AIAA AVIATION Forum
5-9 June 2017, Denver, Colorado
35th AIAA Applied Aerodynamics Conference

10.2514/6.2017-4455

Development of Flutter Constraints for High-fidelity Aerostructural Optimization

Eirikur Jonsson*
Gaetan Kenway†
Joaquim R. R. A. Martins‡
University of Michigan, Ann Arbor, Michigan, United States
Graeme J. Kennedy§
Georgia Institute of Technology, Atlanta, Georgia, United States

High fidelity computational modeling and optimization of aircraft has the potential to allow engineers to produce more efficient designs requiring fewer unforeseen design modifications late in the design process. In order for the optimization algorithm to generate a useful design, all the relevant physics must be considered, including flutter. This is especially important for the high-fidelity aerostructural optimization of commercial aircraft, which is likely to result in wing designs that are prone to flutter. To address this issue, we developed a flutter constraint formulation suitable for gradient-based optimization. This paper investigates the feasibility of using a Doublet-Lattice Method (DLM) based flutter constraint for high-fidelity aerostructural optimization. The p-k flutter equation is solved using a determinant iterative method to obtain the eigenvalues. The Kreisselmeier–Steinhauser (KS) function is used to aggregate the damping values for individual modes into a single value that is used as the flutter constraint. To study the behavior of the flutter constraints using this method, we optimize a simple flat plate problem and perform a flutter analysis for a full transport aircraft using the uCRM configuration. We compute accurate and efficient derivatives for the DLM and the coupled derivatives with respect to structural sizing variables, as well as wing shape variables. These derivatives are computed using an automatic differentiation method and validated using the complex-step method. However, it is found that the current formulation using determinant iteration root finding method, is not adequate or robust, even for the simplest problems and reformulation is required.

I. Introduction

Aircraft design often requires high-fidelity computational modeling and optimization to accurately model all the relevant physical behavior in order to develop effective designs and reduce the occurrence of unforeseen design modifications during the later stages of development. In particular, for transonic wing design, the simultaneous optimization of both the aerodynamic design and the internal structure can yield significant reductions in fuel burn. However, all the relevant physics must be represented in the optimization problem, in order to generate a physically realizable design. For example, previous optimization results carried out by Kenway et al. [1, 2, 3], without flutter constraints produced wings with large aspect ratios as shown in Fig. 1. Such configurations are prone to flutter, which calls into question the usefulness of the results.

Since flutter is a safety and certification-critical phenomenon, it is important to be able to account for it early in the design process. Conservative design approaches may lead to excessively stiff and hence heavy designs, while unconstrained optimization approaches, such as those shown in Fig. 1, may lead to excessively flexible wings. Further, it is not unusual for flutter issues to be identified only at the final design and flight testing stages, at which point design changes are very costly. Therefore, we seek to model the flutter characteristics of the aircraft and account for them in the optimization process.

In the research community, a wing's flutter is often analyzed with a time-accurate coupled CFD-FEM solver similar to that proposed by [4]. However, this method incurs a high computational cost since hundreds if not thousands of time steps are required to simulate the flutter motion making it ill suited for optimization.

The current standard for flutter prediction in industry are methods based on panel codes, Doublet-Lattice Methods (DLM), and Transonic Small Disturbance (TSD) equations [5]. In the transonic regime there is a significant reduction

*PhD Candidate, AIAA student member
†Research Associate, AIAA Member
‡Professor, AIAA Associate Fellow
§Assistant Professor, School of Aerospace Engineering, AIAA Member

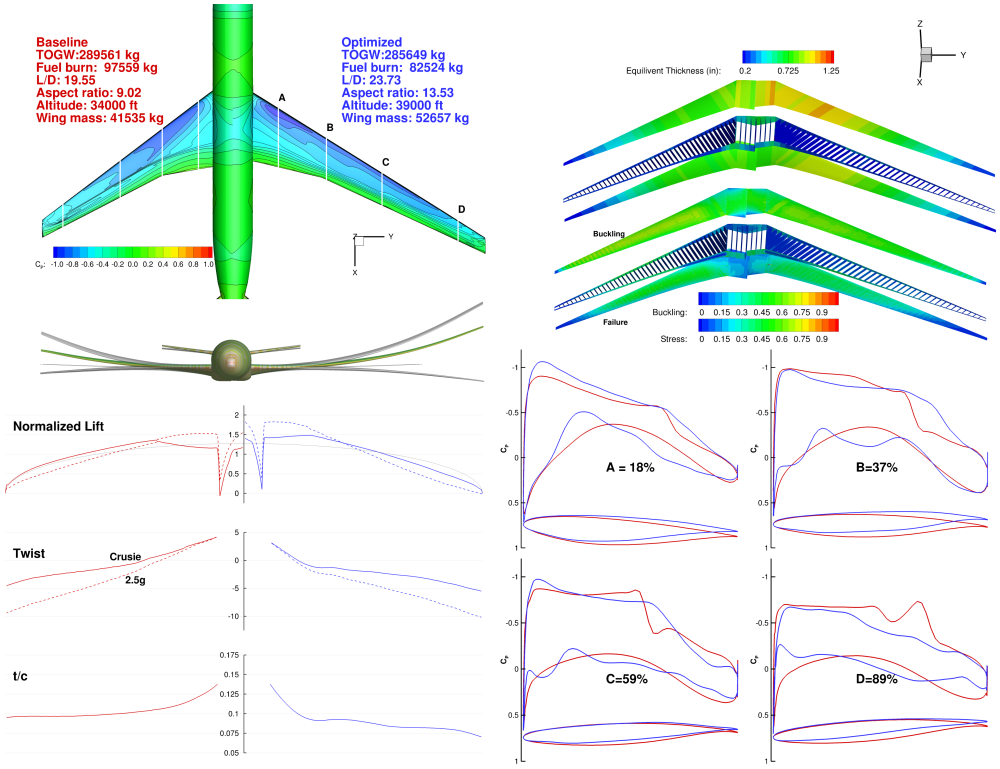

Figure 1. Aerostructural optimization result without flutter constraints [3]: Cp and planform comparison with initial design (upper left); equivalent thickness distribution, stress and buckling KS failure criteria (upper right); comparison of initial and optimized lift distributions, twist distributions and thickness to chord ratio (t/c) (lower left); four airfoils with corresponding Cp distributions (lower right). (notice the increased span ratio)

in the flutter speed, called the transonic dip or *flutter bucket*. Corrections using wind tunnel experimental data can be applied to augment the aerodynamic influence coefficients (AIC) matrix. However, for aerostructural design optimization this data is unavailable. Although other methods, such as full potential flow or the Euler models, are able to predict shock waves in the flow, they have certain limitations. As shown by several authors [6, 7, 8] viscous effects are necessary to predict flutter onset accurately.

Despite the aforementioned shortcomings of linear methods, the DLM method can be used to efficiently calculate flutter onset and thus can be incorporated as a flutter constraint in an otherwise high-fidelity aerostructural optimization. Using this approach, a high-fidelity CFD solver solving the RANS equations will account for nonlinearities in the flow solution, while the optimizer tries to smooth out any shock that may appear on the wing surface.

Stanford et al. [9, 10] implemented a flutter constraint for several mass minimizations of a transport wing. To retain some nonlinearities in the flow, a transonic AIC matrix was constructed from an Euler CFD code. This AIC matrix is however not updated during the optimization process. Further, the derivatives of the mode shapes with respect to design variables are neglected during the optimization. However, the mode shapes are updated at every design iteration. For a mass minimization problem these approximations may be acceptable [11], but varying planform and aspect ratio, these approximations may give unrealistic results.

The critical requirements in the context of our optimizations that have not been addressed so far are: 1) Efficient computation of the flutter derivatives, and 2) Computation of derivatives with respect to structural sizing variables, as well as wing shape variables including planform shape and area.

We implement the DLM method coupled to an FEM solver TACS [12, 13]. The entire code base is differentiated to give sensitivities of an aggregated damping value (flutter eigenvalue) with respect to the design variables. To compute the sensitivities accurately and efficiently we employ an Automatic Differentiation (AD) adjoint method. Sensitivities are verified using the complex-step method. We apply the developed flutter constraint to two geometries of interest, a flat plate verification problem and a transport wing. Further we optimize the plate geometry with respect to planform design variables and do a design space study for the uCRM.

II. Methods

There are several techniques and components necessary to enable flutter computations. In Fig. 2 the overall flutter analysis process is shown. The green boxes represent processes or components. The off diagonal terms are the variables that are passed between components. In the following section we will outline the major characteristics of these components. Further, we outline the theory necessary for predicting flutter.

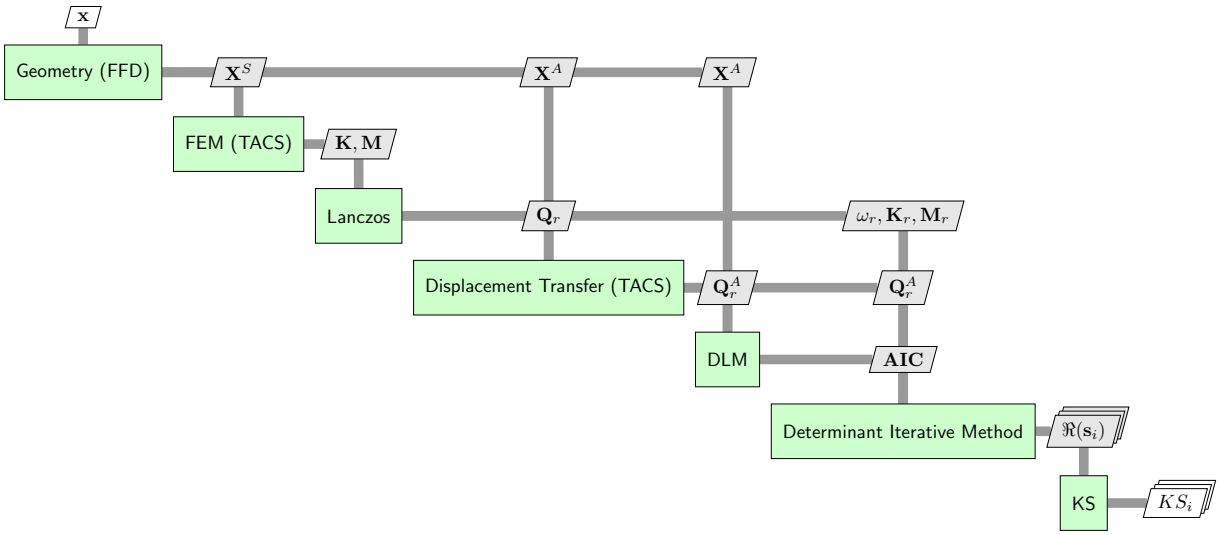

Figure 2. XDSM [36] of the flutter analysis process.

1. Geometric Parametrization

The shape of the wing is parametrized using a Free Form Deformation volume (FFD) approach [25]. This method has also been used with great success in computer graphics to deform solid geometries [26]. In this approach the geometry of interest is embedded within an FFD volume, which can be deformed using a number of control points. To create full-wing design variables such as span or sweep, the control points can be grouped together. Using this method, the structural and the aerodynamic meshes, $\mathbf{X}^S$, $\mathbf{X}^A$ respectively, can be handled the same way, minimizing the effort needed to update the internal structure as the optimizer changes the aerodynamic surface. A simple example of an FFD is shown in Fig. 4 where the red points represent the control points and the black lines connecting them represents the outer edges of the volume.

2. Finite Element Analysis Solver

The structural solver for this work is the Toolkit for the Analysis of Composite Structures (TACS) [12, 13]. TACS is a parallel finite-element solver with the capability to handle poorly conditioned problems, which is common in work involving thin-walled structures typically found in transport aircraft. For such cases, the stiffness matrix condition numbers may exceed $\mathcal{O}(10^9)$, but through the use of a Schur-complement based parallel direct solver, TACS is able to effectively solve these poorly conditioned problems. Sensitivities of structural functions of interest are also computed efficiently using the adjoint method with respect to structural and geometric design parameters.

3. Doublet Lattice Method

The doublet-lattice method (DLM) [14] consists of a lifting surface method that is formulated in the frequency domain. The DLM is based on the vortex-lattice method (VLM) where the VLM is extended to harmonically oscillating surfaces where a flat wake is assumed. A substantial body of literature exists on the DLM. An excellent reference worth mentioning is the work done by [15]. The DLM has been widely adopted in the aeroelastic community and has been a valuable tool for the flutter analysis of subsonic aircrafts. Commercial software tools such as MSC/NASTRAN have adopted the DLM [16]. In this work the implementation is based in part on the method of [14], and the extension by [17].

