{K}_S \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{x} \\ \mathbf{u}_S \end{pmatrix} = 0, \tag{22}$$

where

$$\tilde{\mathbf{K}}_{ii}^* = \begin{pmatrix} \tilde{\mathbf{K}}_{ii} & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad \tilde{\mathbf{K}}_{ib}^* = \begin{pmatrix} 0 & -\tilde{\mathbf{K}}_{ib} \mathbf{U} \\ 0 & 0 \end{pmatrix} \quad (23)$$

and the above matrix partitioning is that corresponding to the partitioning of the unknowns into interior and  $\Gamma$ -boundary ones.

#### 4. The fluid/structure generalized eigenvalue problem

Let

$$\mathbf{K}^* = \tilde{\mathbf{K}}_{ii}^{*-1} \tilde{\mathbf{K}}_{ib}^*. \quad (24)$$

From Eqs. (22)–(24), it follows that

$$\mathbf{x} = \mathbf{K}^* \mathbf{u}_S, \quad \dot{\mathbf{x}} = \mathbf{K}^* \dot{\mathbf{u}}_S. \quad (25)$$

Next, we introduce the variable  $\mathbf{y} = \dot{\mathbf{u}}_S$ , and reduce the second-order coupled system (22) to the following first-order form:

$$\begin{pmatrix} \mathbf{A} & 0 & 0 \\ 0 & \mathbf{M} & 0 \\ 0 & 0 & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{w}} \\ \mathbf{y} \\ \mathbf{u}_S \end{pmatrix} + \begin{pmatrix} \mathbf{H} & (\mathbf{E} + \mathbf{C})\mathbf{K}^* & \mathbf{G}\mathbf{K}^* \\ -\mathbf{P} & 0 & \mathbf{K}_S \\ 0 & \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{y} \\ \mathbf{u}_S \end{pmatrix} = 0. \quad (26)$$

This first-order system of differential equations can be re-written as

$$\dot{\mathbf{q}} = \mathbf{N}\mathbf{q}, \quad (27)$$

where

$$\mathbf{q} = \begin{pmatrix} \mathbf{w} \\ \mathbf{y} \\ \mathbf{u}_S \end{pmatrix}, \quad \mathbf{N} = \begin{pmatrix} -\mathbf{A}^{-1}\mathbf{H} & -\mathbf{A}^{-1}(\mathbf{E} + \mathbf{C})\mathbf{K}^* & -\mathbf{A}^{-1}\mathbf{G}\mathbf{K}^* \\ \mathbf{M}^{-1}\mathbf{P} & 0 & -\mathbf{M}^{-1}\mathbf{K}_S \\ 0 & \mathbf{I} & 0 \end{pmatrix}. \quad (28)$$

The corresponding generalized eigenvalue problem is

$$(\mathbf{N} - \lambda_i \mathbf{I})\mathbf{q}_i = 0, \quad (29)$$

where  $\lambda_i$  is an eigenvalue and  $\mathbf{q}_i$  is an eigenvector.

One can reasonably argue that the generalized eigenvalue problem (29) is valid in all flow regimes – i.e., subsonic, transonic, and supersonic – as long as the viscous effects are not important, because it is derived from the linearization of the Euler equations. This eigenvalue problem has in general complex solutions. It is of great interest in aeroelasticity because the sign of the real part of each eigenvalue  $\lambda_i$  determines whether a given fluid/structure system is stable or not. Our main objective in this paper is to present a fast numerical algorithm for solving the generalized eigenvalue problem (29). For this purpose, we focus on the eigen-solutions that are the least stable – that is, for which  $\Re(\lambda_i)$  is the largest negative number, or is positive.

#### 5. Evaluation of the flux Jacobian matrix

In Eq. (22), only the semi-discrete flux Jacobian matrix  $\mathbf{H}$  requires special attention. In general, the numerical solution of the generalized eigenvalue problem (29) entails the evaluation of matrix-vector products of the form  $\mathbf{N}\mathbf{a}$ , which in turn incur matrix-vector products of the form  $\mathbf{H}\mathbf{b}$ . Here,  $\mathbf{a}$  and  $\mathbf{b}$  denote generic vectors of appropriate dimensions.

The matrix  $\mathbf{H}$  arises almost in every flow computation, in one form or another. Many if not most flow solvers compute only a first-order approximation of the flux Jacobian matrix (and in many cases only an

incomplete first-order approximation) because depending on the chosen semi-discretization method, (a) the construction of a second-order matrix  $\mathbf{H}$  (or even a complete first-order approximation of this matrix) may be a challenging task, and (b) a second-order matrix  $\mathbf{H}$  is in general more ill-conditioned than a first-order counterpart, (c) overall second-order accuracy can still be achieved by the flow solver using a defect/correction algorithm [7]. However, a straightforward defect/correction approach cannot be used for linearized problems such as those arising in this work.

With the advent of automatic differentiation tools such as Adifor [4], generating a second-order approximation of the flux Jacobian matrix has become an easier task. Unfortunately, Adifor is not particularly efficient for evaluating matrix-vector products of the form  $\mathbf{H}\mathbf{b}$  because when asked to generate the differential of a function with respect to a parameter, this software package generates both the function and its sought-after differential with respect to the specified parameter, which almost doubles the computational cost. Furthermore, automatic differentiation is prohibitive when applied to the evaluation of matrix-vector products of the form  $\mathbf{H}^T\mathbf{b}$ , which can arise in other applications, because in that case it requires an “execution tape” to be executed backward. For these reasons, we derive here “manually” a second-order approximation of the Jacobian flux matrix that incurs the same computational cost for evaluating  $\mathbf{H}\mathbf{b}$  and  $\mathbf{H}^T\mathbf{b}$ . More specifically, we focus on the case where the flow equations are semi-discretized by the finite volume method and the Roe upwinding scheme [25].

### 5.1. Finite volume semi-discretization

From Eq. (12), it follows that

$$\mathbf{H}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0) = \frac{\partial \mathbf{F}}{\partial \mathbf{w}}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0).$$

The solution method presented in this paper is independent from the scheme chosen for semi-discretizing the fluid equations. For this reason, we discuss only the case where these equations are semi-discretized by the finite volume method on a tetrahedral mesh from which a dual mesh defined by control volumes or cells is derived, and where the Roe upwinding scheme is used as a Riemann solver.

Hence, at an interior vertex point of the fluid mesh we have

$$[\mathbf{F}(\mathbf{w}_0, \mathbf{x}_0, \dot{\mathbf{x}}_0)]_i = \sum_{j \in \kappa(i)} \text{mes}(\partial C_{ij}) \Phi(w_i, w_j, \vec{n}_{ij}), \tag{30}$$

where because  $\dot{\mathbf{x}}_0 = 0$  (Eq. (18)),  $\Phi$  is the numerical function for the *usual* Roe flux

$$\Phi(w_i, w_j, \vec{n}) = \frac{1}{2} [\vec{F}(w_i) \cdot \vec{n}_{ij} + \vec{F}(w_j) \cdot \vec{n}_{ij}] - \frac{1}{2} |A_{\text{Roe}}(w_i, w_j, \vec{n}_{ij})| \cdot (w_j - w_i), \tag{31}$$

where  $w_i$  is the fluid state vector at vertex  $i$ ,  $\kappa(i)$  the set of vertices connected by an edge to vertex  $i$ ,  $C_i$  the control volume centered at vertex  $i$ , and  $\partial C_{ij}$  is the segment of the boundary of  $C_i$  that intersects the edge  $(ij)$ , and  $\vec{n}_{ij}$  is the unitary outer normal to  $\partial C_{ij}$ .

At the wall boundary, we compute the flux as follows:

$$\Phi_{\text{WALL}}(w_i, \vec{n}) = (0, p_i \vec{n}, 0)^T \tag{32}$$

and at the far-field boundary, we approximate the flux by a non-reflective version of the flux-splitting of Steger and Warming [28]

$$\Phi_{\infty}(w_i, w_{\infty}) = A^+(w_i, \vec{n})w_i + A^-(w_i, \vec{n})w_{\infty}, \tag{33}$$

where  $A^+(w, \vec{n})$  ( $A^-(w, \vec{n})$ ) is constructed from the Jacobian matrix  $A(w, \vec{n})$  of  $\vec{F}(w)\vec{n}$ , by taking into account only the positive (negative) eigenvalues of  $A(w, \vec{n})$ .

Eqs. (30)–(33) specify a first-order semi-discretization of the fluid equations. We denote by  $\mathbf{H}^{(1)}(\mathbf{w}_0)$ , the corresponding flux Jacobian matrix.

To achieve second-order spatial accuracy for the semi-discretization of the fluid equations, we adopt the monotonic upwinding scheme for conservation laws (MUSCL) [6,20] interpolation procedure equipped with a slope limiter. In this case, the numerical flux at an interior vertex point can be written as

$$[\mathbf{F}(\mathbf{w}_0, \mathbf{x}_0)]_i = \sum_{j \in \kappa(i)} \text{mes}(\partial C_{ij}) \Phi(w_{ij}, w_{ji}, \vec{n}_{ij}), \tag{34}$$

where  $w_{ij}$  and  $w_{ji}$  are two interpolated and limited fluid state vectors. The treatment of the wall and far-field boundaries is not modified. We denote by  $\mathbf{H}^{(2)}(\mathbf{w}_0)$  the flux Jacobian matrix associated with this second-order spatial approximation of the fluid equations, and discuss the key steps for evaluating  $\mathbf{H}^{(2)}(\mathbf{w}_0)$  in the two subsequent sections.

### 5.2. The MUSCL procedure

Next, we summarize the basic steps that combine the numerical flux function of Roe  $\Phi$ , a MUSCL interpolation procedure, and the limiter of van Albada [1], in order to achieve second-order space accuracy. Transform the fluid state vector from conservative variables  $w$  to primitive variables  $\tilde{w}$

$$w = \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ E \end{pmatrix} \mapsto \tilde{w} = \begin{pmatrix} \rho \\ v_1 \\ v_2 \\ v_3 \\ p \end{pmatrix},$$

where  $\rho$ ,  $v_1$ ,  $v_2$ ,  $v_3$ ,  $p$ , and  $E$  denote, respectively, the density, three velocity components, pressure, and total energy per unit mass of the fluid.

Compute the nodal gradients as follows:

$$\mathcal{G}_i(\tilde{w}) = \vec{\nabla} \tilde{w}_i = \frac{1}{\text{mes}(C_i)} \sum_{T, i \in T} \frac{\text{mes}(T)}{4} \left\{ \sum_{j \in T} \tilde{w}_j \vec{\nabla} N_j(T) \right\},$$

where  $T$  denotes a tetrahedron connected to vertex  $i$  and  $j$  a generic vertex in tetrahedron  $T$ , and  $N_j$  denotes the shape function associated with vertex  $j$ .