4. Flutter Analysis

In this work, we use a flutter analysis that takes the following form:

$$\mathbf{F}(s) = \left[s^2\mathbf{M}(\mathbf{x}) + \mathbf{K}(\mathbf{x}) - q_\infty \mathbf{A}(s)\right]\mathbf{u} = 0 \tag{4.1}$$

where $\mathbf{F}(s)$ is the overall flutter system, $\mathbf{M}(\mathbf{x})$ and $\mathbf{K}(\mathbf{x})$ are the mass and stiffness matrices from the finite-element equations, which are functions of the design variables $\mathbf{x}$. Furthermore, q_∞ is the dynamic pressure and $\mathbf{A}(s) = \mathbf{T}^T\mathbf{A}_{\text{IC}}(s)\mathbf{T}$, where $\mathbf{A}_{\text{IC}}$ is the aerodynamic influence coefficient matrix, and $\mathbf{T}$ is the load and displacement transfer interpolation matrix. The Laplacian parameter (eigenvalue) $s = \omega(\gamma + i)$ is complex and gives the frequency and damping of the motion. Note that the influence coefficient matrix is complex and is a nonlinear function of s. Therefore, the flutter equation (4.1) is a generalized nonlinear eigenvalue problem with a solution given by the triplet $(s, \mathbf{v}, \mathbf{u})$, where s is the complex eigenvalue and $\mathbf{v}$ and $\mathbf{u}$ are the left and right eigenvectors, respectively. The triplet satisfies the following equations:

$$\left[s^2\mathbf{M}(\mathbf{x}) + \mathbf{K}(\mathbf{x}) - q_\infty \mathbf{A}(s)\right]\mathbf{u} = 0$$
$$\mathbf{v}^H\left[s^2\mathbf{M}(\mathbf{x}) + \mathbf{K}(\mathbf{x}) - q_\infty \mathbf{A}(s)\right] = 0$$

4.1. Reduced Eigenproblem

Instead of using the full eigenvalue problem (4.1), flutter analysis techniques often employ a reduced eigenproblem using a small number of natural frequencies. The reduced modes are the eigenvectors of the problem:

$$\left[\mathbf{K}(\mathbf{x}) - \omega_i^2\mathbf{M}(\mathbf{x})\right]\mathbf{u} = 0,$$

where ω_i is the natural frequency. The eigenvectors $\mathbf{u}$ for $i = 1, \ldots, r$ are collected in the matrix $\mathbf{Q}_r \in \mathbb{R}^{n\times r}$ where n is the size of the square mass and stiffness matrices. These eigenvectors are $\mathbf{M}$-orthonormal, such that $\mathbf{Q}_r^T\mathbf{M}\mathbf{Q}_r = \mathbf{I}_r$. The reduced eigenproblem can now be written as follows:

$$\mathbf{F}_r(\tilde{s}) = \left[\tilde{s}^2\mathbf{M}_r + \mathbf{K}_r - q_\infty \mathbf{A}_r(\tilde{s})\right]\mathbf{u}_r = 0, \tag{4.2}$$

with the solution $(\tilde{s}, \mathbf{u}_r, \mathbf{v}_r)$. The reduced matrices take the form:

$$\mathbf{M}_r = \mathbf{Q}_r^T\mathbf{M}\mathbf{Q}_r = \mathbf{I}_r \in \mathbb{R}^{r\times r},$$
$$\mathbf{K}_r = \mathbf{Q}_r^T\mathbf{K}\mathbf{Q}_r = \text{diag}\{\omega_i^2\} \in \mathbb{R}^{r\times r},$$
$$\mathbf{A}_r(\tilde{s}) = \mathbf{Q}_r^T\mathbf{A}(\tilde{s})\mathbf{Q}_r \in \mathbb{C}^{r\times r}.$$

Note that $\mathbf{A}_r$ has no sparsity structure and is a dense matrix in general, while $\mathbf{M}_r$ and $\mathbf{K}_r$ are diagonal.

4.2. Solution method for the Generalized Reduced Eigenvalue problem

In this work, we use a shift and invert Lanczos method, which is a generalized eigenvalue solution algorithm. The Lanczos algorithm extracts eigenvalues for symmetric generalized eigenvalue problems. Here, we use this algorithm to solve for the natural frequencies of the structural problem without aerodynamic loads:

$$\mathbf{K}\mathbf{u} = \lambda\mathbf{M}\mathbf{u}.$$

Instead of solving this problem directly, we use a shift and invert strategy to zero-in on the desired spectrum to reduce the number of iterations required. This strategy increases the computational cost of each iteration, but reduces the overall cost to solve the eigenproblem. The shift and invert technique produces the following eigenproblem that has the same eigenvectors but transformed eigenvalues [18, 19]

$$\mathbf{M}(\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{M}\mathbf{u} = \mu\mathbf{M}\mathbf{u},$$

where the transformed eigenvalue μ is related to the original eigenvalue λ through the relationship:

$$\mu = \frac{1}{\lambda - \sigma}.$$

When σ is chosen such that it lies close to the desired λ, the corresponding transformed eigenvalues, μ, become well separated, making the Lanczos algorithm more efficient.

The Lanczos algorithm uses an $\mathbf{M}$-orthonormal subspace, written as $\mathbf{V}_m \in \mathbb{R}^{n\times m}$, such that $\mathbf{V}_m^T\mathbf{M}\mathbf{V}_m = \mathbf{I}_m$. In exact arithmetic, this subspace can be formed directly from the Lanczos three-term recurrence. However, the resulting subspace loses orthogonality as the algorithm converges to an eigenvalue due to numerical truncation errors. Instead, we use an expensive, but effective, full-orthonormalization procedure (Gram–Schmidt) that enforces $\mathbf{M}$-orthonormality. The Lanczos method can be easily extended to find multiple eigenpairs $(\lambda_i, \mathbf{u})$ by using multiple Ritz values.

Lanczos method for computing eigenvalues and eigenvectors of $\mathbf{K}\mathbf{u} = \lambda\mathbf{M}\mathbf{u}$
Given: m, $\hat{\mathbf{v}}_1$, σ, ϵ_{tol}
Factor the matrix $(\mathbf{K} - \sigma\mathbf{M})$
Set $i = 1$
while $i \leq m$ **do**
  $\hat{\mathbf{v}}_{i+1} = (\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{M}\mathbf{v}_i$
  Set $j = 1$
  while $j \leq i$ **do** ▷ Full $\mathbf{M}$-orthonormalization
    $h_{ji} = \mathbf{v}_j^T\mathbf{M}\hat{\mathbf{v}}_{i+1}$
    $\hat{\mathbf{v}}_{i+1} \leftarrow \hat{\mathbf{v}}_{i+1} - h_{ji}\mathbf{v}_j$
    $j \leftarrow j + 1$
  end while
  $\alpha_i \leftarrow h_{ii}$
  $\beta_i = \sqrt{\hat{\mathbf{v}}_{i+1}^T\mathbf{M}\hat{\mathbf{v}}_{i+1}}$
  $\mathbf{v}_{i+1} = \hat{\mathbf{v}}_{i+1}/\beta_i$
  $\mathbf{T}_i = \text{tridiag}_k\{\beta_{k-1}, \alpha_k, \beta_k\}$ ▷ Solve the reduced eigenproblem
  Solve $\mathbf{T}_i\mathbf{y}_i = \theta\mathbf{y}_i$ for $(\theta, \mathbf{y}_i)$
  if $\beta_i\mathbf{y}_i^T\mathbf{e}_i < \epsilon_{tol}$ **then** ▷ Test for convergence
    $\mathbf{u} = \mathbf{V}_i\mathbf{y}_i$
    $\lambda = \frac{1}{\theta} + \sigma$
    break
  end if
  $i \leftarrow i + 1$
end while

4.3. Flutter solution method

The flutter solution algorithm that we use is given by [20], which solves the reduced nonlinear eigenvalue problem Eq. (4.2). This method is a secant method applied to the determinant equation:

$$\Delta(s) = \det \mathbf{Q}_r^T\mathbf{F}(s)\mathbf{Q}_r.$$

Note that the columns of $\mathbf{Q}_r \in \mathbb{R}^{n\times r}$ are the eigenvectors from the natural frequency eigenproblem. Given initial guesses s_1, and s_2, the method computes s_{k+2} as follows:

$$s_{k+2} = \frac{s_{k+1}\Delta(s_k) - s_k\Delta(s_{k+1})}{\Delta(s_k) - \Delta(s_{k+1})},$$

the iteration is continued until $|\Delta(s_{k+2})| \leq \epsilon_{tol}$ for some specified tolerance. In this implementation all flutter eigenvalues s, at a velocity (or dynamic pressure) range of interest, are found for mode i first before advancing to the next mode.

When evaluating the flutter determinant, the Aerodynamic Influence Coefficient (AIC) matrix is not evaluated explicitly but rather interpolated using 4$^{\text{th}}$ order b-splines from a precomputed AIC matrices at a range of reduced frequencies. These AIC matrices are precomputed at startup for a specified range of reduced frequencies where the upper limit is the maximum reduced frequency k_{max} of the reduced problem. The number of evaluation points for this range is chosen a priori and is equally spaced over the selected range. This procedure is commonly applied for an AIC method like the DLM and will improve the overall efficiency of the code as the evaluation of the AIC matrix is generally an expensive operation. Although the AIC matrix is independent of the structures, a new set of precomputed

AIC matrices are constructed for each optimization iteration as the planform of the geometry, hence the aerodynamic mesh, is changed or updated.

4.4. Flutter Constraint Aggregation

The Kreisselmeier–Steinhauser (KS) function [21, 22, 23] can be used to aggregate constraints into a single composite function. The function is C_1 continuous and gives an estimate of the maximum for a given set of constraints. The KS function can be written as:

$$KS(\mathbf{g}(\mathbf{x})) = \frac{1}{\rho} \ln \left(\sum_{j=1}^{m} e^{\rho g_j(\mathbf{x})} \right) \tag{4.3}$$

where $\mathbf{g}(\mathbf{x})$ are set of constraints at a design point $\mathbf{x}$ and ρ is the KS parameter. This parameter can be used as a buffer or tolerance and is analogous to a penalty parameter used in constrained optimization. As $\rho \to \infty$ the KS function approaches the maximum, but too large a value can cause sharp changes in the gradient. To avoid numerical difficulties due to overflow we use an alternate form of the KS function

$$KS(\mathbf{g}(\mathbf{x})) = g_{\max}(\mathbf{x}) + \frac{1}{\rho} \ln \left(\sum_{j=1}^{m} e^{\rho(g_j(\mathbf{x}) - g_{\max}(\mathbf{x})} \right) \tag{4.4}$$

where $g_{\max}(\mathbf{x})$ is the maximum of all constraints evaluated at current design $\mathbf{x}$ and is taken to be constant. As for the value, the first derivative with respect to the design variables $\mathbf{x}$ of both forms should be equal, subject to finite precision arithmetics.

$$\frac{\partial KS(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}} = \frac{\sum_{j=1}^{m} e^{\rho g_j(\mathbf{x})} \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}}}{\sum_{j=1}^{m} e^{\rho g_j(\mathbf{x})}} \tag{4.5}$$

$$\frac{\partial KS(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}} = \frac{\sum_{j=1}^{m} e^{\rho(g_j(\mathbf{x}) - g_{\max}(\mathbf{x}))} \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}}}{\sum_{j=1}^{m} e^{\rho(g_j(\mathbf{x}) - g_{\max}(\mathbf{x}))}} \tag{4.6}$$

In this work, for each mode i, the (KS) function is used to aggregate the real parts of all the flutter eigenvalues $\Re(\mathbf{s}_i)$, which are obtained over a specified range of dynamic pressures, into one value, KS_i. This reduces the number of constraints to the number of modes used in the analysis process. For the geometry under investigation to be flutter free the KS value has to be less than zero for all modes ($KS_i < 0$).

4.5. Code Verification

Here we present a brief verification study of the DLM code and the methods used in our flutter analysis. Natural frequencies of the flat plate problem presented in Section IV are obtained using the Lanczos method. The subspace size is 10, and the first 4 modes are extracted and used. A comparison with MSC/NASTRAN is shown in Table 1, where the overall agreement is good.

Table 1. Flat plate natural frequencies agree well with NASTRAN. Units presented here are radians.