For each edge  $(ij)$ , compute the second-order state vectors  $(\tilde{w}_{ij}, \tilde{w}_{ji})$  component by component as follows:

$$\begin{aligned} v_{1_{ij}} &= v_{1_i} - \frac{1}{2} l \left( v_{1_i} - v_{1_j}, v_{1_j} - v_{1_i} - 4 \left[ \beta \vec{\nabla} v_{1_i} \cdot \vec{i}\vec{j} + \frac{1-2\beta}{2} (v_{1_j} - v_{1_i}) \right] \right), \\ v_{1_{ji}} &= v_{1_j} + \frac{1}{2} l \left( v_{1_i} - v_{1_j}, v_{1_j} - v_{1_i} - 4 \left[ \beta \vec{\nabla} v_{1_j} \cdot \vec{i}\vec{j} + \frac{1-2\beta}{2} (v_{1_j} - v_{1_i}) \right] \right), \end{aligned} \tag{35}$$

where  $\beta$  is an upwinding parameter that can be varied between 0 and 1, and  $l$  is the *limiter* function given by

$$l(a, b) = \begin{cases} \frac{a(b^2 + \varepsilon) + b(a^2 + \varepsilon)}{a^2 + b^2 + 2\varepsilon} & \text{if } ab > 0, \\ \frac{\varepsilon}{2} \frac{a + b}{a^2 + b^2 + 2\varepsilon} & \text{if } ab = 0, \\ 0 & \text{otherwise,} \end{cases}$$

where  $\varepsilon$  is a specified small number (i.e.,  $\varepsilon = 10^{-16}$ ). Eq. (35) can be also written in compact form as

$$\tilde{w}_{ij} = \mathcal{L}(\tilde{w}_i, \tilde{w}_j, \vec{\nabla} \tilde{w}_i, \vec{i}\vec{j}), \quad \tilde{w}_{ji} = \mathcal{L}(\tilde{w}_j, \tilde{w}_i, \vec{\nabla} \tilde{w}_j, \vec{j}\vec{i}). \tag{36}$$

Transform the fluid state vector back to the conservative variables and evaluate  $\Phi$  using the second-order state vectors

$$\Phi(w_{ij}, w_{ji}, \vec{n}_{ij}) = \Phi(\mathcal{U}^{-1}(\tilde{w}_{ij}), \mathcal{U}^{-1}(\tilde{w}_{ji}), \vec{n}_{ij}) = \tilde{\Phi}(\tilde{w}_{ij}, \tilde{w}_{ji}, \vec{n}_{ij}).$$

### 5.3. Linearization of the second-order semi-discretization

From Eq. (34), it follows that

$$\left[ \frac{\partial \mathbf{F}}{\partial \mathbf{w}}(\mathbf{w}, \mathbf{x}) \cdot \mathbf{b} \right]_i = \sum_{j \in \kappa(i)} \text{mes}(\partial C_{ij}) \frac{\partial}{\partial \mathbf{w}} \left( \Phi(w_{ij}, w_{ji}, \vec{n}_{ij}) \right) \cdot \mathbf{b} \tag{37}$$

$$\Rightarrow \left[ \frac{\partial \mathbf{F}}{\partial \mathbf{w}}(\mathbf{w}, \mathbf{x}) \cdot \mathbf{b} \right]_i = \sum_{j \in \kappa(i)} \text{mes}(\partial C_{ij}) \left( \frac{\partial \tilde{\Phi}}{\partial \tilde{w}_{ij}}(\tilde{w}_{ij}, \tilde{w}_{ji}, \vec{n}_{ij}) \cdot \frac{\partial \tilde{w}_{ij}}{\partial \mathbf{w}}(\mathbf{w}) \cdot \mathbf{b} + \frac{\partial \tilde{\Phi}}{\partial \tilde{w}_{ji}}(\tilde{w}_{ij}, \tilde{w}_{ji}, \vec{n}_{ij}) \cdot \frac{\partial \tilde{w}_{ji}}{\partial \mathbf{w}}(\mathbf{w}) \cdot \mathbf{b} \right). \tag{38}$$

From Eq. (36), we obtain

$$\begin{aligned} \frac{\partial \tilde{w}_{ij}}{\partial \mathbf{w}}(\mathbf{w}) \cdot \mathbf{b} &= \frac{\partial \mathcal{L}}{\partial \tilde{w}_i}(\mathcal{U}(w_i), \mathcal{U}(w_j), \mathcal{G}_i(\mathbf{U}(\mathbf{w})), \vec{i}_j) \cdot \frac{\partial \mathcal{U}}{\partial w}(w_i) \cdot b_i + \frac{\partial \mathcal{L}}{\partial \tilde{w}_j}(\mathcal{U}(w_i), \mathcal{U}(w_j), \mathcal{G}_i(\mathbf{U}(\mathbf{w})), \vec{i}_j) \\ &\cdot \frac{\partial \mathcal{U}}{\partial w}(w_j) \cdot b_j + \frac{\partial \mathcal{L}}{\partial \nabla \tilde{w}_i}(\mathcal{U}(w_i), \mathcal{U}(w_j), \mathcal{G}_i(\mathbf{U}(\mathbf{w})), \vec{i}_j) \cdot \frac{\partial \mathcal{G}_i}{\partial \mathbf{w}}(\mathbf{U}(\mathbf{w})) \cdot \frac{\partial \mathbf{U}}{\partial \mathbf{w}}(\mathbf{w}) \cdot \mathbf{b}, \end{aligned}$$

where  $\mathbf{U}(\mathbf{w})$  is the operator built from  $\mathcal{U}$  on vectors of size  $5n_F$ . A similar expression can be derived for  $(\partial \tilde{w}_{ji} / \partial \mathbf{w})(\mathbf{w})$ .

From the above expression, it follows that the evaluation of the flux Jacobian matrix  $\mathbf{H}$  can be obtained from the linearization of the basic steps outlined in Section 5.2

Linearization of the transformation of the state vector from conservative to primitive variables

$$\frac{\partial \mathcal{U}}{\partial \mathbf{w}}(\mathbf{w}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{v_1}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v_2}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{v_3}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ (\gamma - 1) \frac{v_1^2 + v_2^2 + v_3^2}{2} & -(\gamma - 1)v_1 & -(\gamma - 1)v_2 & -(\gamma - 1)v_3 & \gamma - 1 \end{pmatrix} \tag{39}$$

linearization of the gradient interpolation

$$\frac{\partial \mathcal{G}_i}{\partial \tilde{\mathbf{w}}}(\tilde{\mathbf{w}}) \cdot \mathbf{b} = \mathcal{G}_i(\mathbf{b})$$

linearization of the interpolation-limitation procedure

$$(v_1 + \delta v_1)_{ij} = v_{1ij} + \delta v_{1i} + \alpha_{ij}(\delta v_{1j} - \delta v_{1i}) + \beta_{ij} \vec{\nabla} \delta v_{1i} \cdot \vec{i}_j,$$

$$(v_1 + \delta v_1)_{ji} = v_{1ji} + \delta v_{1j} + \alpha_{ji}(\delta v_{1j} - \delta v_{1i}) + \beta_{ji} \vec{\nabla} \delta v_{1j} \cdot \vec{i}_j,$$

where

$$\begin{aligned} \alpha_{ij} &= \frac{1}{2}(l_a - (4\beta - 1)l_b) \left( v_{1_i} - v_{1_j}, v_{1_j} - v_{1_i} - 4 \left[ \beta \vec{\nabla} v_{1_i} \cdot \vec{i}_j + \frac{1-2\beta}{2}(v_{1_j} - v_{1_i}) \right] \right), \\ \beta_{ij} &= 2\beta l_b \left( v_{1_i} - v_{1_j}, v_{1_j} - v_{1_i} - 4 \left[ \beta \vec{\nabla} v_{1_i} \cdot \vec{i}_j + \frac{1-2\beta}{2}(v_{1_j} - v_{1_i}) \right] \right), \\ \alpha_{ji} &= -\frac{1}{2}(l_a - (4\beta - 1)l_b) \left( v_{1_i} - v_{1_j}, v_{1_j} - v_{1_i} - 4 \left[ \beta \vec{\nabla} v_{1_j} \cdot \vec{i}_j + \frac{1-2\beta}{2}(v_{1_j} - v_{1_i}) \right] \right), \\ \beta_{ji} &= -2\beta l_b \left( v_{1_i} - v_{1_j}, v_{1_i} - v_{1_j} - 4 \left[ \beta \vec{\nabla} v_{1_j} \cdot \vec{i}_j + \frac{1-2\beta}{2}(v_{1_j} - v_{1_i}) \right] \right) \end{aligned}$$

and

$$l_a(a, b) = \begin{cases} \frac{(b^2 + \varepsilon)(b^2 - a^2 + 2\varepsilon + 2ab)}{(a^2 + b^2 + 2\varepsilon)^2} & \text{if } ab > 0, \\ \frac{1}{2} \frac{(b^2 + \varepsilon)(b^2 - a^2 + 2\varepsilon)}{(a^2 + b^2 + 2\varepsilon)^2} & \text{if } ab = 0, \\ 0 & \text{otherwise} \end{cases}$$

and  $l_b$  is obtained by exchanging  $a$  and  $b$ .

Linearization of the transformation of the state vector back to conservative variables and the evaluation of the numerical flux function; note that the conservation property satisfied by  $\tilde{\Phi}$  implies

$$\tilde{\Phi}(\tilde{w}_i, \tilde{w}_j, \vec{n}) = -\tilde{\Phi}(\tilde{w}_j, \tilde{w}_i, -\vec{n})$$

and therefore

$$\frac{\partial \tilde{\Phi}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j, \vec{n}) = -\frac{\partial \tilde{\Phi}}{\partial \tilde{w}_j}(\tilde{w}_j, \tilde{w}_i, -\vec{n}).$$

Hence, we need to compute only  $\partial \tilde{\Phi} / \partial \tilde{w}_i$ . The explicit computation of this matrix is detailed in Appendix A.

At the wall boundary, we have

$$\Phi_{\text{WALL}}(w_i + \delta w_i, \vec{n}) = \Phi_{\text{WALL}}(w_i, \vec{n}) + (\gamma - 1) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}(v_1^2 + v_2^2 + v_3^2)\vec{n} & -v_1\vec{n} & -v_2\vec{n} & -v_3\vec{n} & \vec{n} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \delta w_i. \quad (40)$$

At the far-field boundary, we use the software Adifor to construct the  $5 \times 5$  matrix associated with the linearization of  $\Phi_\infty$ .