Mode number	NASTRAN	TACS	Error (%)
1	7.130	7.134	0.058
2	44.575	44.693	0.263
3	57.704	58.268	0.969
4	125.070	125.838	0.610

Verification of the flutter analysis code is performed using the uCRM wingbox presented in Section V. Specifically, in Fig. 3, we compare the flutter eigenvalues for the first eight modes of the structure with MSC/NASTRAN. Overall trends of our implementation are good although some discrepancy is present. This is likely due to several reasons such as: (1) The precomputed AIC matrix could have a different resolution when constructed (evaluated for a different

number of reduced frequencies), resulting in a discrepancy when the AIC is interpolated for a specific reduced frequency. (2) The use of a parabolic approximation kernel for the AIC matrix rather than the quartic approximation [17] found in MSC/NASTRAN, and the different flutter equation formulation used in MSC/NASTRAN [24].

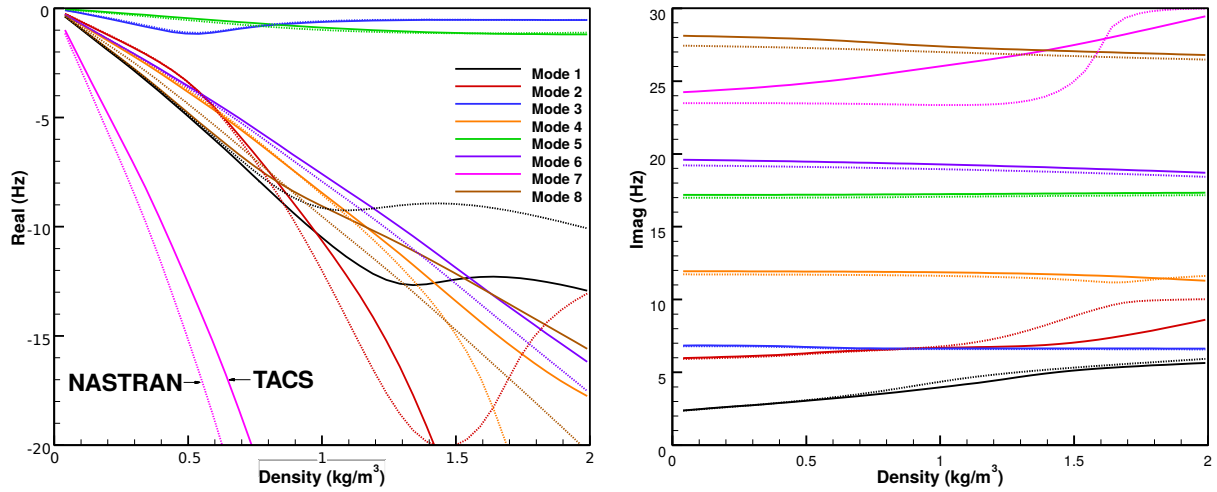

Figure 3. Real and imaginary parts shown for the flutter eigenvalue of a CRM wingbox. Solid lines show results from this proposed work, with the DLM and TACS flutter implementation, and the dashed lines are from MSC/NASTRAN aeroelastic module.

5. Optimization Algorithm

The optimization routine used in this work is SNOPT (Sparse Nonlinear OPTimizer) [27]. SNOPT is a gradient based optimizer that implements a sequential quadratic programming (SQP) algorithm. SNOPT uses an augmented Lagrangian merit function, and the Hessian of the Lagrangian is approximated using a quasi-Newton approach. This optimizer is designed to perform well for optimization problems featuring many sparse nonlinear constraints and it requires a low number of function calls. SNOPT is wrapped with a sparse implementation of pyOpt [28].

III. Derivative implementation for the flutter problem

To produce accurate gradient information, the code is analytically as well as automatically differentiated (AD). We now describe the AD approach that we used in this work.

1. Automatic Differentiation (AD)

Automatic differentiation, also known as algorithmic differentiation, is a well established method that systematically applies the differentiation chain rule to source code. This method uses source transformation tools that takes in the original computer program, augments it, and generates a new code, such that it computes the analytical derivatives along with the original program [29, 30]. Two modes exist, the forward mode and the reverse mode. For a generic system with scalar input x and output y we can write it as:

$$\begin{aligned} \text{System} \quad & x \rightarrow \boxed{F(x)} \rightarrow y \\ \text{Forward AD} \quad & \dot{x} \rightarrow \boxed{F'(x)} \rightarrow \dot{y} \\ \text{Reverse AD} \quad & \bar{x} \leftarrow \boxed{F'^{*}(x)} \leftarrow \bar{y} \end{aligned}$$

where the arrows represent the flow of information, the box represents the system or a function. The forward mode, know as the tangent, is denoted with a dot ($\dot{}$) over the variable. Given some small variations on the input (independent) variables $\mathbf{x}$ we can compute the resulting variations of the dependent variables $\mathbf{y}$. The Jacobian matrix $\mathbf{J}$ contains the partial derivatives of each output (dependent) variable y_j with respect to each independent variable x_i.

The forward mode thus computes $d\mathbf{y} = \mathbf{J}d\mathbf{x}$ for each given $d\mathbf{x}$ or

$$d\mathbf{y} = \mathbf{J}d\mathbf{x}$$
$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_m \end{bmatrix}$$

Conversely the reverse mode, known as the adjoint, is denoted by a bar ($\bar{}$) over the variable. The order of operations reverses and we compute the transposed Jacobian product $d\mathbf{x} = \mathbf{J}^* d\mathbf{y}$ for each given $d\mathbf{y}$ or

$$d\mathbf{x} = \mathbf{J}^* d\mathbf{y}$$
$$\begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \frac{\partial y_2}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_m \end{bmatrix}$$

In other words, the gradient of the independent variable is a linear combination of the variation in the dependent variable. This is a very important observation particularly in with fewer output variables than input variables. Since many aerodynamic optimization formulations contain many more design variables than outputs of interest (or $n_x >> n_I$) we use the reverse mode or the adjoint mode in order to obtain the derivatives as efficiently as possible. The AD source-transformation tool, Tapenade [31, 32] is used in this work.

2. Flutter Derivatives

A simplified flow of information for the flutter calculation, obtaining derivatives in forward and reverse mode can be written as:

$$\mathbf{x} \rightarrow \mathbf{K}, \mathbf{M} \rightarrow \mathbf{Q}_r, \mathbf{K}_r, \mathbf{M}_r \rightarrow \mathbf{s}_i \rightarrow KS_i$$
$$\dot{\mathbf{x}} \rightarrow \dot{\mathbf{K}}, \dot{\mathbf{M}} \rightarrow \dot{\mathbf{Q}}_r, \dot{\mathbf{K}}_r, \dot{\mathbf{M}}_r \rightarrow \dot{\mathbf{s}}_i \rightarrow \dot{KS}_i$$
$$\overline{\mathbf{x}} \leftarrow \overline{\mathbf{K}}, \overline{\mathbf{M}} \leftarrow \overline{\mathbf{Q}}_r, \overline{\mathbf{K}}_r, \overline{\mathbf{M}}_r \leftarrow \overline{\mathbf{s}}_i \leftarrow \overline{KS}_i$$

where $\mathbf{K}, \mathbf{M}$ are the stiffness and mass matrices, and $\mathbf{Q}_r$ are the reduced natural modes shapes. KS_i represents the real part of flutter eigenvalues $\mathbf{s}_i$ for mode i aggregated over the velocity range interest using the KS function defined in Eq. (4.4). Note that in reverse the $\overline{\mathbf{K}}, \overline{\mathbf{M}}$ matrices are not explicitly computed, but the derivatives are directly accumulated onto the design variables.

Total derivatives of all flutter constraints $d\mathbf{KS}/d\mathbf{x}$ is generated with a combination of AD and analytically differentiated code. Routines such as LAPACK's [33] complex and real linear solves, and determinant, are not automatically differentiated, but instead must be differentiated analytically and implemented as such.

3. Derivative Verification

To demonstrate correct and accurate derivatives we perform a rigorous verification. Each function in the code is unit tested where sensitivities are computed using a second order central finite-difference stencil, a complex-step method [34] as well as forward and reverse mode AD. The finite-difference stencil used here is

$$\frac{dI}{dx} = \frac{I(x+h) - I(x-h)}{2h} + O(h^2),$$

with a step size ranging from $h = 10^{-3}$ to $h = 10^{-6}$ depending on the function under consideration. Note that in order to get a reasonable prediction with finite-difference, a step size study was performed to get the most accurate gradient possible. For the complex-step method the sensitivity of a function is computed as

$$\frac{dI}{dx} = \frac{\Im[I(x+ih)]}{h} + O(h^2),$$

where $i = \sqrt{-1}$ and a step size of $h = 10^{-40}$ is used. The complex-step method does not suffer from a subtractive cancellation errors unlike the finite-difference method. The step size can be made very small, hence the $O(h^2)$ truncation error becomes negligible.

One drawback of the complex-step method is that it cannot be used without modifying programs that already contain complex numbers and perform complex arithmetic. Such programs need to be modified such that the complex number is represented by two real numbers, one for the real part and one for the imaginary part. Complex arithmetic thus needs to be performed using manually defined functions and cannot be done using intrinsic functions. In this work, all code has been modified such that it utilizes only real numbers for complex calculations and is complex-step safe.

Since we have many more design variables than functions of interest $I(x)$, we want to employ the adjoint or the reverse mode AD in the optimization. One technique to verify the reverse mode is to implement the forward mode and perform a *dot product test*. The forward mode and the complex-step should match close to machine precision so implementing the forward mode as well is an important step. The *dot product test* [35] can be written as:

$$\begin{aligned}
\bar{\mathbf{x}}^* \dot{\mathbf{x}} &= \left(\mathbf{J}^* \bar{\mathbf{y}}\right)^* \dot{\mathbf{x}} \\
&= \bar{\mathbf{y}}^* \left(\mathbf{J} \dot{\mathbf{x}}\right) \\
&= \bar{\mathbf{y}}^* \dot{\mathbf{y}} \qquad set \quad \bar{\mathbf{y}} = \dot{\mathbf{y}} \\
&= \dot{\mathbf{y}}^* \dot{\mathbf{y}}
\end{aligned}$$

This equality should be exact to machine precision.

3.1. Intermediate derivatives

We now present derivative results of the flutter constraint with respect to the reduced stiffness matrix $\mathbf{K}_r$, the aerodynamic mesh nodes $\mathbf{X}^A$ and $\mathbf{Q}_r^A$ which are the reduced mode shapes, $\mathbf{Q}_r$, transferred on to the aerodynamic mesh. As shown in Table 2 sensitivities are very accurate as the complex-step and the reverse AD agree very well. As expected, the second order finite difference method does not perform as well, and is sensitive to variation in step size. In order to get the best finite difference derivative the step size was varied from $h = 10^{-3}$ to $h = 10^{-6}$ depending on which derivative was being calculated.

3.2. Flat plate derivatives

Sensitivities for the flat plate, geometry presented in Section IV are considered. Sensitivities for the design variable of interest, chord, span and thickness are shown in Table 3. The thickness variable is for the entire plate as opposed to one thickness variable per element. As before, finite difference offers less accuracy compared to AD or complex-step. Note that the number of matching digits for the full derivative chain is somewhat less than what is presented in Table 2. Once passed through the reverse implementation of TACS and the Lanczos method some accuracy is lost. The reasons behind this are not obvious at this point and need further investigation.

Table 2. Intermediate sensitivities of the constraint on mode 1, KS_1, with respect to a single value in the reduced stiffness $\mathbf{K}_r$ matrix, aerodynamic mesh points $\mathbf{X}^A$ matrix and the reduced transferred mode shapes $\mathbf{Q}_r^A$. Reverse AD compares very well to complex-step and is close to matching machine precision. Finite difference however lacks accuracy compared to the other methods.

	$\frac{\partial KS_1}{\partial \mathbf{K}_r}$	$\frac{\partial KS_1}{\partial \mathbf{X}^A}$	$\frac{\partial KS_1}{\partial \mathbf{Q}_r^A}$
Finite Difference	0.004366413230982**9**	0.105889599089081**8**	-0.023006325067243**0**
Complex-step	0.004366413545420**5**	0.105889598234296**1**	-0.023006327351353**4**
AD (Reverse)	0.004366413545414**2**	0.105889598234296**3**	-0.023006327351352**3**

3.3. uCRM derivatives

Here we present sensitivities for the uCRM geometry presented in Section V. Sensitivities for the KS function with respect to selected variables are listed in Table 4. As before, the finite difference values are obtained using a second order stencil. A step size study would be desirable for each design variable or element, but this was not done due to the large number of variables. Instead, the step size that gave the best overall results was chosen, which is $h = 10^{-6}$. The

Table 3. Sensitivities of constraint on mode 1, KS_1, with respect to the design variables, chord, span and plate thickness.