## 6. Solution of the generalized eigenvalue problem

### 6.1. Model reduction for the structure

Let  $n_S$  denotes the number of structural degrees of freedom and  $n_F$  denotes the number of fluid unknowns. The coupled fluid/structure eigenvalue problem admits  $2n_S + n_F$  solutions. However, an analyst is in general interested only in a relatively small number of eigenpairs. For example for flutter analysis, the eigenvalues of interest are those corresponding to a low frequency or a small damping. Furthermore, the eigensolutions of practical interest involve only a subset of the “dry” (uncoupled) modes. For all of the above reasons, it is reasonable, but not mandatory, to represent the structure by its first  $m$  dry modes, and therefore to switch from the displacement variable  $u_S$  to the modal variable (or generalized coordinate)  $u_{S_m}$  defined by

$$u_S = \Psi_m u_{S_m}, \quad (41)$$

where  $\Psi_m$  is an  $n_S \times m$  matrix storing the  $m$  lower dry eigenvectors  $\psi_i$ . These eigenvectors are solution of the generalized structural eigenvalue problem

$$K\psi_i = \omega_i^2 M\psi_i.$$

In general,  $m$  is relatively small compared to  $n_S$ , say  $10 \leq m \leq 30$ , and therefore the reduced mass and stiffness matrices

$$M_m = \Psi_m^T M \Psi_m, \quad K_m = \Psi_m^T K_S \Psi_m$$

and the reduced matrices

$$K_m^* = K^* \Psi_m, \quad P_m = \Psi_m^T P$$

have all small sizes.

Hence, the representation (41) reduces the generalized eigenvalue problem (29) to

$$(N_m - \mu_i I) \mathbf{q}_{m_i} = 0, \tag{42}$$

where

$$\mathbf{q}_m = \begin{pmatrix} \mathbf{w} \\ \mathbf{y}_m \\ \mathbf{u}_{S_m} \end{pmatrix} \tag{43}$$

and

$$N_m = \begin{pmatrix} -A^{-1}H & -A^{-1}(E + C)K_m^* & -A^{-1}GK_m^* \\ M_m^{-1}P_m & 0 & -M_m^{-1}K_m \\ 0 & I & 0 \end{pmatrix}. \tag{44}$$

### 6.2. Inverse orthogonal iteration

We are most interested in the extraction of the complex eigenvalues of  $N_m$  (and their corresponding eigenvectors) whose negative real parts are the closest to zero. These eigensolutions can be computed by a generalization of the inverse power method known as the method of inverse orthogonal iteration [16]. However, rather than operating directly on  $N_m$ , we choose to operate on  $(I - hN_m)$ ,  $h > 0$ , for reasons that will become clearer in the sequel. The matrices  $N_m$  and  $(I - hN_m)$  share the same eigenvectors, and if  $v_i$  is an eigenvalue of  $(I - hN_m)$ , then

$$\mu_i = \frac{1 - v_i}{h} \tag{45}$$

is the corresponding eigenvalue of  $N_m$ .

The convergence rate of the inverse orthogonal iteration is governed by the ratio

$$\frac{|v_{i+1}|}{|v_i|} = \frac{|1 - h\mu_{i+1}|}{|1 - h\mu_i|} = \frac{|(1/h) - \mu_{i+1}|}{|(1/h) - \mu_i|},$$

where  $||$  denotes the module of a complex number. This ratio is furthest from 1 when  $1/h$  is small. Hence,  $h$  is chosen relatively large to accelerate the convergence of the inverse orthogonal iteration.

Let  $r = 2m + n_F$ . Given an  $r$ -by- $p$  matrix  $T^{(0)}$  with  $p \leq m$  orthonormal columns, the following inverse orthogonal iteration procedure generates a sequence of matrices  $T^{(k)} \in \mathcal{R}^{r \times p}$  whose ranges  $\text{Range}(T^{(k)})$  converge to the subspace spanned by the  $p$  eigenvectors associated with the smallest eigenvalues of  $(I - hN_m)$ :

For  $k = 1, 2, \dots$

$$\text{Solve } (I - hN_m)Z^{(k)} = T^{(k-1)}.$$

Orthonormalize  $\mathbf{Z}^{(k)}$  using a  $Q$ – $R$  factorization  $\mathbf{T}^{(k)}\mathbf{R}^{(k)} = \mathbf{Z}^{(k)}$ .

Declare convergence if  $\|\mathbf{Z}^{(k)} - \mathbf{T}^{(k-1)}\|_2 \leq \tau \|\mathbf{T}^{(0)}\|_2$ .

The  $p \times p$  real matrix product  $\mathbf{T}^{(k)\top}\mathbf{Z}^{(k)}$  converges to an upper triangular matrix with either 1-by-1 or 2-by-2 diagonal blocks. These diagonal blocks lead to the smallest complex eigenvalues  $v_i$  from which one can deduce the smallest complex eigenvalues  $\mu_i$ . A good approximation of the eigenvectors  $\mathbf{q}_{m_i}$ ,  $i = 1, \dots, p$ , can then be obtained by computing the complete set of eigenvectors  $\mathbf{S}_p$  of the  $p$ -by- $p$  matrix  $\mathbf{T}^{(k)\top}\mathbf{Z}^{(k)}$ , and setting  $\mathbf{Q}_{m_p} = \mathbf{T}^{(k)}\mathbf{S}_p$ .

### 6.3. Linear equation solver

The most computationally intensive kernel of the inverse orthogonal iteration outlined in Section 6.2 is the solution of a linear problem of the form

$$\mathbf{B}\mathbf{Z}^{(k)} = (\mathbf{I} - h\mathbf{N}_m)\mathbf{Z}^{(k)} = \mathbf{T}^{(k-1)}. \quad (46)$$

We propose to solve the above problem, which arises at each inverse orthogonal iteration with a different right-hand side, by a pre-conditioned iterative algorithm equipped with a carefully constructed initial guess.

#### 6.3.1. Construction of the initial guess

At the  $k$ th step of the inverse orthogonal iteration procedure,  $\mathbf{Z}^{(k-1)}$  is readily available, and the objective is to solve  $p$  systems of the form

$$(\mathbf{I} - h\mathbf{N}_m)\mathbf{z}_i^{(k)} = \mathbf{t}_i^{(k-1)}.$$

A good initial guess for  $\mathbf{z}_i^{(k)}$  can be constructed by the following Rayleigh–Ritz procedure. First, we select the initial guess in the subspace spanned by  $\text{Range}(\mathbf{Z}^{(k-1)})$

$$\mathbf{z}_i^{(k)(0)} = \mathbf{Z}^{(k-1)}\boldsymbol{\lambda}.$$

Then, we compute  $\boldsymbol{\lambda}$  as to minimize the two-norm of the initial residual  $\|\mathbf{t}_i^{(k-1)} - (\mathbf{I} - h\mathbf{N}_m)\mathbf{Z}^{(k-1)}\boldsymbol{\lambda}\|_2$ . This leads to

$$\frac{d}{d\boldsymbol{\lambda}} \left( \boldsymbol{\lambda}^\top \mathbf{T}^{(k-2)\top} \mathbf{T}^{(k-2)} \boldsymbol{\lambda} - 2\boldsymbol{\lambda}^\top \mathbf{T}^{(k-2)\top} \mathbf{t}_i^{(k-1)} + \mathbf{t}_i^{(k-1)\top} \mathbf{t}_i^{(k-1)} \right) = 0.$$

Given that  $\mathbf{T}^{(k-2)}$  is an orthogonal matrix, it follows that the solution of the above equation is

$$\boldsymbol{\lambda} = \mathbf{T}^{(k-2)\top} \mathbf{t}_i^{(k-1)}.$$

Hence, the proposed initial guess for  $\mathbf{z}_i^{(k)}$  is

$$\mathbf{z}_i^{(k)(0)} = \mathbf{Z}^{(k-1)} \mathbf{T}^{(k-2)\top} \mathbf{t}_i^{(k-1)}.$$

In order to minimize storage requirements, we do not store the  $r \times p$  matrix  $\mathbf{Z}^{(k-1)}$  while performing the inverse orthogonal iteration. Instead, we store the smaller  $p \times p$  matrix  $\mathbf{R}^{(k-1)}$ . We reconstruct at each iteration of the eigensolver  $\mathbf{Z}^{(k-1)} = \mathbf{R}^{(k-1)}\mathbf{T}^{(k-1)}$ , and therefore compute the initial guess  $\mathbf{z}_i^{(k)(0)}$  as follows:

$$\mathbf{z}_i^{(k)(0)} = \mathbf{R}^{(k-1)} \mathbf{T}^{(k-1)} \mathbf{T}^{(k-2)\top} \mathbf{t}_i^{(k-1)}.$$

In summary, we initialize the solution by an iterative algorithm of each problem of the form given in (38) by

$$\mathbf{Z}_0^{(k)} = \mathbf{R}^{(k-1)} \mathbf{T}^{(k-1)} \mathbf{T}^{(k-2)\top} \mathbf{T}^{(k-1)}.$$

6.3.2. Elimination of the structure unknowns

Let  $\Delta \mathbf{Z}^{(k)} = \mathbf{Z}^{(k)} - \mathbf{Z}_0^{(k)}$  and  $\Delta \mathbf{T}^{(k)} = \mathbf{T}^{(k-1)} - \mathbf{B} \Delta \mathbf{Z}_0^{(k)}$ . Problem (46) can be re-written as

$$\mathbf{B} \Delta \mathbf{Z}^{(k)} = \Delta \mathbf{T}^{(k)}. \tag{47}$$

The above system of linear equations with multiple right-hand sides can also be written as  $p$  linear  $2 \times 2$  block systems of the form

$$\begin{pmatrix} \mathbf{B}_{\text{FF}} & \mathbf{B}_{\text{FS}} \\ \mathbf{B}_{\text{SF}} & \mathbf{B}_{\text{SS}} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{z}_{\text{F}}^{(k)} \\ \Delta \mathbf{z}_{\text{S}}^{(k)} \end{pmatrix} = \begin{pmatrix} \Delta \mathbf{t}_{\text{F}}^{(k)} \\ \Delta \mathbf{t}_{\text{S}}^{(k)} \end{pmatrix},$$

where

$$\mathbf{B}_{\text{FF}} = \mathbf{I} + h\mathbf{A}^{-1}\mathbf{H}, \quad \mathbf{B}_{\text{FS}} = (h\mathbf{A}^{-1}(\mathbf{E} + \mathbf{C})\mathbf{K}_m^* \quad h\mathbf{A}^{-1}\mathbf{G}\mathbf{K}_m^*),$$

$$\mathbf{B}_{\text{SF}} = \begin{pmatrix} -h\mathbf{M}_m^{-1}\mathbf{P}_m \\ 0 \end{pmatrix} \quad \mathbf{B}_{\text{SS}} = \begin{pmatrix} 0 & h\mathbf{M}_m^{-1}\mathbf{K}_m \\ -h\mathbf{I} & 0 \end{pmatrix}$$

and where the subscripts F and S refer to the fluid and structure unknowns, respectively.