	$\frac{dKS_1}{dx_{\text{chord}}}$	$\frac{dKS_1}{dx_{\text{span}}}$	$\frac{dKS_1}{dx_{\text{thickness}}}$
Finite Difference	1.59234199	-2.55801140	3930.43055292
Complex-step	1.59403748	-2.56075992	3929.75519514
AD (Reverse)	1.59481803	-2.56035533	3929.74618753

last two columns give the relative error between the finite difference and complex-step, and reverse AD and complex-step. We note that the reverse AD and complex-step compare very well as the relative error is three to six orders of magnitude smaller than compared to finite difference.

Table 4. Derivatives of constraint on mode 1, KS_1, versus the design variables for 8 spars, 8 ribs and 8 skin elements of the structure. RAD is reverse AD, FD is finite differencing, and CS is the complex step.

Variable	i	FD	CS	RAD	Relative error FD-CS	Relative Error RAD-CS
Spar Elements	1	-2.014985468	-1.999974758	-1.999965907	7.51E-003	4.43E-006
	2	-0.088696547	-0.090194566	-0.090198427	1.66E-002	4.28E-005
	3	-0.095666766	-0.085257553	-0.085258866	1.22E-001	1.54E-005
	4	0.203193368	0.210156947	0.210212359	3.31E-002	2.64E-004
	5	0.263933667	0.259278622	0.259260509	1.80E-002	6.99E-005
	6	-0.071987279	-0.032305006	-0.032318974	1.23E+000	4.32E-004
	7	0.006071139	-0.000334429	-0.000327057	1.92E+001	2.25E-002
	8	-0.020179093	0.000032238	0.000026615	6.27E+002	2.11E-001
Rib Elements	1	0.253309222	0.224356643	0.224359214	1.29E-001	1.15E-005
	2	0.191743890	0.175807433	0.175809351	9.06E-002	1.09E-005
	3	0.207341327	0.230292459	0.230290811	9.97E-002	7.16E-006
	4	0.185916602	0.181973005	0.181975321	2.17E-002	1.27E-005
	5	0.237979611	0.233055462	0.233046967	2.11E-002	3.65E-005
	6	0.181381180	0.185529844	0.185526430	2.24E-002	1.84E-005
	7	0.231692141	0.231781620	0.231775388	3.86E-004	2.69E-005
	8	0.150392537	0.184741162	0.184729822	1.86E-001	6.14E-005
Skin Elements	1	0.441049044	0.434946729	0.434919882	1.40E-002	6.17E-005
	2	0.276180216	0.287471575	0.287450964	3.93E-002	7.17E-005
	3	0.247139578	0.285488076	0.285466885	1.34E-001	7.42E-005
	4	0.470635907	0.436378482	0.436469380	7.85E-002	2.08E-004
	5	0.328335009	0.287666266	0.287733113	1.41E-001	2.32E-004
	6	0.291624919	0.285354574	0.285422324	2.20E-002	2.37E-004
	7	0.485841301	0.477884284	0.477930152	1.67E-002	9.60E-005
	8	0.306040695	0.312406435	0.312433467	2.04E-002	8.65E-005

IV. Flat plate flutter analysis and optimization

1. Model description

The flat plate geometry shown in Fig. 4 is chosen for verification purposes due to its simplicity and short optimization turnaround. The flat plate structure, shown in red is embedded within a larger flat aerodynamic mesh, shown in gray. The red circles are control points of the FFD volume, which both meshes have been embedded in. The structural model consists of 12 elements in the streamwise direction and 40 elements in the spanwise direction a total of 480 finite elements. The finite element used here is the MITC4 shell element. The aerodynamic model consists of 10 elements in the streamwise direction and 15 elements in the spanwise direction. Material properties, dimensions, and discretization for the baseline flat plate are summarized in Table 5.

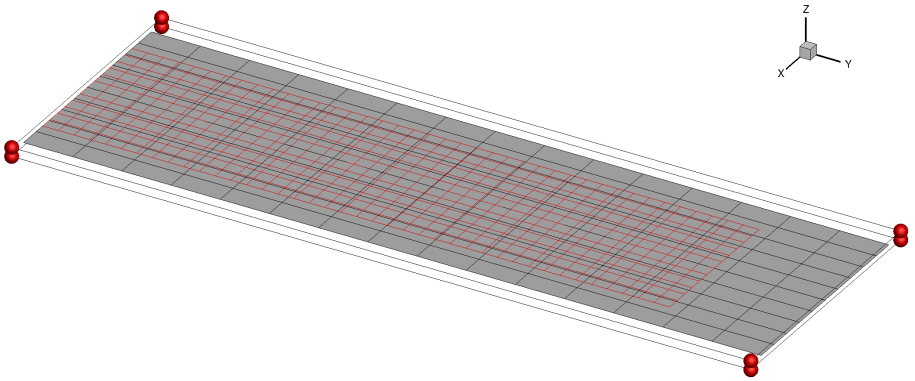

Figure 4. Flat plate structural and aerodynamic mesh shown in red and black respectively. The plate is cantilevered at the left edge. A Free-Form-Deformation (FFD) volume is also shown with 8 control points which are depicted as red spheres connected by solid black lines.

Table 5. Flat plate mechanical properties, dimensions, and discretization of the structure and the aerodynamic surface.

Variable	Symbol	Value
Density	ρ_s	2800 kg/m^3
Modulus of Elasticity	E	70 GPa
Poisson ratio	ν	0.3
Yield stress	σ_y	400 KPa
Thickness	t	0.002 m
Structure Span	b_s	0.85 m
Structure Chord	c_s	0.21 m
Number of Finite elements, streamwise	n_x	12
Number of Finite elements, spanwise	n_y	40
Span	b	1.0 m
Chord	c	0.3 m
Number of DLM elements, streamwise	n_x	10
Number of DLM elements, spanwise	n_y	15
Planform area	A_{init}	0.3 m^2

For the flat plate analysis and optimization the air density is kept fixed and the Mach number is set to zero (incompressible flow). The velocity range is varied in 40 increments from 2 to 15 m/s for each design. The flight conditions are summarized in Table 6. The flutter problem is solved at each velocity (dynamic pressure effectively) for each mode. In this work we constrain the first 4 modes of the plate. The first 4 natural modes and mode shapes are calculated using the Lanczos algorithm with a subspace of size 10.

Table 6. Flat plate operating conditions under investigation. The airspeed is incremented from 2 to 15 m/s in 40 increments.

Variable	Symbol	Value
Mach	M_∞	0
Lift Coefficient	C_L	0.5
Air Density	ρ_∞	1.225 kg/m^3
Air Speed Range	U_∞	2-15 m/s
Numer of Speed Increments	n_U	40
Flight speed (range eqn.)	V	1.0 m/s
Thrust specific fuel consumption	c_T	1.0 lb/(lbf $\cdot$ h)
Fixed weight	W_{fixed}	1.0 kg
Fuel weight	W_{fuel}	0.25 kg

2. Problem statement

For this simple flat plate problem, the design variables are thickness of the entire plate $x_{\text{thickness}}$, span x_{span} and chord length x_{chord}. No sweep, taper or dihedral is applied here. The objective is to maximize range using the Breguet range equation,

$$R = \frac{V}{c_T}\frac{C_L}{C_D}\ln\left(\frac{W_{\text{init}}}{W_{\text{final}}}\right) \tag{2.1}$$

where R is range, V/c_T is the flight speed to thrust-specific fuel consumption ratio, C_L/C_D is the lift to drag ratio, and $W_{\text{init}}/W_{\text{final}}$ is the initial to final cruise weight ratio. For simplicity we assume that $V/c_T = 1$, and we define the cruise weights as

$$\begin{aligned} W_{\text{final}} &= W_{\text{fixed}} + W_{\text{plate}} \\ W_{\text{init}} &= W_{\text{final}} + W_{\text{fuel}} \end{aligned} \tag{2.2}$$

where the W_{fixed} is a fixed weight and W_{fuel} is the fuel weight. The lift coefficient is fixed at $C_L = 0.5$ and the drag coefficient is calculated assuming it consists only of the lift induced drag.

$$C_D = \frac{C_L^2}{\pi e AR} \tag{2.3}$$

where the wing span efficiency factor is set to $e = 1$ for simplicity. AR is the aspect ratio defined as $AR = b^2/S$ where b is the reference span and S is the planform area. The range can be increased by reducing the drag coefficient and by reducing the thickness of the plate. The drag coefficient is reduced by increasing the aspect ratio as other parameters are fixed.

The chord and span directly affect the aspect ratio so we want to formulate the problem in terms of the aspect ratio rather than directly the chord and the span. By adding an area equality constraint we ensure that there is a link between the chord and span. This will reduce the number of design variables making this in effect a problem with two design variables, thickness and aspect ratio. This allows us to make contour plots that can give valuable insight into the design space. Flutter constraints are added on the first 4 modes such that $KS_i \leq 0$ for $i = 1, \ldots, 4$ and the planform area must be kept constant. Due to the primitive nature of the flutter constraint applied here, no minimum flutter speed or flutter point is defined explicitly. The constraint forces the geometry to be flutter free for the entire flight envelope. For this particular problem no flutter must occur for the velocity range of 2-15 m/s. One could then think of the minimum flutter point to be 15 m/s as no constraints are enforced for higher velocities. The initial area is given in Table 5. The side constraints are as follows, chord and span are specified such that the aspect ratio is allowed to vary from $1 \leq AR \leq 6$, and the material thickness of the plate is allowed to vary from $0.0012 \leq t \leq 0.0025$

m. The optimization problem is summarized in Table 7 and Fig. 5 shows the flutter analysis implementation as it is applied in the optimization.

Table 7. Optimization formulation of the flat plate problem

	Function/variable	Description	Quantity
maximize	Range	Breguet equation	
with respect to	x_{span}	Plate span	1
	x_{chord}	Plate chord	1
	$x_{\text{thickness}}$	Plate thickness	1
		Total design variables	**3**
subject to	A - $A_{\text{init}} = 0.0$	Fixed plate area	1
	$\text{KS}_i \leq 0$	KS aggregate of damping values for modes $i = 1, \ldots, 4$	4
		Total constraints	**5**

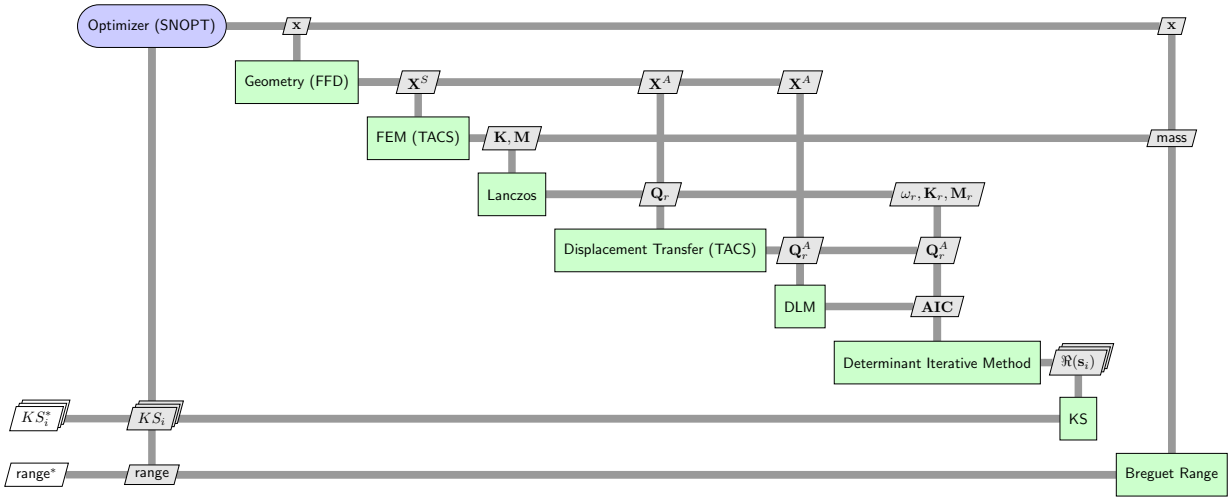

Figure 5. XDSM [36] showing the flutter constraint as applied in the optimization.