It follows that

$$\Delta \mathbf{z}_{\text{S}}^{(k)} = \mathbf{S}^{-1} \left( \Delta \mathbf{t}_{\text{S}}^{(k)} - \mathbf{B}_{\text{SF}} \mathbf{B}_{\text{FF}}^{-1} \Delta \mathbf{t}_{\text{F}}^{(k)} \right) \tag{48}$$

and

$$\Delta \mathbf{z}_{\text{F}}^{(k)} = \mathbf{B}_{\text{FF}}^{-1} \Delta \mathbf{t}_{\text{F}}^{(k)} - \mathbf{B}_{\text{FF}}^{-1} \mathbf{B}_{\text{FS}} \Delta \mathbf{z}_{\text{S}}^{(k)}, \tag{49}$$

where

$$\mathbf{S} = \mathbf{B}_{\text{SS}} - \mathbf{B}_{\text{SF}} \mathbf{B}_{\text{FF}}^{-1} \mathbf{B}_{\text{FS}}. \tag{50}$$

Note that  $\mathbf{S}$  is a  $2m \times 2m$  matrix that needs to be computed *only once*, and can be stored. Hence, the main computational cost associated with the solution of the problems (48) and (49) is that corresponding to the solution of problems of the form

$$\mathbf{B}_{\text{FF}} \mathbf{x} = \mathbf{g}, \tag{51}$$

where  $\mathbf{g}$  is either a  $\Delta \mathbf{t}_{\text{S}}^{(k)}$  vector, or a column of  $\mathbf{B}_{\text{FS}}$ . Such problems can be solved approximately, but at the price of a slower convergence of the inverse orthogonal iteration procedure. Hence, the overall efficiency of the proposed eigensolver calls for a balance between solving accurately the system of equations (51), and optimizing the convergence rate of the inverse orthogonal iteration method.

Finally, we note that

$$\mathbf{A} \mathbf{B}_{\text{FF}} = \mathbf{A} + h\mathbf{H}.$$

Hence, if  $h$  is identified with a time-step  $\Delta t$ , the above linear operator becomes nearly identical to the one that arises at each Newton step of an implicit unsteady flow solver, which shows that the proposed frequency analysis method can re-use a significant amount of existing flow solver technology. This is essentially the reason why we have reformulated the original generalized eigenvalue problem  $N_m \mathbf{q}_{m_i} = \mu_i \mathbf{q}_{m_i}$  as  $(\mathbf{I} - hN_m) \mathbf{q}_{m_i} = \nu_i \mathbf{q}_{m_i}$ .

6.3.3. Pre-conditioned iterative algorithms

Next, we turn our attention to the solution by pre-conditioned iterative algorithms of the problems of the form

$$\mathbf{B}_{\text{FF}} \mathbf{x} = \mathbf{g} \tag{52}$$

that arise in Eqs. (48) and (49). Here and throughout the remainder of this paper,  $\mathbf{B}_{\text{FF}} = \mathbf{I} + h\mathbf{A}^{-1}\mathbf{H}$  is based on the second-order flux Jacobian matrix  $\mathbf{H}^{(2)}$ .

Such algorithms compute a sequence  $\mathbf{y}^0, \mathbf{y}^1, \dots, \mathbf{y}^k$  for which  $\mathbf{x}^k = \mathbf{R}^k \mathbf{y}^k$  converges to  $\mathbf{x}$  [26]. Convergence is usually monitored by

$$\|\mathbf{L}^k (\mathbf{B}_{\text{FF}} \mathbf{x}^k - \mathbf{g})\|_2 \leq \delta \|\mathbf{L}^0 \mathbf{g}\|_2. \quad (53)$$

The linear operators  $\mathbf{L}^k$  and  $\mathbf{R}^k$  are the left and right pre-conditioners, respectively, and  $\delta > 0$  is a pre-selected tolerance.

We present three such pre-conditioned algorithms that share the same left pre-conditioner  $\mathbf{L}^k = \mathbf{L} = \mathbf{D}^{-1}$  – and therefore the same stopping criterion (53) – but differ in the choice of the right pre-conditioner  $\mathbf{R}^k$ . Here and in the sequel,  $\mathbf{D}$  is a block diagonal matrix constructed from the  $5 \times 5$  diagonal blocks of the matrix  $\mathbf{B}_{\text{FF}}^{(1)}$ , where

$$\mathbf{B}_{\text{FF}}^{(1)} = \mathbf{I} + h\mathbf{A}^{-1}\mathbf{H}^{(1)}. \quad (54)$$

We remind the reader that  $\mathbf{H}^{(1)}$  is the first-order approximation of the flux Jacobian matrix. Hence, in all three solvers that we describe next,  $\mathbf{B}_{\text{FF}}$  is left pre-conditioned by the diagonal blocks of its first-order counterpart.

Next, we specify these three different iterative solvers.

#### 6.3.4. The JAC–JAC algorithm

The first method we describe here, labeled JAC–JAC, is a two-level iterative method with outer and inner iterations. It is easier to implement and requires less memory than the two other methods that we describe in the following two sections. However, it is less robust and less computationally efficient than both of them. The outer iteration of the JAC–JAC algorithm is the following block-Jacobi scheme  $\mathbf{y}^0 = \mathbf{0}$  for  $k = 0, 1, \dots$

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \mathbf{D}^{-1}(\mathbf{B}_{\text{FF}} \mathbf{R}^k \mathbf{y}^k - \mathbf{g}).$$

The right pre-conditioner  $\mathbf{R}^k$  is also chosen as the block-Jacobi iterative scheme. Since the number of block-Jacobi iterations can vary with the pre-conditioning step, it follows that the right pre-conditioner can depend on  $k$ , which explains the superscript  $k$ . More specifically, the right pre-conditioning step computes  $\mathbf{x}^k = \mathbf{R}^k \mathbf{y}^k$  via the solution of the auxiliary problem  $\mathbf{B}_{\text{FF}}^{(1)} \mathbf{x}^k = \mathbf{D} \mathbf{y}^k$  by the block-Jacobi iterative algorithm  $\mathbf{x}_0^k = \mathbf{0}$  for  $j = 0, 1, \dots$

$$\mathbf{x}_{j+1}^k = \mathbf{x}_j^k - \mathbf{D}^{-1}(\mathbf{B}_{\text{FF}}^{(1)} \mathbf{x}_j^k - \mathbf{D} \mathbf{y}^k). \quad (55)$$

We monitor the convergence of the above inner iterations by

$$\|\mathbf{B}_{\text{FF}}^{(1)} \mathbf{x}_j^k - \mathbf{D} \mathbf{y}^k\|_2 \leq \eta \|\mathbf{D} \mathbf{y}^k\|_2,$$

where  $\eta > 0$  is a specified tolerance. Moreover, we note that for large CFL numbers – that is, for large values of  $h$  (see Section 6.3.2) – the iterative algorithm (55) may not converge. For this reason, we impose a limit on the maximum number of iterations defining this right pre-conditioner. Global convergence is reached when the outer iterations converge to the solution of problem (51).

#### 6.3.5. The JAC–RAS1 algorithm

The second solution method we present, labeled JAC–RAS1, differs from the previous one only in the choice of the right pre-conditioner. Here, we define  $\mathbf{R}$  as the RAS pre-conditioner that was introduced in [27]. The RAS algorithm is a variant of the classical additive Schwarz pre-conditioner (AS) [9] that was shown to deliver a superior performance on both sequential and parallel computational platforms. Here, we outline its basic steps in order to keep this paper as self-contained as possible.

Let

$$\tilde{\mathbf{B}} = \mathbf{D}^{-1} \mathbf{B}_{\text{FF}}^{(1)} \quad (56)$$

and let  $\mathcal{N}$  denote the set of nodes in the fluid mesh, and  $\mathcal{N}_i, i = 1, \dots, N$  a partition of this set into  $N$  overlapping set of nodes

$$\mathcal{N} = \bigcup_{i=1}^N \mathcal{N}_i.$$

To obtain a non-overlapping node-based mesh partition, we identify a unique subdomain as the sole owner of each interface node (see Fig. 1). Hence, for each  $\mathcal{N}_i$ , we construct  $\mathcal{N}_i^{(0)} \subset \mathcal{N}_i$  such that

$$\bigcup_{i=1}^N \mathcal{N}_i^{(0)} = \mathcal{N} \quad \text{and} \quad \mathcal{N}_i^{(0)} \cap \mathcal{N}_j^{(0)} = \emptyset \quad \text{for } i \neq j.$$

Let  $n$  be the total number of nodes in  $\mathcal{N}$ . For each subdomain  $\mathcal{N}_i^{(0)}$ , we define a restriction operator  $\mathbf{I}_i^0$ . In matrix terms,  $\mathbf{I}_i^0$  is a  $5n \times 5n$  block subidentity matrix where each  $5 \times 5$  diagonal block is set to the identity matrix if the corresponding node belongs to  $\mathcal{N}_i^{(0)}$ , and to zero otherwise. Similarly, we construct  $\mathbf{I}_i$  for each  $\mathcal{N}_i$ . We also define

$$\tilde{\mathbf{B}}_i = \mathbf{I}_i \tilde{\mathbf{B}} \mathbf{I}_i.$$

Although  $\tilde{\mathbf{B}}_i$  is not invertible, we can invert its restriction to the subspace  $L_i$

$$\tilde{\mathbf{B}}_i^{-1} \equiv \left( (\tilde{\mathbf{B}}_i)_{|L_i} \right)^{-1},$$

where  $L_i$  is the restriction of  $\mathcal{R}^{5n}$  to  $\mathcal{N}_i$  extended by zero outside  $\mathcal{N}_i$ . In the numerical experiments reported in this paper,  $\tilde{\mathbf{B}}_i^{-1}$  are obtained by the incomplete factorization ILU(0) and stored in  $5 \times 5$  block diagonal compressed row format.

We introduce the RAS operator as follows:

$$\mathbf{R}_{\text{RAS}} \equiv \mathbf{I}_1 \tilde{\mathbf{B}}_1^{-1} \mathbf{I}_1^0 + \dots + \mathbf{I}_N \tilde{\mathbf{B}}_N^{-1} \mathbf{I}_N^0$$

and define the right pre-conditioning step  $\mathbf{x}^k = \mathbf{R}^k \mathbf{y}^k$  as the solution of the auxiliary problem  $\mathbf{B}_{\text{FF}}^{(1)} \mathbf{x}^k = \mathbf{D} \mathbf{y}^k$  by the RAS pre-conditioned GMRES iterative algorithm. We monitor the convergence of the resulting inner iterations by

$$\left\| \mathbf{B}_{\text{FF}}^{(1)} \mathbf{x}_j^k - \mathbf{D} \mathbf{y}^k \right\|_2 \leq \eta \left\| \mathbf{D} \mathbf{y}^k \right\|_2,$$

where  $\eta > 0$  is a specified tolerance.

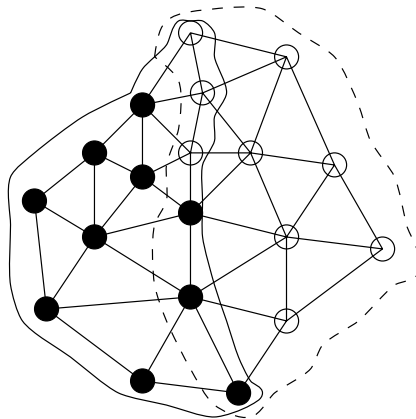


Fig. 1. A two-subdomain mesh partition.  $\mathcal{N}_1^{(0)}$  contains all the ‘●’ nodes,  $\mathcal{N}_2^{(0)}$  contains all the ‘○’ nodes, and  $\mathcal{N}_1^{(0)} \cap \mathcal{N}_2^{(0)} = \emptyset$ .  $\mathcal{N}_1^{(1)}$  contains all the nodes inside the solid curve, and  $\mathcal{N}_2^{(1)}$  contains all the nodes inside the dotted curve.

### 6.3.6. The JAC–RAS2 algorithm

The two algorithms JAC–JAC and JAC–RAS1 described above have two things in common:

- both contain inner and outer iterations;
- both left and right pre-conditioners of both algorithms are based on the first-order flux Jacobian matrix  $\mathbf{H}^{(1)}$ .

Hence, in both JAC–JAC and JAC–RAS1 algorithms, few matrix-vector products of the form  $\mathbf{B}_{\text{FF}}\mathbf{x}$  are performed compared to matrix-vector products of the form  $\mathbf{B}_{\text{FF}}^{(1)}\mathbf{x}$ . It follows that both of these algorithms can afford performing the matrix-vector products of the form  $\mathbf{H}^{(2)}\mathbf{x}$  by using Adifor, which is convenient if an explicit formula of  $\mathbf{H}^{(2)}\mathbf{x}$  is not available, but less efficient as discussed in Section 5.

Here, we present a third algorithm for solving problem (51) labeled JAC–RAS2 that:

- does not contain inner iterations;
- will be shown to have superior convergence properties than both algorithms JAC–JAC and JAC–RAS1;
- but require the availability of an explicit expression for matrix-vector products of the form  $\mathbf{H}^{(2)}\mathbf{x}$  (see Appendix A) as it performs mostly matrix-vector products of the form  $\mathbf{B}_{\text{FF}}\mathbf{x}$ .

This algorithm consists in solving problem (51) by the block-Jacobi pre-conditioned GMRES algorithm. In other words, JAC–RAS2 solves problem (51) by applying the GMRES method to the solution of

$$\mathbf{D}^{-1}\mathbf{B}_{\text{FF}}\mathbf{R}_{\text{RAS}}\mathbf{y} = \mathbf{D}^{-1}\mathbf{g}$$

and outputting at convergence  $\mathbf{x} = \mathbf{R}_{\text{RAS}}\mathbf{y}$ .

## 7. Application to the flutter analysis of the AGARD wing 445.6

Here, we validate the proposed frequency analysis method and illustrate some aspects of its computational performance.

### 7.1. Validation

We consider the flutter analysis of the AGARD wing 445.6 [29]. This wing is an AGARD standard aeroelastic configuration with a 45° quarter-chord sweep angle, a panel aspect ratio of 1.65, a taper ratio of 0.66, and a NACA 65A004 airfoil section (Fig. 2). The model selected here is the so-called 2.5 ft weakened

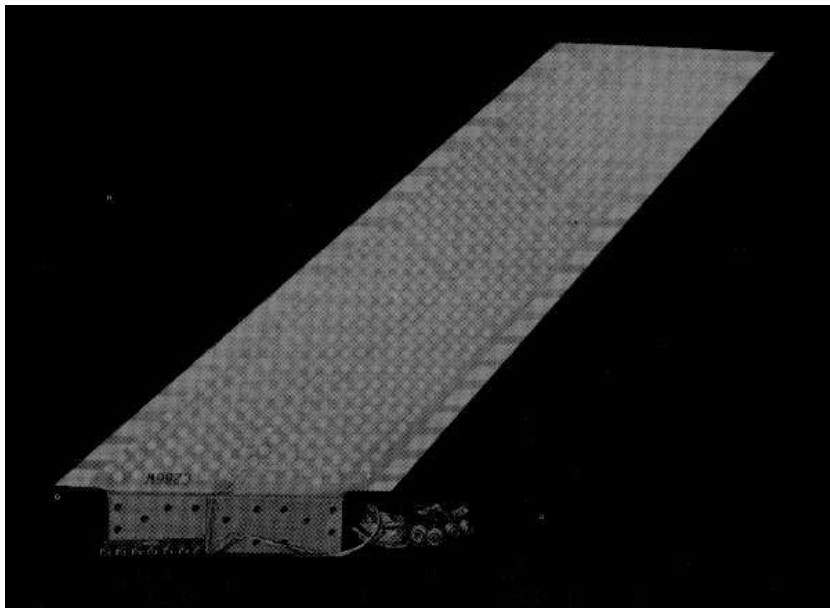


Fig. 2. The AGARD wing 445.6.

model 3 whose measured modal frequencies and wind-tunnel flutter test results are reported in [29], and for which computational aeroelastic data can be found in [17,19].

We construct an undamped finite element model of the wing with 800 triangular composite shell elements and 2646 degrees of freedom, using the information given in [29]. This model yields natural mode shapes and frequencies that are similar to those derived experimentally. More specifically, the frequencies associated with the first four natural modes of this finite element model are, respectively, 9.83, 39.54, 50.50, and 96.95 Hz. They differ from the experimental ones by only 2.5%, 3.6%, 4.5%, and 5.9%, respectively.

We also generate a three-dimensional unstructured tetrahedral CFD Euler mesh using GHS3D [15]. This mesh contains 22 014 vertices only.

We apply the eigensolution method proposed in this paper to the prediction of the flutter speed index as a function of the Mach number, using the second-order approximation of the flux Jacobian matrix. More specifically, we represent the structure by its first 10 dry modes ( $m = 10$ ), and employ six trial vectors to initialize the method of inverse orthogonal iteration. For each target Mach number, we set the freestream density as in [29], and vary the freestream pressure until we observe the onset of flutter characterized by a vanishing real part of a computed eigenvalue. We report our results in Fig. 3 and compare them with the experimental data published in [29], and the computational results published in [17,19]. Note that the flutter envelopes reported in [17,19] are derived from the solution of nonlinear transient aeroelastic response problems where the unsteady flow computations are performed using an Euler solver.

The results reported in Fig. 3 show that in the range  $0.499 \leq M_\infty \leq 0.960$ , the flutter speed indices predicted by our computational methodology compare favorably with the experimental data, and with both nonlinear computational results published in [17,19]. In the supersonic regime, our results compare more favorably with experimental data than those of [19], but less favorably than those of [17]. Indeed, in the supersonic regime our computational results validate those of [19]. However, the results reported in this work as well as those reported in [17,19] correspond to Euler flow solutions, and as stated in [19], the modeling of the flow physics by the Euler equations is incomplete for the AGARD wing 445.6 in the early supersonic regime.

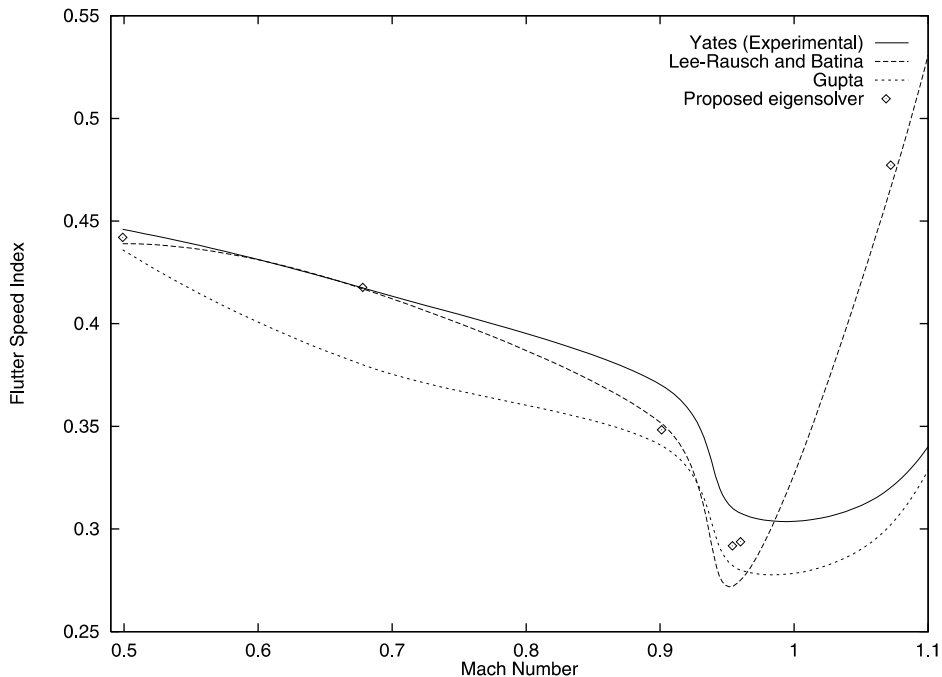


Fig. 3. Flutter speed index (AGARD wing 445.6).

## 7.2. Performance results

All computations discussed in Section 7.1 have been performed on a three-processor origin 2000, and were parallelized using the mesh partitioning paradigm and the TOP/DOMDEC software package [12].

In general, we have observed that for this mesh with 22 014 vertices, a matrix-vector product of the form  $\mathbf{H}^{(2)}(\mathbf{w}_0) \cdot \mathbf{b}$  requires 0.27 s when using the manual derivation presented in Appendix A, and 1.10 s when using Adifor. On the other hand, a matrix-vector product of the form  $\mathbf{H}^{(1)}(\mathbf{w}_0) \cdot \mathbf{b}$  consumes only 0.14 s. These observations justify the concern formulated in Section 5 about using Adifor for this application. They also justify the design of algorithms JAC–JAC and JAC–RAS1 for the event where for convenience, a software package such as Adifor must be used for generating matrix-vector products of the form  $\mathbf{H}^{(2)}\mathbf{x}$ .