3. Design space analysis

Before running an optimization, we analyze the design space since we have the luxury of plotting the design space. A contour plot is generated by sweeping over both the aspect ratio range $1 \leq AR \leq 6$ and the thickness range $0.0012 \leq t \leq 0.0025$ m. Due to a relatively low cost of each analysis, a grid of 32 steps in each variable is used giving a total of 1024 points. Figure 6 shows the contour plot of the objective function where the four constraints have been applied to the contour, one at a time. The objective function appears smooth with a clear maximum at $AR = 6, t = 0.0012$ m as expected. For each flutter constraint where $KS_i > 0$, the objective function value has been blanked out as this part of the design space is infeasible. Constraints boundaries on modes 1,2,3, and 4 are shown with black, blue, red, and purple curves, respectively. For mode 1 there appear small pockets of infeasible region in the design space. This anomaly will be addressed later. For the other modes 2-4 are represented as continuous lines in the design space.

All constraints have been applied to the objective function contour plot in Fig. 7. The constraint pockets appearing on mode 1 will never be active as these scattered regions are blanked out by mode 2 and 3. Similarly, constraint boundaries for modes 2 and 3 appearing in the low aspect ratio region will never be active as mode 4 will blank them

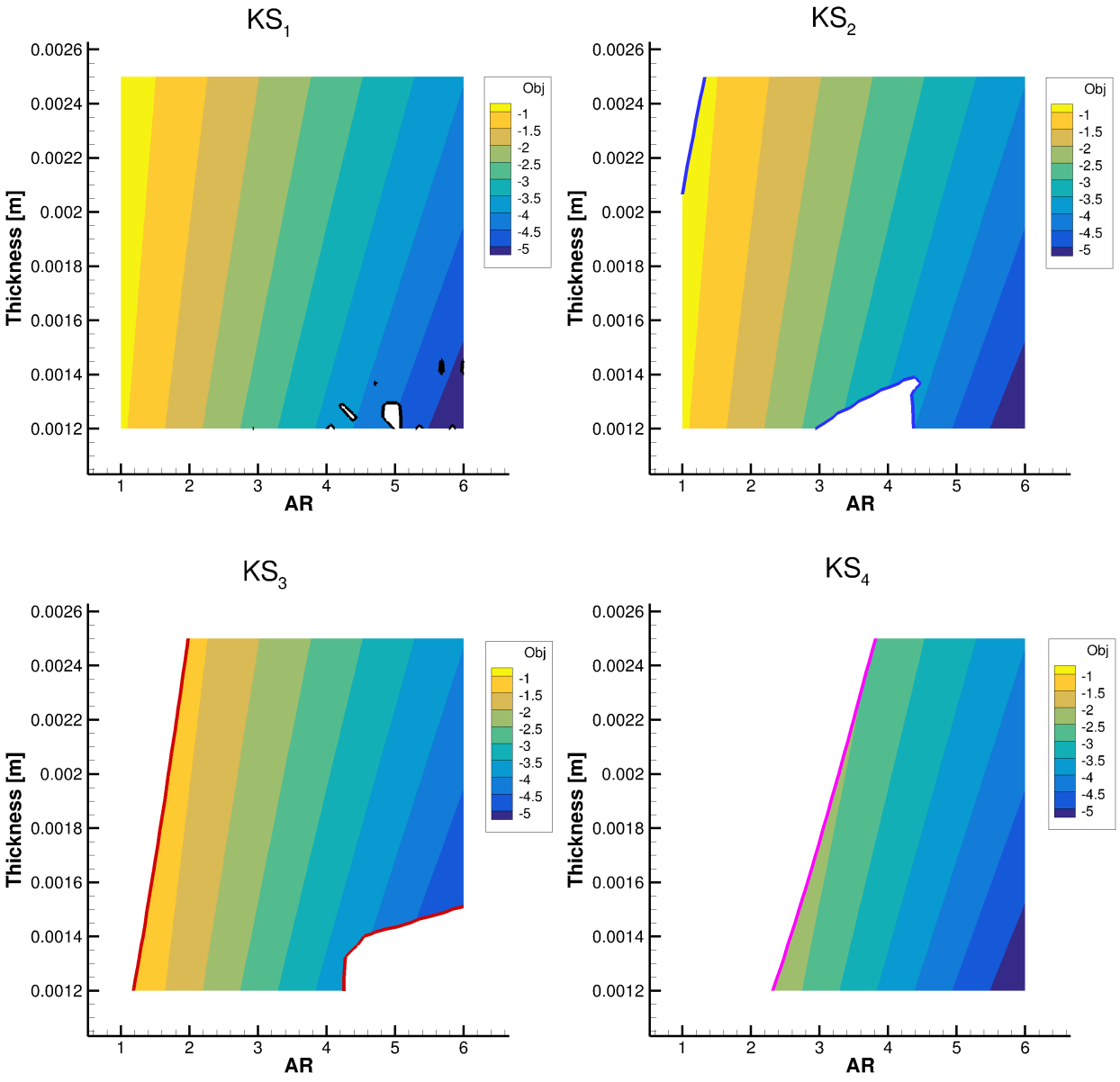

Figure 6. Contour plot of the objective function shown with the four flutter constraints applied to the contour plot. Blanked out regions represent values of where the constraint is violated or $KS_i > 0$. To generate the contour, the design space is sampled using 32 points in both variables for a total of 1024 design points.

out. Hence, modes 2, 3, and 4 are the only needed constraints for this optimization as mode 1 will never be active. After applying the constraints to the design space the optimum is at $AR^* \approx 6, t^* \approx 0.0015$.

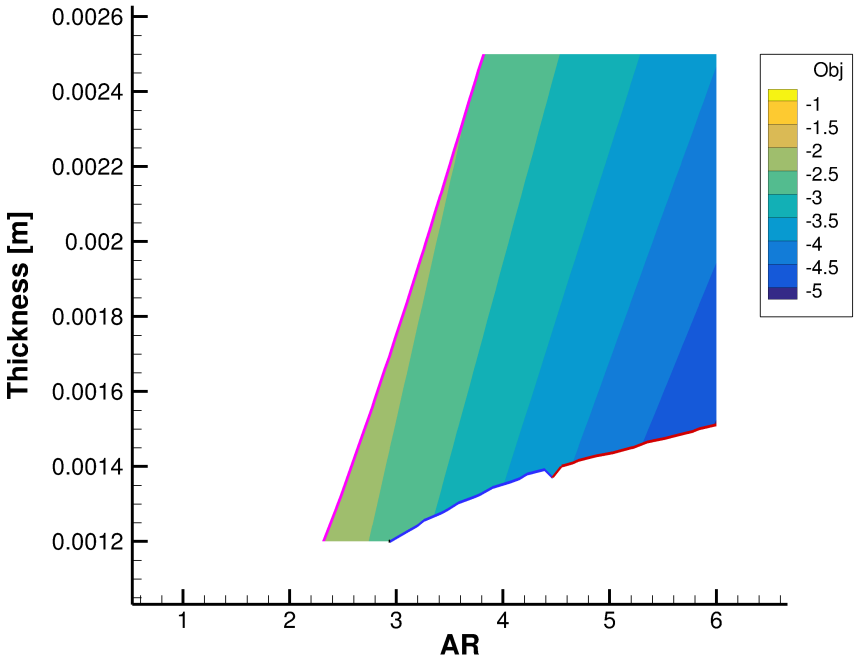

Figure 7. Objective function shown where the infeasible design space has been blanked out by the four flutter constraints. Constraints on modes 2, 3 and 4, (blue, red and purple respectively), are the ones that enclose the feasible design space. The mode 1 constraint (black) is blanked out by either constraint 2 or 3 and thus will never become active.

4. Optimization results

The flat plate is optimized using flight conditions in Table 6. Multiple random initial designs within and outside the feasible design space where chosen to test the robustness of the code and optimizer. The optimizer is able to find or get very close to the global optimum, but experiences numerical difficulties close to the global optimum in almost all cases.

To illustrate the issue we give the following example. One initial design is chosen ($AR = 4.38, t = 0.002$), which in this case is within the feasible design space. The optimizer exits indicating numerical difficulties after 6 major iterations with the objective function and design variables equal to -5.025 and $AR^* = 5.98, t^* = 0.001497$, respectively. Note that we are maximizing the range, so the value is in fact positive (5.025). Although the optimum is labeled as the global optimum (with an asterisk) it is not. The aspect ratio, for instance, is not at its maximum of 6, which indicates that the optimum found is not the exact global optimum. The initial and the optimized geometry along with their damping plot is shown in Fig. 8.

By inspecting Fig. 8, we see that the initial geometry (dashed lines) is a feasible design as it does not have any modes with positive damping values for the velocity range of interest 2-15 m/s. As stated in Table 7 the KS aggregate for each mode must be less than zero. For the optimized geometry (solid lines) we see that mode 3, once aggregated, is the one that makes the KS_3 constraint active. Other constraints are inactive. This had already been revealed in our previous design space study. The point on mode 3 that causes the constraint to be active is at the end of the flight envelope, 15 m/s, where the value is zero, hence causing the constraint to be active. Note that the constraints are not shown explicitly in Fig. 8 and neither are the KS_i aggregate values. All constraints would form a horizontal line at 0 rad/s spanning the entire velocity range (2-15 m/s). There are no constraints before or after 2 m/s and 15 m/s, respectively.

Numerical difficulties close to the optimum could indicate incorrect or noisy gradients, or issues in the design space and problem formulation, such as discontinuous constraints. Further investigation of the design space around

the optimum is needed; this is addressed in the next section.

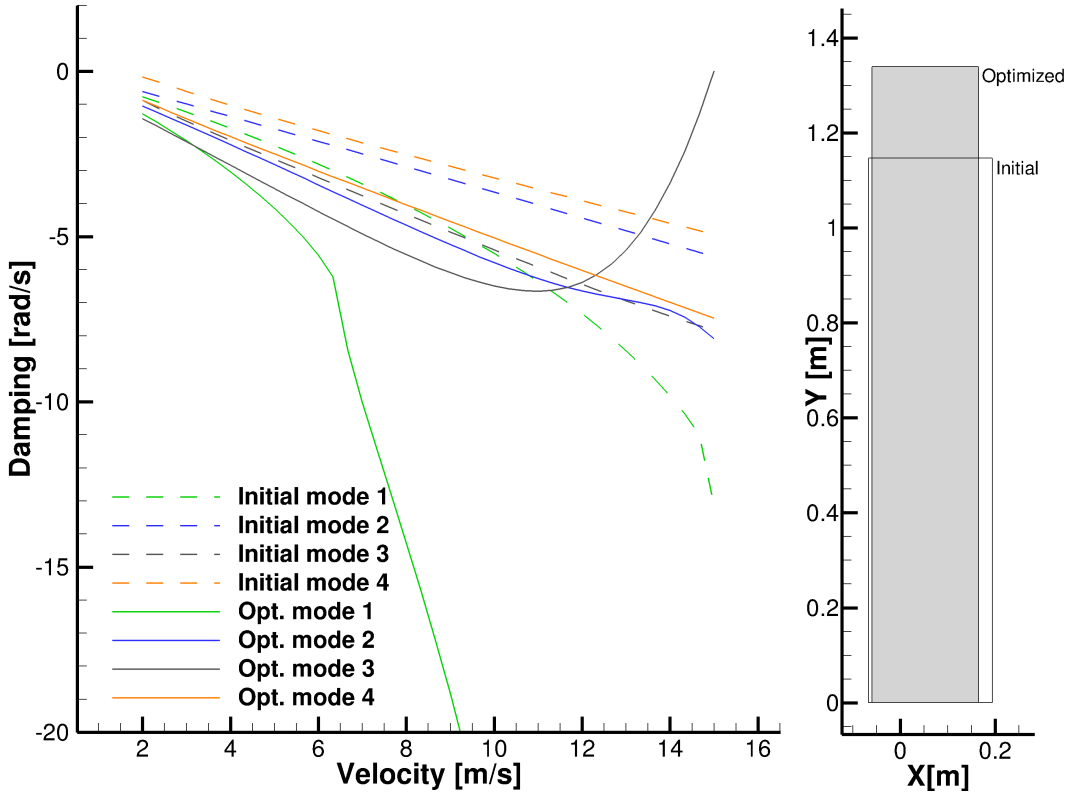

Figure 8. Damping and planform view of the initial ($AR = 4.38, t = 0.002$) and the optimized ($AR^* = 5.98, t^* = 0.001497$) design. As expected mode 3 is causing the constraint KS_3 to be active. It can be seen that the maximum for mode 3 is at 15 m/s and it is very close to zero. Other constraints are inactive. Constraints are not shown explicitly in the figure. Initial geometry shown with outlines and the optimized geometry shown in gray solid color.