First, we report in Table 1 on the performances of the JAC–JAC and JAC–RAS1 algorithms for the solution of one instance of the right pre-conditioning problem, which can be written in both cases as  $\mathbf{B}_{\text{FF}}^{(1)}\mathbf{x} = \mathbf{D}\mathbf{y}$ . Note that this right pre-conditioning problem is similar to problem (51) when constructed with a first-order flux Jacobian matrix. We summarize both the outer iteration counts and CPU results (between parentheses), for various CFL numbers. In all cases, we set the tolerance of the convergence criterion governing the inner iterations to  $\eta = 10^{-1}$ . We remind the reader that (a)  $h$  can be identified with the time-step of an existing unsteady flow solver (see Section 6.3.2) and therefore varying  $h$  can be achieved by varying the CFL number for a flow solver, and (b) the higher is the CFL number the faster is the expected convergence of the eigensolver (see Section 6.2). The reader can observe that in all cases, the JAC–RAS1 algorithm outperforms significantly the JAC–JAC solver. More importantly, the reader can also observe that the RAS pre-conditioner is almost insensitive to  $h$  or the CFL number. This is important because the larger is the value of  $h$ , the better is the expected convergence rate of the eigensolver, but the more ill-conditioned is the matrix  $\mathbf{B}_{\text{FF}} = \mathbf{I} + h\mathbf{A}^{-1}\mathbf{H}$ .

In Table 2, we report on the performances of all three solvers for the solution of one problem of the form  $\mathbf{B}_{\text{FF}}\mathbf{x} = \mathbf{y}$ . The CFL number is fixed to  $\text{CFL} = 9000$ , and the number of inner iterations is varied by varying the parameter  $\eta$ . The reader can observe that (a) in the case of the JAC–RAS1 solver, the right pre-conditioning auxiliary problem does not need to be solved very accurately to ensure a good performance, and (b) all solvers perform more iterations than in the case of a first-order flux Jacobian matrix, which highlights the well-known difference in conditioning between first-order and second-order space accurate Jacobian matrices, and (c) the JAC–RAS2 performs less iterations than the JAC–JAC and JAC–RAS1 algorithms whose numbers of iterations are given by the product of the number of inner iterations and the number of outer iterations.

Table 1  
Performance results for the solution of  $\mathbf{B}_{\text{FF}}^{(1)}\mathbf{x} = \mathbf{D}\mathbf{y}$

| CFL      | $\eta = 10^{-1}$ |             |             |             |
|----------|------------------|-------------|-------------|-------------|
|          | 100              | 1000        | 10 000      | 100 000     |
| JAC–JAC  | 7 (1.05 s)       | 26 (3.78 s) | 50 (7.12 s) | 65 (9.27 s) |
| JAC–RAS1 | 2 (0.66 s)       | 3 (1.03 s)  | 4 (1.25 s)  | 5 (1.67 s)  |

Table 2  
Convergence of the outer iterations for  $\mathbf{B}_{\text{FF}}\mathbf{x} = \mathbf{y}$  –  $\text{CFL} = 9000$  –  $\delta = 10^{-3}$

| JAC–JAC |       | JAC–RAS1 |       | JAC–RAS2 |
|---------|-------|----------|-------|----------|
| Outer   | Inner | Outer    | Inner | Outer    |
| 17      | 32    | 13       | 8     | –        |
| 29      | 16    | 15       | 4     | –        |
| 46      | 8     | 23       | 2     | –        |
| Diverge | 4     | 35       | 1     | 27       |

Table 3

Performance of the inverse orthogonal iteration – CFL = 9000,  $p = 2$ , and  $\tau = 10^{-3}$

|     | JAC–JAC | JAC–RAS1 | JAC–RAS2 | JAC–RAS2–Adifor |
|-----|---------|----------|----------|-----------------|
| CPU | 7650 s  | 2345 s   | 1002 s   | 3562 s          |

Table 4

Performance results of the inverse orthogonal iteration equipped with JAC–RAS2 for different values of  $p$  – CFL = 9000,  $\tau = 10^{-3}$

| $p$ | CPU    | Iteration counts |
|-----|--------|------------------|
| 2   | 1002 s | 44               |
| 4   | 3020 s | 82               |
| 6   | 5860 s | 160              |

The overall performance results reported in Table 3 for the eigensolver and the case of two eigensolutions ( $p = 2$ ) also show that the JAC–RAS2 solver is significantly faster than the two other solvers. These results also highlight the importance of using the explicit expression of  $\mathbf{H}^{(2)}\mathbf{x}$  given in Appendix A rather than using Adifor.

Finally, we report in Table 4 on the performance of the eigensolver equipped with the JAC–RAS2 algorithm, for different numbers  $p$  of target eigensolutions. The reported results show that, for this problem, the number of iterations of the eigensolver grows linearly with  $p$ , but its solution time grows superlinearly with  $p$ .

### Acknowledgements

The authors acknowledge the support by the Air Force Office of Scientific Research under Grant F49620-98-1-0112.

### Appendix A

The objective of this appendix is to evaluate the quantity

$$\frac{\partial \tilde{\Phi}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j, \vec{n}),$$

where  $\tilde{w}_i$  is the restriction to vertex  $i$  of the fluid state vector expressed in terms of the primitive variables, and  $\Phi$  is the flux of Roe. In conservative variables, this flux can be written as

$$\Phi(w_i, w_j, \vec{n}) = \frac{1}{2} [\vec{F}(w_i) \cdot \vec{n} + \vec{F}(w_j) \cdot \vec{n}] - \frac{1}{2} |A_{\text{Roe}}(w_i, w_j, \vec{n})| \cdot (w_j - w_i),$$

where  $\vec{n}$  is a unitary vector.

In primitive variables, this flux becomes

$$\begin{aligned} \tilde{\Phi}(\tilde{w}_i, \tilde{w}_j, \vec{n}) = & \frac{1}{2} (\tilde{F}(\tilde{w}_i) \cdot \vec{n} + \tilde{F}(\tilde{w}_j) \cdot \vec{n}) - \frac{1}{2} \mathcal{P}^{-1}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \vec{n}) \cdot |A(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \vec{n})| \\ & \cdot \mathcal{P}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \vec{n}) \cdot (\tilde{w}_j - \tilde{w}_i), \end{aligned}$$

where

$$\vec{F}(\tilde{w}) \cdot \vec{n} = \begin{bmatrix} \rho \vec{v} \cdot \vec{n} \\ \rho v_1 \vec{v} \cdot \vec{n} + p n_1 \\ \rho v_2 \vec{v} \cdot \vec{n} + p n_2 \\ \rho v_3 \vec{v} \cdot \vec{n} + p n_3 \\ \rho \mathcal{H} \vec{v} \cdot \vec{n} \end{bmatrix},$$

where  $\vec{v}$  is the flow velocity vector,  $\mathcal{H}$  the total specific enthalpy given by

$$\mathcal{H} = \frac{\gamma P}{\rho(\gamma - 1)} + \frac{1}{2}(v_1^2 + v_2^2 + v_3^2)$$

and  $\mathcal{M}$  is the averaging function associated with the flux of Roe.

$$\mathcal{M}(\tilde{w}_i, \tilde{w}_j) = \frac{1}{\sqrt{\rho_i} + \sqrt{\rho_j}} \begin{bmatrix} \sqrt{\rho_i} \rho_i + \sqrt{\rho_j} \rho_j \\ \sqrt{\rho_i} v_{1i} + \sqrt{\rho_j} v_{1j} \\ \sqrt{\rho_i} v_{2i} + \sqrt{\rho_j} v_{2j} \\ \sqrt{\rho_i} v_{3i} + \sqrt{\rho_j} v_{3j} \\ \sqrt{\rho_i} \mathcal{H}_i + \sqrt{\rho_j} \mathcal{H}_j \end{bmatrix}.$$

$\mathcal{P}^{-1}$  is the matrix whose columns are the right eigenvectors of the Jacobian matrix of  $\vec{F}$ .

$$\mathcal{P}^{-1}(\tilde{w}, \vec{n}) = \begin{bmatrix} n_1 & n_2 & n_3 & \frac{1}{2c^2} & \frac{1}{2c^2} \\ v_1 n_1 & v_1 n_2 - n_3 & v_1 n_3 + n_2 & \frac{v_1 + cn_1}{2c^2} & \frac{v_1 - cn_1}{2c^2} \\ v_2 n_1 + n_3 & v_2 n_2 & v_2 n_3 - n_1 & \frac{v_2 + cn_2}{2c^2} & \frac{v_2 - cn_2}{2c^2} \\ v_3 n_1 - n_2 & v_3 n_2 + n_1 & v_3 n_3 & \frac{v_3 + cn_3}{2c^2} & \frac{v_3 - cn_3}{2c^2} \\ \left( \frac{\|\vec{v}\|^2}{2} \vec{n} - \vec{n} \times \vec{v} \right) \cdot \vec{e}_1 & \left( \frac{\|\vec{v}\|^2}{2} \vec{n} - \vec{n} \times \vec{v} \right) \cdot \vec{e}_2 & \left( \frac{\|\vec{v}\|^2}{2} \vec{n} - \vec{n} \times \vec{v} \right) \cdot \vec{e}_3 & \frac{p + c\vec{v} \cdot \vec{n}}{2c^2} & \frac{p - c\vec{v} \cdot \vec{n}}{2c^2} \end{bmatrix},$$

$c$  is for the speed of sound

$$c = \sqrt{\frac{\gamma P}{\rho}}$$

and  $\vec{e}_1, \vec{e}_2$  and  $\vec{e}_3$  define a basis of  $\mathcal{R}^3$ . We denote by  $r_p(\tilde{w}, \vec{n})$ , the column vectors of  $\mathcal{P}^{-1}(\tilde{w}, \vec{n})$ .