5. Further analysis of flutter constraints

To analyze the numerical difficulties experienced in the optimization we turn our attention to the smoothness of the constraints. Figure 9 shows each of the constraints plotted over the entire design space. Each row of plots represents the constraints where the first row of figures correspond to the constraint on mode 1 and the last row corresponds to the constraint on mode 4. The left column shows each constraint for the full design space while the right column shows a zoomed in region $5.8 \leq AR \leq 6.2$ and $0.00148 \leq t \leq 0.00152$ m where the optimum is found. The zoomed region is sampled with 16 extra points in each variable. The zoom region is highlighted with a red square on the full design space in the left column.

Investigating the left column in Fig. 9 we find that the KS aggregated values for mode 1 (KS_1) has discontinuous regions close to the optimum in the lower right corner of the design space. Looking at the zoomed region where the optimum is found we see that the constraint is not smooth which certainly contributes to the numerical difficulties that the optimizer is experiences. By removing mode 1 as a constraint from the optimization problem will likely improve the behavior of the optimization, but this would only be a temporary fix. By investigating the remaining mode constraints 2, 3, and 4 we see that there is a discontinuity appearing between aspect ratios of 2–3 for all thicknesses. Constraints 2, 3, and 4 are however smooth close to the optimum.

This is further illustrated in Fig. 10. The thickness is fixed as $t = 0.001619$ m and the aspect ratio range $1 \leq AR \leq 6$ is sampled with 256 points. The constraints, KS_i, are shown in the top plot in two dimensions. The damping and frequency plots are shown at three aspect ratios of interest labeled from A to C. The numerical value of the aspect ratio is not important here as it is only chosen to explain the discontinuity issue. Aspect ratios chosen are indicated by gray vertical lines in the top plot.

By inspecting the top figure we immediately notice the discontinuity in constraint 4 and constraint 1 as well as the mode hopping or mode swapping where constraint 2 and constraint 3 switch places. By comparing the damping plots at aspect ratios A and B we see that mode 4 is picking up another flutter root that is close to the root that it is

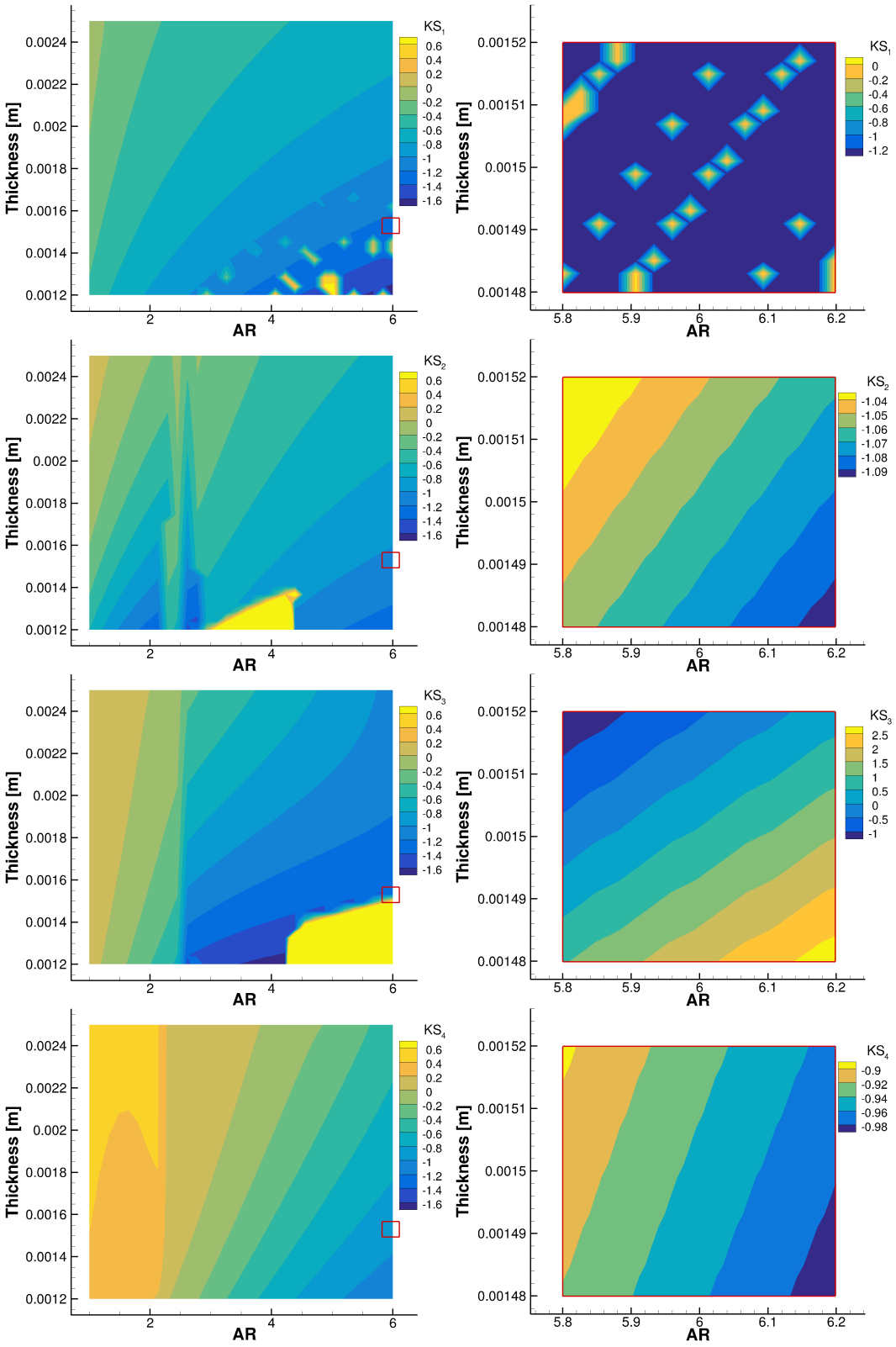

Figure 9. KS values for mode 1 to 4 shown in the left column for the entire design space. The right column shows the region $5.8 \leq AR \leq 6.2$ and $0.00148 \leq t \leq 0.00152$ m sampled using 16×16 stencil. The zoom region is highlighted with a red square on the full design space in the left column.

starting at. Comparing the frequency plots for aspect ratios A and B, we notice that as the frequency of modes 2 and 3 are close to each other for aspect ratio A. The eigenvalue solution method (determinant iteration) can not distinguish between the modes in B, and the same root is picked up for both. When increasing the aspect ratio further past B, we see that modes 2 and mode 3 switch, and are therefore the constraints are not representing the correct mode. Inspecting the frequency plot at aspect ratio C, we notice that the frequency for mode 1 becomes zero. When this happens, two real roots emerge as the complex root (and its conjugate) have zero imaginary part. This should be indicated as a bifurcation on the damping plot. In this case we are only tracking one of the real roots as the current algorithm is only capable of tracking one for each mode. This explains the discontinuity in mode 1 at aspect ratio C. The bifurcated real root starts to appear between $B \leq AR \leq C$ but, depending on which real root the algorithm "locks" onto, changes between different AR causing these discontinuities to show up. This can be prevented by using an algorithm that can track both real roots. To avoid mode swapping and track the real roots properly, a more sophisticated algorithm than presented here is needed.

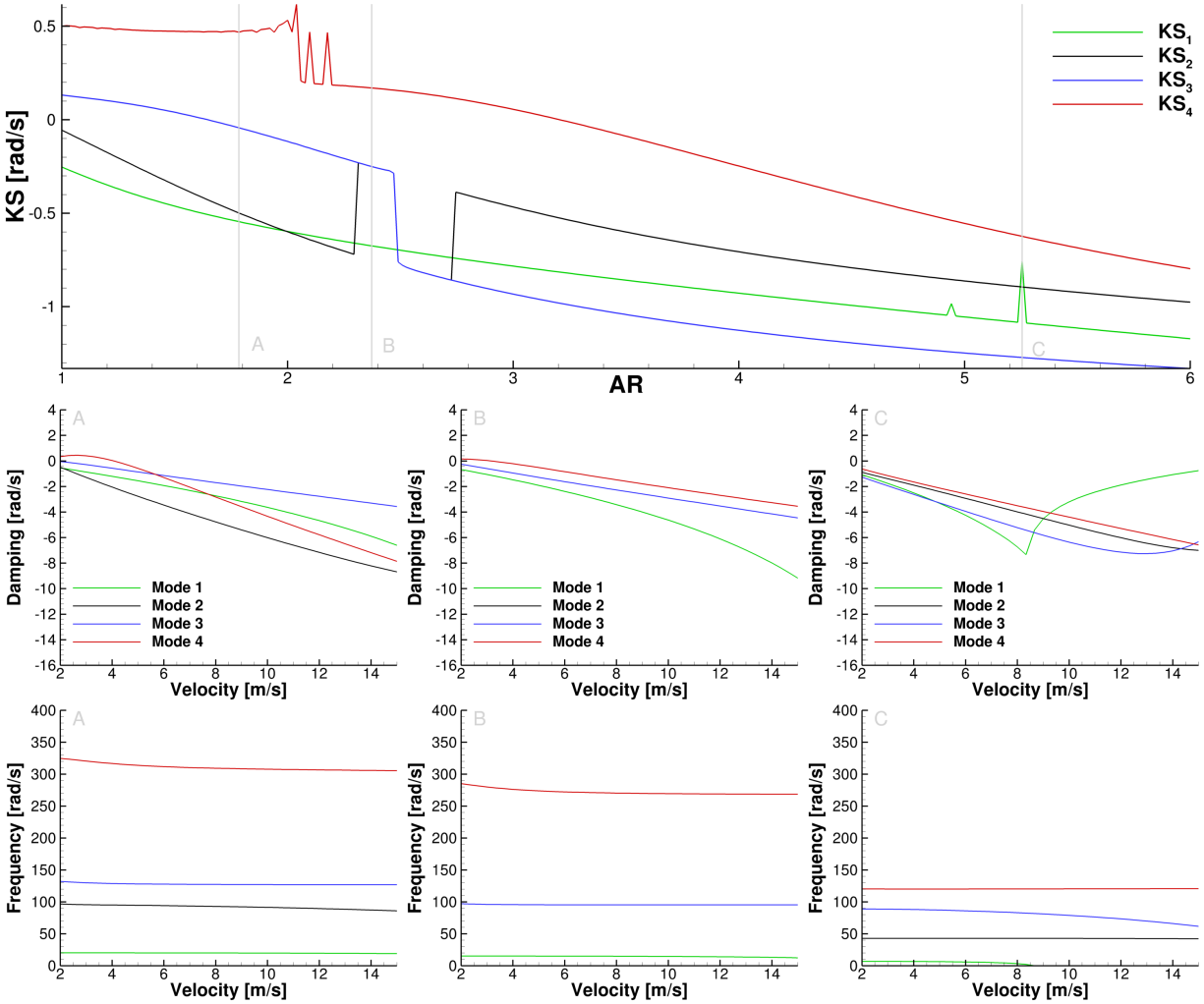

Figure 10. Top plot shows the KS values where thickness is fixed as $t = 0.001619$ m and the aspect ratio range, $1 \leq AR \leq 6$, is sampled with 256 points. Damping and frequency plots are shown for three different aspect ratios labeled A to C and show the modes migrate history with varying air speed. Refer to text for full details.

V. uCRM flutter analysis

1. Model description

The geometry under consideration is the undeflected Common Research Model (uCRM) developed by Kenway et al. [37] for aerostructural optimization. This geometry is based off the Common Research Model which has been studied extensively in multiple Drag Prediction Workshops (DPW) as well as by the Aerodynamic Design Optimization

Discussion Group test cases [38, 39, 40]. The purpose of the uCRM is to provide the undeformed jig geometry and structural layout for aerostructural optimization, which reflects the overall size and shape of a typical transport aircraft such as the Boeing 777-200ER.