Matrix  $A$  is the diagonal matrix of the eigenvalues of the Jacobian matrix of  $\vec{F}$

$$A(\tilde{w}, \vec{n}) = \begin{bmatrix} \vec{v} \cdot \vec{n} & 0 & 0 & 0 & 0 \\ 0 & \vec{v} \cdot \vec{n} & 0 & 0 & 0 \\ 0 & 0 & \vec{v} \cdot \vec{n} & 0 & 0 \\ 0 & 0 & 0 & \vec{v} \cdot \vec{n} + c & 0 \\ 0 & 0 & 0 & 0 & \vec{v} \cdot \vec{n} - c \end{bmatrix}$$

and  $\mathcal{P}$  is the matrix whose lines are the left eigenvectors of the Jacobian matrix of  $\vec{F}$

$$\mathcal{P}(\tilde{w}, \tilde{n}) = \begin{bmatrix} \left(\tilde{n} - \frac{(\gamma-1)\|\tilde{v}\|^2}{2c^2}\tilde{n} + \tilde{n} \times \tilde{v}\right) \cdot \tilde{e}_1 & n_1(\gamma-1)\frac{v_1}{c^2} & n_3 + n_1(\gamma-1)\frac{v_2}{c^2} & -n_2 + n_1(\gamma-1)\frac{v_3}{c^2} & -n_1\frac{\gamma-1}{c^2} \\ \left(\tilde{n} - \frac{(\gamma-1)\|\tilde{v}\|^2}{2c^2}\tilde{n} + \tilde{n} \times \tilde{v}\right) \cdot \tilde{e}_2 & -n_3 + n_2(\gamma-1)\frac{v_1}{c^2} & n_2(\gamma-1)\frac{v_2}{c^2} & n_1 + n_2(\gamma-1)\frac{v_3}{c^2} & -n_2\frac{\gamma-1}{c^2} \\ \left(\tilde{n} - \frac{(\gamma-1)\|\tilde{v}\|^2}{2c^2}\tilde{n} + \tilde{n} \times \tilde{v}\right) \cdot \tilde{e}_3 & n_2 + n_3(\gamma-1)\frac{v_1}{c^2} & -n_1 + n_3(\gamma-1)\frac{v_2}{c^2} & n_3(\gamma-1)\frac{v_3}{c^2} & -n_3\frac{\gamma-1}{c^2} \\ (\gamma-1)\frac{\|\tilde{v}\|^2}{2} - c\tilde{v} \cdot \tilde{n} & cn_1 - (\gamma-1)v_1 & cn_2 - (\gamma-1)v_2 & cn_3 - (\gamma-1)v_3 & \gamma-1 \\ (\gamma-1)\frac{\|\tilde{v}\|^2}{2} + c\tilde{v} \cdot \tilde{n} & -cn_1 - (\gamma-1)v_1 & -cn_2 - (\gamma-1)v_2 & -cn_3 - (\gamma-1)v_3 & \gamma-1 \end{bmatrix}.$$

In the sequel, we denote by  $l_p^T(\tilde{w}, \tilde{n})$  the line vectors of  $\mathcal{P}(\tilde{w}, \tilde{n})$ .

Hence, the differentiation of  $\tilde{\Phi}$  gives

$$\begin{aligned} \frac{\partial \tilde{\Phi}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j, \tilde{n}) \cdot b &= \frac{1}{2} \tilde{A}(\tilde{w}_i, \tilde{n}) \cdot b - \frac{1}{2} \left( \frac{\partial \mathcal{P}^{-1}}{\partial \tilde{w}}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot \frac{\partial \mathcal{M}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j) \cdot b \right) |A(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n})| \\ &\cdot \mathcal{P}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot (\tilde{w}_j - \tilde{w}_i) - \frac{1}{2} \mathcal{P}^{-1}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \\ &\cdot \left( \frac{\partial |A|}{\partial \tilde{w}}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot \frac{\partial \mathcal{M}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j) \cdot b \right) \mathcal{P}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot (\tilde{w}_j - \tilde{w}_i) \\ &- \frac{1}{2} \mathcal{P}^{-1}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot |A(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n})| \cdot \left( \frac{\partial \mathcal{P}}{\partial \tilde{w}}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot \frac{\partial \mathcal{M}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j) \cdot b \right) \\ &+ \frac{1}{2} \mathcal{P}^{-1}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot |A(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n})| \cdot \mathcal{P}(\mathcal{M}(\tilde{w}_i, \tilde{w}_j), \tilde{n}) \cdot b, \end{aligned}$$

where

$$\tilde{A}(\tilde{w}, \tilde{n}) = \begin{bmatrix} \tilde{v} \cdot \tilde{n} & \rho n_1 & \rho n_2 & \rho n_3 & 0 \\ v_1 \tilde{v} \cdot \tilde{n} & \rho \tilde{v} \cdot \tilde{n} + \rho v_1 n_1 & \rho v_1 n_2 & \rho v_1 n_3 & n_1 \\ v_2 \tilde{v} \cdot \tilde{n} & \rho v_2 n_1 & \rho \tilde{v} \cdot \tilde{n} + \rho v_2 n_2 & \rho v_2 n_3 & n_2 \\ v_3 \tilde{v} \cdot \tilde{n} & \rho v_3 n_1 & \rho v_3 n_2 & \rho \tilde{v} \cdot \tilde{n} + \rho v_3 n_3 & n_3 \\ \frac{\|\tilde{v}\|^2}{2} \tilde{v} \cdot \tilde{n} & \rho v_1 \tilde{v} \cdot \tilde{n} + \rho n_1 \mathcal{H} & \rho v_2 \tilde{v} \cdot \tilde{n} + \rho n_2 \mathcal{H} & \rho v_3 \tilde{v} \cdot \tilde{n} + \rho n_3 \mathcal{H} & \frac{\gamma}{\gamma-1} \tilde{v} \cdot \tilde{n} \end{bmatrix},$$

$$\frac{\partial \mathcal{M}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j) = \frac{1}{2\sqrt{\rho_i}(\sqrt{\rho_i} + \sqrt{\rho_j})} \begin{bmatrix} 3\rho_i - \frac{\sqrt{\rho_i}\rho_i + \sqrt{\rho_j}\rho_j}{\sqrt{\rho_i} + \sqrt{\rho_j}} & 0 & 0 & 0 & 0 \\ v_{1i} - \frac{\sqrt{\rho_i}v_{1i} + \sqrt{\rho_j}v_{1j}}{\sqrt{\rho_i} + \sqrt{\rho_j}} & 2\rho_i & 0 & 0 & 0 \\ v_{2i} - \frac{\sqrt{\rho_i}v_{2i} + \sqrt{\rho_j}v_{2j}}{\sqrt{\rho_i} + \sqrt{\rho_j}} & 0 & 2\rho_i & 0 & 0 \\ v_{3i} - \frac{\sqrt{\rho_i}v_{3i} + \sqrt{\rho_j}v_{3j}}{\sqrt{\rho_i} + \sqrt{\rho_j}} & 0 & 0 & 2\rho_i & 0 \\ \frac{\|\tilde{v}\|^2}{2} - \frac{\gamma p}{\rho(\gamma-1)} - \frac{\sqrt{\rho_i}\mathcal{H}_i + \sqrt{\rho_j}\mathcal{H}_j}{\sqrt{\rho_i} + \sqrt{\rho_j}} & 2\rho_i v_{1i} & 2\rho_i v_{2i} & 2\rho_i v_{3i} & \frac{2\gamma}{\gamma-1} \end{bmatrix}$$

and

$$\frac{\partial \mathcal{P}^{-1}}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b = \left[ \frac{\partial r_1}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \frac{\partial r_2}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \frac{\partial r_3}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \frac{\partial r_4}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \frac{\partial r_5}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \right]$$

with

$$\frac{\partial r_1}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & n_1 & 0 & 0 & 0 \\ 0 & 0 & n_1 & 0 & 0 \\ 0 & 0 & 0 & n_1 & 0 \\ 0 & v_1 n_1 & v_2 n_1 + n_3 & v_3 n_1 - n_2 & 0 \end{bmatrix},$$

$$\frac{\partial r_2}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & n_2 & 0 & 0 & 0 \\ 0 & 0 & n_2 & 0 & 0 \\ 0 & 0 & 0 & n_2 & 0 \\ 0 & v_1 n_2 - n_3 & v_2 n_2 & v_3 n_2 + n_1 & 0 \end{bmatrix},$$

$$\frac{\partial r_3}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & n_3 & 0 & 0 & 0 \\ 0 & 0 & n_3 & 0 & 0 \\ 0 & 0 & 0 & n_3 & 0 \\ 0 & v_1 n_3 + n_2 & v_2 n_3 - n_1 & v_3 n_3 & 0 \end{bmatrix},$$

$$\frac{\partial r_4}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} \frac{1}{2\gamma p} & 0 & 0 & 0 & -\frac{\rho}{2\gamma p^2} \\ \frac{v_1}{2\rho c^2} + \frac{n_1}{4\rho c} & \frac{1}{2c^2} & 0 & 0 & -\frac{v_1}{2pc^2} - \frac{n_1}{4pc} \\ \frac{v_2}{2\rho c^2} + \frac{n_2}{4\rho c} & 0 & \frac{1}{2c^2} & 0 & -\frac{v_2}{2pc^2} - \frac{n_2}{4pc} \\ \frac{v_3}{2\rho c^2} + \frac{n_3}{4\rho c} & 0 & 0 & \frac{1}{2c^2} & -\frac{v_3}{2pc^2} - \frac{n_3}{4pc} \\ \frac{1}{2\gamma} + \frac{\vec{v} \cdot \vec{n}}{4\rho c} & \frac{n_1}{2c} & \frac{n_2}{2c} & \frac{n_3}{2c} & -\frac{\vec{v} \cdot \vec{n}}{4pc} \end{bmatrix},$$

$$\frac{\partial r_5}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} \frac{1}{2\gamma p} & 0 & 0 & 0 & -\frac{\rho}{2\gamma p^2} \\ \frac{v_1}{2\rho c^2} - \frac{n_1}{4\rho c} & \frac{1}{2c^2} & 0 & 0 & -\frac{v_1}{2pc^2} + \frac{n_1}{4pc} \\ \frac{v_2}{2\rho c^2} - \frac{n_2}{4\rho c} & 0 & \frac{1}{2c^2} & 0 & -\frac{v_2}{2pc^2} + \frac{n_2}{4pc} \\ \frac{v_3}{2\rho c^2} - \frac{n_3}{4\rho c} & 0 & 0 & \frac{1}{2c^2} & -\frac{v_3}{2pc^2} + \frac{n_3}{4pc} \\ \frac{1}{2\gamma} - \frac{\vec{v} \cdot \vec{n}}{4\rho c} & -\frac{n_1}{2c} & -\frac{n_2}{2c} & -\frac{n_3}{2c} & \frac{\vec{v} \cdot \vec{n}}{4pc} \end{bmatrix}.$$

The differentiation of  $|A|$  is given by the differentiation of the eigenvalues  $\lambda_i$

$$\frac{\partial \lambda_1}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ 0 \end{bmatrix},$$

$$\frac{\partial \lambda_4}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} -\frac{c}{2\rho} \\ n_1 \\ n_2 \\ n_3 \\ \frac{c}{2\rho} \end{bmatrix},$$

$$\frac{\partial \lambda_5}{\partial \tilde{w}}(\tilde{w}, \vec{n}) = \begin{bmatrix} \frac{c}{2\rho} \\ n_1 \\ n_2 \\ n_3 \\ -\frac{c}{2\rho} \end{bmatrix}$$

and the differentiation of the absolute value is performed as

$$\begin{cases} |x + \delta x| \rightarrow |x| + \delta x & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ |x + \delta x| \rightarrow |x| - \delta x & \text{if } x < 0. \end{cases}$$