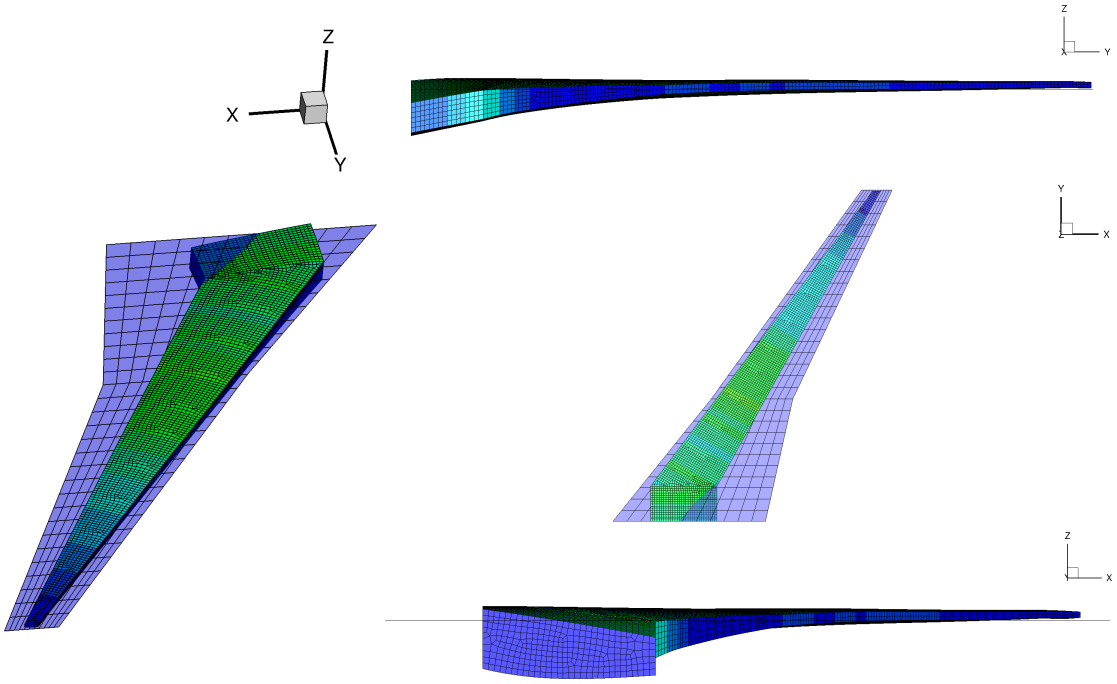

Figure 11. The uCRM wingbox shown in green along with the aerodynamic mesh shown in purple. The FFD volume is not shown here.

The aerodynamic surface of the wing is discretized using 11 elements in both the streamwise and spanwise directions for the inboard section, and 11 and 18 elements in streamwise and spanwise directions for the outboard section of the wing. The wingbox is composed of 10584 MITC4 shell finite elements. The aerodynamic mesh and wingbox are shown in Fig. 11 where the aerodynamic mesh is shown in purple, and the wingbox in green. The FFD used (not shown) encapsulates both meshes. The material properties and discretization parameters for the uCRM are summarized in Table 8. Skin, spar and rib thickness, and stiffener height, thickness and pitch are not included here.

For the analysis of the uCRM, the Mach number is fixed at $M = 0.85$. The air speed and the air density are varied in 79 increments as a pair ranging from 253 - 305 m/s and 0.04 - 1.99 kg/m^3 receptively. The operating conditions are summarized in Table 9. We use the Lanczos method to calculate the 20 first natural modes and mode shapes using a subspace size of 50.

2. Problem statement

To gain insight into the uCRM flutter characteristics we formulate the problem similarly as the flat plate problem where we use two effective design variables, the aspect ratio and thickness scaling variable. Skin, spar and rib thickness and stiffener height, thickness and pitch are modeled as individual variables but are then scaled using a single global scaling factor t_{scaling} that is allowed to vary from $0.8 \leq t_{\text{scaling}} \leq 1.75$. When $t_{\text{scaling}} = 1.0$ all structural variables are at their initial value. When $t_{\text{scaling}} < 1.0$, the value of the variables decrease. This approach is used here in order to visualize the design space in a similar manner as for the flat plate example.

The objective is to maximize the range using Eq. (2.1), but here the range is in nautical miles. As before the lift coefficient is fixed at $C_L = 0.5$. The drag coefficient is modeled using the lift induced drag and the zero lift drag coefficient,

$$C_D = \frac{C_L^2}{\pi e AR} + C_{D_0} \tag{2.1}$$

For simplicity we set $e = 1$ and $C_{D_0} = 0.0162$. The flight speed in the range equation is fixed to $V = 250$ m/s and the thrust-specific fuel consumption is fixed as $c_T = 0.53$ lb/(lbf $\cdot$ h). The initial and final cruise weights are defined

Table 8. uCRM mechanical properties, dimensions and discretization of the structure and the aerodynamic surface.

Variable	Symbol	Value
Density	ρ_s	2780 kg/m^3
Modulus of Elasticity	E	70 GPa
Poisson ratio	ν	0.3
Yield stress	σ_y	420 MPa
Span	b	58.6 m
Aspect ratio	AR	9.0
Reference Area	S	383.7 m ^{2}
Sweep (leading edge)	Λ	37.4 deg
Number of finite elements	n_{FE}	10584
Number of DLM elements, inboard streamwise	n_x^i	11
Number of DLM elements, inboard spanwise	n_y^i	11
Number of DLM elements, outboard streamwise	n_x^o	11
Number of DLM elements, outboard spanwise	n_y^o	18

Table 9. The uCRM operating conditions under investigation. The airspeed and density are incremented together in 79 steps.

Variable	Symbol	Value
Cruise Mach	M_∞	0.85
Cruise lift coefficient	C_L	0.5
Cruise air density	ρ_∞	0.04–1.99 kg/m^3
Cruise air speed range	U_∞	253–305 m/s
Number of speed/density increments	n_U	79
Zero lift drag coefficient	C_{D_0}	0.0169
Flight speed (range eqn.)	V	250 m/s
Thrust specific fuel consumption	c_T	0.53 lb/(lbf $\cdot$ h)
Fixed weight	W_{fixed}	197000 kg
Fuel weight	W_{fuel}	90000 kg

as

$$W_{\text{final}} = W_{\text{fixed}} + 2.5W_{\text{wing}}$$
$$W_{\text{init}} = W_{\text{final}} + W_{\text{fuel}} \tag{2.2}$$

All parameters used here are summarized in Table 9. Flutter constraints are added to the first 10 modes , ($KS_i \leq 0$ for $i = 1, \ldots, 10$), and area is kept constant. As for the plate problem, the flutter constraints is enforced over the entire flight envelope (density and velocity pairs) and thus there is no flutter point specified explicitly. The optimization problem is summarized in Table 10. The aspect ratio is allowed to vary from $8.18 \leq AR \leq 16.43$, and as mentioned earlier the thickness scaling factor can vary from $0.8 \leq t_{\text{scaling}} \leq 1.75$.

Table 10. Overview of the simplified optimization problem for the uCRM wing. All structural variables are scaled using one scaling parameter.

	Function/variable	Description	Quantity
maximize	Breguet range equation		
with respect to	x_{planform}	Planform variables	3
	x_{struct}	Structural sizing variables	582
		Total design variables	**585**
subject to	$A - A_{\text{init}} = 0.0$	Fixed area	1
	$\text{KS}_i \leq 0$	KS aggregate of damping values for modes $i = 1, \ldots, 10$	10
		Total constraints	**11**

3. Design space analysis

As for the flat plate problem the design space is analyzed. To generate the contour plot the design space is sampled using 32 samples in each variable, the aspect ratio and the thickness scaling, giving a total of 1024 samples. Figure 12 shows the KS aggregated values for the first 9 modes, plotted as contour plots. None of the modes, excluding mode 9 which shows discontinuities, violate the constraint (i.e. $KS_i > 0$). Mode 2 and 5 have however regions where they are very close to zero or practically zero. Mode 9, shows similar discontinuities as the flat plate geometry where $KS_9 >> 0$. Mode 10, which is not shown here, shows similar behavior as the others.

Figure 13 highlights this further using a slice at fixed thickness scaling of $t_{\text{scaling}} = 1.1$ over all aspect ratios. Inspecting the figure we see that mode 1 experiences a discontinuity at lower aspect ratios. Similar behavior was seen with the flat plate case where the flutter root finding method is "locking" onto a different eigenvalue. Mode 9 show similar behavior and experiences a discontinuity around $AR = 12.7$. Here this mode has likely bifurcated and split into two real roots and the root finding method is "locking" onto which ever root it finds first. As noted before mode 5 is also very close to zero and is practically zero at the higher aspect ratios. This design space appears to be very uninteresting as none of the constraints become active although mode 2 and 5 come close. Optimization of this simplified problem does thus offer limited insight into the flutter characteristics of the uCRM. No optimization is thus done at this point.

A possible explanation why none of the modes are active could be due to the scaling of the entire structure using only one thickness scaling variables. By lowering the bounds on the thickness scaling parameter we might able to produce points where the structure flutters. Another explanation why the structure does not flutter is the omission of the masses due to the engine as well as leading edge and trailing edge devices such as slats, flaps and ailerons. One possible way of modeling these masses is to lump them at as a point mass at discrete locations and connect them to the structure using rigid body elements (RBE3) elements such that they do not add stiffness to the structure. Their effect is thus purely inertial. Figure 14 is a sketch of such a solution where the rigid links are shown as lines extending out from the structure. These lumped masses have been shown to be very important [9] as they reduce the torsional frequency substantially resulting in a lower flutter speed of the structure.

VI. Conclusions

Sensitivities of the implemented flutter constraints in this work are both accurate and computed efficiently. This allows us to perform an optimization of problems with tens or hundreds of design variables. However, as illustrated with a simple flat plate problem and the uCRM geometry, the determinant iteration root finding method, is not adequate

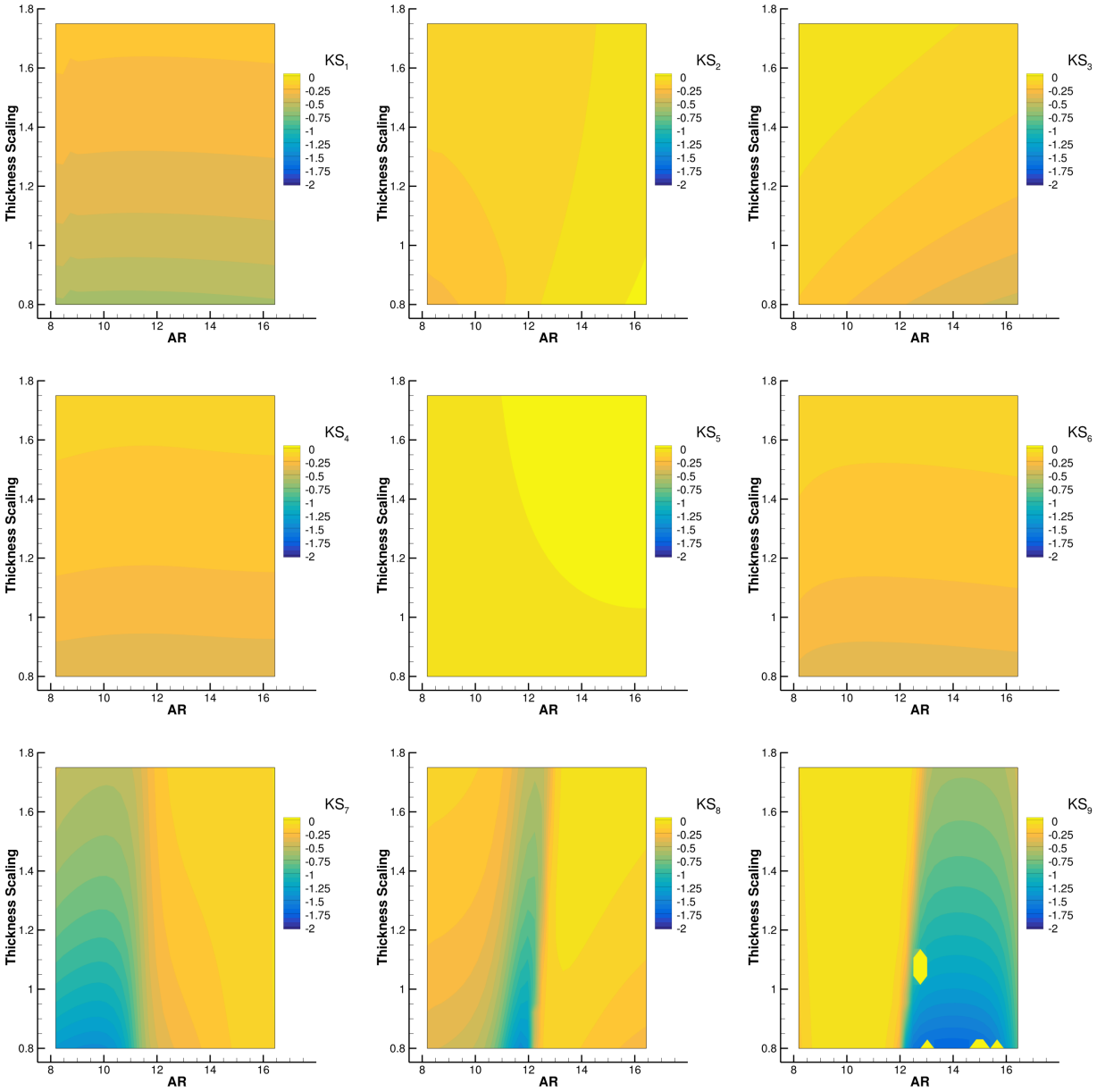

Figure 12. KS values for mode 1 to 9 shown for the entire design space. Design space was sampled with 32×32 stencil