Furthermore, we have

$$\frac{\partial \mathcal{P}}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b = \begin{bmatrix} \frac{\partial I_1^T}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \\ \frac{\partial I_2^T}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \\ \frac{\partial I_3^T}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \\ \frac{\partial I_4^T}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \\ \frac{\partial I_5^T}{\partial \tilde{w}}(\tilde{w}, \vec{n}) \cdot b \end{bmatrix},$$

where

$$\frac{\partial I_1}{\partial \tilde{w}}(\tilde{w}, \tilde{n}) = \begin{bmatrix} -\frac{n_1(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} & -\frac{n_1(\gamma-1)v_1}{c^2} & -\frac{n_1(\gamma-1)v_2}{c^2} - n_3 & -\frac{n_1(\gamma-1)v_3}{c^2} + n_2 & \frac{n_1(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} \\ \frac{n_1 v_1(\gamma-1)}{\gamma p} & \frac{n_1 \rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_1 v_1 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_1 v_2(\gamma-1)}{\gamma p} & 0 & \frac{n_1 \rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_1 v_2 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_1 v_3(\gamma-1)}{\gamma p} & 0 & 0 & \frac{n_1 \rho(\gamma-1)}{\gamma p} & -\frac{n_1 v_3 \rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_1(\gamma-1)}{\rho c^2} & 0 & 0 & 0 & \frac{n_1(\gamma-1)}{\rho c^2} \end{bmatrix},$$

$$\frac{\partial I_2}{\partial \tilde{w}}(\tilde{w}, \tilde{n}) = \begin{bmatrix} -\frac{n_2(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} & -\frac{n_2(\gamma-1)v_1}{c^2} + n_3 & -\frac{n_2(\gamma-1)v_2}{c^2} & -\frac{n_2(\gamma-1)v_3}{c^2} - n_1 & \frac{n_2(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} \\ \frac{n_2 v_1(\gamma-1)}{\gamma p} & \frac{n_2 \rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_2 v_1 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_2 v_2(\gamma-1)}{\gamma p} & 0 & \frac{n_2 \rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_2 v_2 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_2 v_3(\gamma-1)}{\gamma p} & 0 & 0 & \frac{n_2 \rho(\gamma-1)}{\gamma p} & -\frac{n_2 v_3 \rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_2(\gamma-1)}{\rho c^2} & 0 & 0 & 0 & \frac{n_2(\gamma-1)}{\rho c^2} \end{bmatrix},$$

$$\frac{\partial I_3}{\partial \tilde{w}}(\tilde{w}, \tilde{n}) = \begin{bmatrix} -\frac{n_3(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} & -\frac{n_3(\gamma-1)v_1}{c^2} - n_2 & -\frac{n_3(\gamma-1)v_2}{c^2} + n_1 & -\frac{n_3(\gamma-1)v_3}{c^2} & \frac{n_3(\gamma-1)\|\tilde{v}\|^2}{2\rho c^2} \\ \frac{n_3 v_1(\gamma-1)}{\gamma p} & \frac{n_3 \rho(\gamma-1)}{\gamma p} & 0 & 0 & -\frac{n_3 v_1 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_3 v_2(\gamma-1)}{\gamma p} & 0 & \frac{n_3 \rho(\gamma-1)}{\gamma p} & 0 & -\frac{n_3 v_2 \rho(\gamma-1)}{\gamma p^2} \\ \frac{n_3 v_3(\gamma-1)}{\gamma p} & 0 & 0 & \frac{n_3 \rho(\gamma-1)}{\gamma p} & -\frac{n_3 v_3 \rho(\gamma-1)}{\gamma p^2} \\ -\frac{n_3(\gamma-1)}{\rho c^2} & 0 & 0 & 0 & \frac{n_3(\gamma-1)}{\rho c^2} \end{bmatrix},$$

$$\frac{\partial I_4}{\partial \tilde{w}}(\tilde{w}, \tilde{n}) = \begin{bmatrix} \frac{c}{2\rho} \tilde{v} \cdot \tilde{n} & (\gamma-1)v_1 - cn_1 & (\gamma-1)v_2 - cn_2 & (\gamma-1)v_3 - cn_3 & -\frac{c}{2\rho} \tilde{v} \cdot \tilde{n} \\ -\frac{cn_1}{2\rho} & -(\gamma-1) & 0 & 0 & \frac{cn_1}{2\rho} \\ -\frac{cn_2}{2\rho} & 0 & -(\gamma-1) & 0 & \frac{cn_2}{2\rho} \\ -\frac{cn_3}{2\rho} & 0 & 0 & -(\gamma-1) & \frac{cn_3}{2\rho} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\frac{\partial I_5}{\partial \tilde{w}}(\tilde{w}, \tilde{n}) = \begin{bmatrix} -\frac{c}{2\rho} \tilde{v} \cdot \tilde{n} & (\gamma - 1)v_1 + cn_1 & (\gamma - 1)v_2 + cn_2 & (\gamma - 1)v_3 + cn_3 & \frac{c}{2p} \tilde{v} \cdot \tilde{n} \\ \frac{cn_1}{2\rho} & -(\gamma - 1) & 0 & 0 & -\frac{cn_1}{2p} \\ \frac{cn_2}{2\rho} & 0 & -(\gamma - 1) & 0 & -\frac{cn_2}{2p} \\ \frac{cn_3}{2\rho} & 0 & 0 & -(\gamma - 1) & -\frac{cn_3}{2p} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

which concludes the evaluation of  $\frac{\partial \tilde{\Phi}}{\partial \tilde{w}_i}(\tilde{w}_i, \tilde{w}_j, \tilde{n})$ .

## References

- [1] G.D. Van Albada, B. Van Leer, W.W. Roberts, A comparative study of computational methods in cosmic gas dynamics, *Astron. Astrophys.* 108 (1982) 76–84.
- [2] J.T. Batina, Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA Paper No. 89-0115, AIAA 27th Aerospace Sciences Meeting, Reno, Nevada, 1989.
- [3] X.-C. Cai, C. Farhat, M. Sarkis, A minimum overlap restricted additive Schwarz pre-conditioner and applications in 3D flow simulations, *Contemporary Math.* 218 (1998) 478–484.
- [4] G. Corliss, A. Griewank, P. Hovland, Adifor – generating derivative codes from FORTRAN programs, *Scientific Programming* 1 (1992).
- [5] R. Dat, J.L. Meurzec, On the flutter calculations by the so-called reduced frequency scanning method, *La Recherche Aérospatiale* 133 (1969).
- [6] A. Dervieux, Steady Euler simulations using unstructured meshes, Von Kármán Institute Lecture Series, 1985.
- [7] J.-A. Désidéri, P.W. Hemker, Convergence analysis of the defect-correction iteration for hyperbolic problems, *SIAM J. Sci. Comput.* 16 (1995) 88–118.
- [8] J. Donea, An arbitrary Lagrangian–Eulerian finite element method for transient fluid-structure interactions, *Comput. Methods Appl. Mech. Engrg.* 33 (1982) 689–723.
- [9] M. Dryja, O.B. Widlund, Domain decomposition algorithms with small overlap, *SIAM J. Sci. Comput.* 15 (1994) 604–620.
- [10] J.W. Edwards, Unsteady aerodynamic modeling and active aeroelastic control, SUDAAR 504 Report, Ph.D. Thesis, Stanford University, 1997.
- [11] C. Farhat, High performance simulation of coupled nonlinear transient aeroelastic problems, AGARD Report R-807, Special Course on Parallel Computing in CFD, 1995.
- [12] C. Farhat, S. Lantéri, H.D. Simon, Top/domdec, a software tool for mesh partitioning and parallel processing, *J. Comput. Syst. Engrg.* 6 (1995) 13–26.
- [13] C. Farhat, M. Lesoinne, P. LeTallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity, *Comput. Methods Appl. Mech. Engrg.* 157 (1998) 95–114.
- [14] C. Farhat, M. Lesoinne, N. Maman, Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution, *Int. J. Numer. Methods Fluids* 21 (1995) 807–835.
- [15] P.L. George, Improvement on delaunay based 3D automatic mesh generator, *Finite Elements Anal. Des.* 25 (1997) 297–317.
- [16] G.H. Golub, C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.
- [17] K.K. Gupta, Development of a finite element aeroelastic analysis capability, *J. Aircraft* 33 (1996) 995–1002.
- [18] H. Hassig, An approximate true damping solution of the flutter equation by determinant iteration, *J. Aircraft* 8 (1971) 885–889.
- [19] E.M. Lee-Rausch, J.T. Batina, Wing-flutter boundary prediction using unsteady Euler aerodynamic method, AIAA Paper No. 93-1422, 1993.
- [20] B. Van Leer, Towards the ultimate conservative difference scheme V: a second-order sequel to Godounov’s method, *J. Comput. Phys.* 32 (1979) 361–370.
- [21] M. Lesoinne, C. Farhat, Stability analysis of dynamic meshes for transient aeroelastic computations, AIAA Paper 93-3325, in: Proceedings of the 11th AIAA Computational Fluid Dynamics Conference, Orlando, Florida, 1993.
- [22] M. Lesoinne, C. Farhat, Re-engineering of an aeroelastic code for solving eigen problems in all flight regimes, in: P.P. Freidmann, M.P. Paidoussis (Eds.), *Fluid-Structure Interactions, Aeroelasticity, Flow-Induced Vibration and Noise*, ASME, AD-vol. 53(3), 1997, pp. 205–215. Also published as AIAA Paper 97-0647, 35th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, 6–9 January 1997.
- [23] S. Piperno, C. Farhat, B. Larroutourou, Partitioned procedures for the transient solution of coupled aeroelastic problems, *Comput. Methods Appl. Mech. Engrg.* 124 (1995).

- [24] J.R. Richardson, A more realistic method for routine flutter calculations, AIAA Symposium on Structural Dynamics and Aeroelasticity, Boston, Mass, 1965.
- [25] P.L. Roe, Approximate Reimann solvers, parameters vectors and difference schemes, *J. Comput. Phys.* 43 (1981) 357–371.
- [26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1995.
- [27] M. Sarkis, B. Koobus, A scaled and minimum overlap restricted additive Schwarz method with application to aerodynamics, *Comput. Methods Appl. Mech. Engrg.* 184 (2000) 391–400.
- [28] J. Steger, R.F. Warming, Flux vector splitting for the inviscid gas dynamic with applications to finite-difference methods, *J. Comput. Phys.* 40 (1981) 263–293.
- [29] E.C. Yates, Agard standard aeroelastic configuration for dynamic response, candidate configuration i. – wing 445.6, NASA TM-100492, 1987.