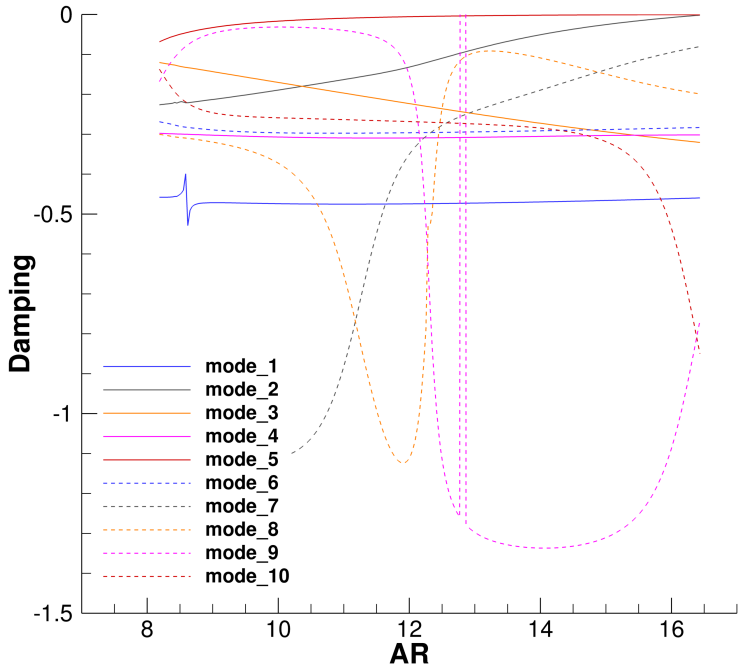

Figure 13. KS values for mode 1 to 10 shown as a slice at thickness scaling $t_{\text{scaling}} = 1.1$ for all aspect ratios. Notice the discontinuity in mode 9.

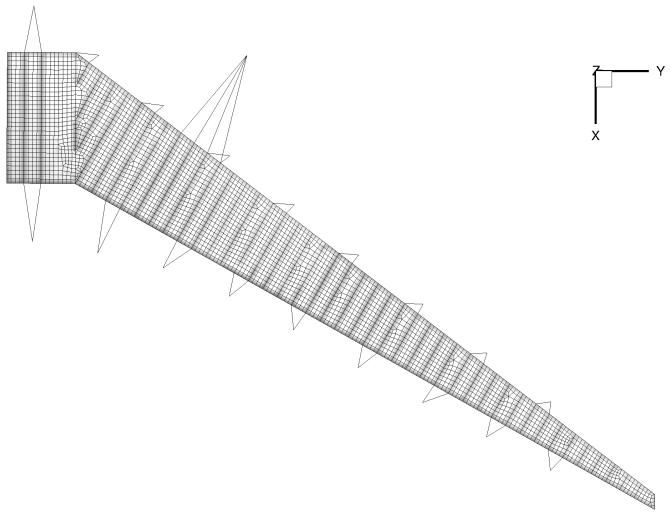

Figure 14. uCRM wingbox, where the engine mass as well as leading and trailing edge devices are modeled as lumped point masses connected to the structure using rigid body elements (RBE3); adapted from [9].

or robust, even for the simplest problems. Issues encountered in this work that relate to the method can be described as follows:

1. If an eigenvalue changes rapidly with increasing velocity the method might converge to an incorrect value due to a large step size in velocity. This issue can be mitigated by substantially increasing the number of velocities analyzed.

2. If two eigenvalues have similar or same imaginary parts this method can "lock" onto an incorrect eigenvalue. This issue was frequently observed in this work. Substantially increasing the number of velocity points analyzed may sometimes alleviate this issue, although it might become a computational burden if large number of modes and velocities are needed for the flutter analysis.

3. This method offers no direct way of tracking two real roots that emerge from a bifurcated root where the imaginary part vanishes.

To determine the flutter speed accurately and efficiently, and to implement it as a constraint in an optimization, these issues need to be addressed. Using more sophisticated root finding methods or reformulating the problem are options that we are currently investigating.

Necessary capabilities need to be developed to model the rigid body elements (RBE3), or something similar, to support lumped discrete masses that do not add stiffness to the structure. These point masses are lumped from the engine, leading and trailing edge devices and have been shown to be very important as they decrease the predicted flutter speed [9].

Acknowledgements

Funding for this research was provided by NASA under grant number NNX11AI19A program mangers Karen Taminger and Christine Jutte. The authors would also like to thank Bret Stanford for his support.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575 [41].

References

[1] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "A Scalable Parallel Approach for High-Fidelity Aerostructural Analysis and Optimization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Honolulu, HI, April 2012, AIAA 2012-1922.

[2] Kenway, G. K. W. and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, January 2014, pp. 144–160. doi:10.2514/1.C032150.

[3] Kenway, G. W. K. and Martins, J. R. R. A., "High-fidelity aerostructural optimization considering buffet onset," *Proceedings of the 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Dallas, TX, June 2015, AIAA 2015-2790.

[4] Liu, F., Cai, J., Zhu, Y., Tsai, H., and F. Wong, A., "Calculation of wing flutter by a coupled fluid-structure method," *Journal of Aircraft*, Vol. 38, No. 2, 2001, pp. 334–342.

[5] Yurkovich, R., "Status of unsteady aerodynamic prediction for flutter of high-performance aircraft," *Journal of Aircraft*, Vol. 40, No. 5, 2003, pp. 832–842.

[6] Schuster, D. M., Liu, D. D., and Huttsell, L. J., "Computational aeroelasticity: success, progress, challenge," *Journal of Aircraft*, Vol. 40, No. 5, 2003, pp. 843–856.

[7] Bennett, R. M. and Edwards, J. W., "An overview of recent developments in computational aeroelasticity," *AIAA paper*, , No. 98-2421, 1998.

[8] Gibbons, M. D., "Aeroelastic calculations using CFD for a typical business jet model," 1996.

[9] Stanford, B. K. and Dunning, P. D., "Optimal Topology of Aircraft Rib and Spar Structures Under Aeroelastic Loads," *Journal of Aircraft*, Vol. 52, No. 4, Sept. 2014, pp. 1298–1311. doi:10.2514/1.C032913.

[10] Stanford, B., Wieseman, C. D., and Jutte, C., "Aeroelastic Tailoring of Transport Wings Including Transonic Flutter Constraints," *AIAA SciTech*, American Institute of Aeronautics and Astronautics, Jan. 2015, pp. –. doi:10.2514/6.2015-1127.

[11] Haftka, R. T. and Gürdal, Z., *Elements of structural optimization*, Vol. 11, Springer Science & Business Media, 2012.

[12] Kennedy, G. J. and Martins, J. R. R. A., "Parallel Solution Methods for Aerostructural Analysis and Design Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, Sept. 2010, AIAA 2010–9308.

[13] Kennedy, G. J. and Martins, J. R. R. A., "A parallel aerostructural optimization framework for aircraft design studies," *Structural and Multidisciplinary Optimization*, Vol. 50, No. 6, December 2014, pp. 1079–1101. doi:10.1007/s00158-014-1108-9.

[14] Albano, E. and Rodden, W. P., "A doublet-lattice method for calculating lift distributions on oscillating surfaces in subsonic flows." *AIAA Journal*, Vol. 7, No. 2, Feb. 1969, pp. 279–285. doi:10.2514/3.5086.

[15] Blair, M., "A compilation of the mathematics leading to the doublet lattice method," Tech. rep., DTIC Document, 1992.

[16] MSC Software Corp, *MSC Nastran 2012 Release Guide*.

[17] Rodden, W. P., Taylor, P. F., and McIntosh, S. C., "Further Refinement of the Subsonic Doublet-Lattice Method," *Journal of Aircraft*, Vol. 35, No. 5, Sept. 1998, pp. 720–727. doi:10.2514/2.2382.

[18] Trefethen, L. N. and Bau III, D., *Numerical linear algebra*, Vol. 50, SIAM, 1997.

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

[20] Hassig, H. J., "An approximate true damping solution of the flutter equation by determinant iteration." *Journal of Aircraft*, Vol. 8, No. 11, 1971, pp. 885–889.

[21] Kreisselmeier, G. and Steinhauser, R., "Systematic Control Design by Optimizing a Vector Performance Index," *International Federation of Active Controls Syposium on Computer-Aided Design of Control Systems, Zurich, Switzerland*, 1979.

[22] Poon, N. M. K. and Martins, J. R. R. A., "An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis," *Structural and Multidisciplinary Optimization*, Vol. 34, No. 1, July 2007, pp. 61–73. doi:10.1007/s00158-006-0061-7.

[23] Wrenn, G. A., "An Indirect Method for Numerical Optimization Using the Kreisselmeier–Steinhauser Function," Tech. Rep. CR-4220, NASA, 1989.

[24] Rodden, W. P., Bellinger, E. D., Harder, R. L., and Center., L. R., *Aeroelastic addition to NASTRAN*, National Aeronautics and Space Administration, Scientific and Technical Information Branch, 1979.

[25] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, Sept. 2010. doi:10.2514/6.2010-9231, AIAA 2010-9231.

[26] Sederberg, T. W. and Parry, S. R., "Free-form Deformation of Solid Geometric Models," *SIGGRAPH Comput. Graph.*, Vol. 20, No. 4, Aug. 1986, pp. 151–160. doi:10.1145/15886.15903.

[27] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[28] Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, January 2012, pp. 101–118. doi:10.1007/s00158-011-0666-3.

[29] Griewank, A. and Walther, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., 2008.

[30] Mader, C. A. and Martins, J. R. R. A., "Derivatives for Time-Spectral Computational Fluid Dynamics Using an Automatic Differentiation Adjoint," *AIAA Journal*, Vol. 50, No. 12, December 2012, pp. 2809–2819. doi:10.2514/1.J051658.

[31] Hascoët, L. and Pascual, V., "TAPENADE 2.1 User's Guide," Technical report 300, INRIA, 2004.

[32] Pascual, V. and Hascoët, L., "Extension of TAPENADE Towards Fortran 95," *Automatic Differentiation: Applications, Theory, and Tools*, edited by H. M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, Lecture Notes in Computational Science and Engineering, Springer, Berlin, Germany, 2005.

[33] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd ed., 1999.

[34] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, September 2003, pp. 245–262. doi:10.1145/838250.838251.

[35] Hascoet, L. and Pascual, V., "The Tapenade automatic differentiation tool: Principles, model, and specification," *ACM Trans. Math. Softw.*, Vol. 39, No. 3, May 2013, pp. 20:1–20:43. doi:http://dx.doi.org/10.1145/2450153.2450158.

[36] Lambe, A. B. and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, August 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

[37] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Aerostructural Optimization of the Common Research Model Configuration," *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, June 2014. doi:10.2514/6.2014-3274, AIAA 2014-3274.

[38] Vassberg, J. C., DeHaan, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," 2008, AIAA 2008-6919.

[39] Vassberg, J. and Jameson, A., "Influence of Shape Parameterization on Aerodynamic Shape Optimization," Tech. rep., Von Karman Institute, Brussels, Belgium, April 2014.

[40] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "RANS-based Aerodynamic Shape Optimization Investigations of the Common Research Model Wing," *Proceedings of the AIAA Science and Technology Forum and Exposition (SciTech)*, National Harbor, MD, January 2014. doi:10.2514/6.2014-0567, AIAA 2014-0567.

[41] Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., Roskies, R., Scott, J. R., and Wilkins-Diehr, N., "XSEDE: Accelerating Scientific Discovery," *Computing in Science & Engineering*, Vol. 16, No. 5, September 2014, pp. 62–74. doi:10.1109/MCSE.2014.80.