International Forum on Aeroelasticity and Structural Dynamics
IFASD 2024
17-21 June 2024, The Hague, The Netherlands

DIFFERENTIABLE AEROELASTIC FRAMEWORK SUITABLE FOR INDUSTRIAL MODELLING OF NONLINEAR LOADS ON ACCELERATORS

Alvaro Cea[1] **and Rafael Palacios**[1]

[1]Imperial College London
Exhibition Road, London SW7 2AZ United Kingdom
alvaro.cea-esteban15@imperial.ac.uk
r.palacios@imperial.ac.uk

Keywords: Nonlinear aeroelasticity, dynamic loads, high-performance, JAX

Abstract: This paper presents a new simulation environment for time-domain nonlinear aeroelastic analysis built for performance and that is suitable for modern hardware architectures such as GPUs. The numerical library JAX and a novel description of the aircraft dynamics are brought together into a highly vectorised codebase that achieves two orders of magnitude accelerations compare to conventional implementations. This brings full-vehicle simulations to run close to if not in real-time, thus opening new possibilities for aircraft design optimization and aeroelastic analysis.
The computational edge provided by GPUs is shown in a free-flying very flexible structure. It follows an extensive verification by comparison with MSC Nastran full-FE linear and nonlinear structural solutions on a representative aeroplane model. Furthermore, the nonlinear gust response of an industrial configuration with over half a million degrees-of-freedom is computed, and it is faster than its frequency-based, linear equivalent as implemented by commercial packages. Therefore this could be harnessed by aircraft loads engineers to add geometrically nonlinear effects to their existing workflows at no extra computational effort. Finally, automatic differentiation on both static and dynamic responses is validated against finite-differences.

1 INTRODUCTION

The ever-growing need for performance and operating costs reduction, together with the current push for sustainability in aviation, are driving new aircraft designs outside the conventional envelope. A particular feature are very high aspect ratio wings to minimise induced drag, which when combined with advancements in lighter materials to reduce vehicle weight, can significantly increase wing flexibility. In such a scenario, aeroelastic analysis are expected to become critical in the very early phases of the wing design process: while the field was more important in post-design stages to ensure in-flight integrity, it now becomes paramount to capture the cross-couplings between disciplines. As highlighted in [1], formulations that include nonlinear effects should be developed that not only enhance current modelling techniques but that also allow rapid data turnaround for the industry. Real-time, hardware-in-the-loop flight simulators would also benefit of actively controlled, deformable airplane models. This leads to a more nonlinear landscape, where the overall aerodynamic performance needs to be calculated around a flight shape with large deformations [2]; the input for efficient control laws account for the steady state and nonlinear couplings [3]; and the loads ultimately sizing the wings are atmospheric disturbances computed in the time-domain [4]. This is also the case for more radical

configurations that may or may not exhibit high flexibility but whose aeroelastic behaviour is more uncertain. A more holistic approach to the design also increases the complexity of the processes exponentially, and the trade-offs and cost-benefit analysis may not be possible until robust computational tools are in-place to simulate the different assumptions. High fidelity structural [5] and aero-structural optimizations [6] which also include nonlinear flutter constraints due to geometric nonlinearities are good examples of the push for further integration in the design process.

Certification of new air vehicles is another important aspect that requires 100,000s of load cases simulations [7], as it considers manoeuvres and gust loads at different velocities and altitudes, and for a range of mass cases and configurations. For roll manoeuvres, for instance, it has been shown the importance of fully coupled nonlinear aeroelastic-flight dynamics solutions [8]. This poses another challenge for new methods that aim to include new physics since they normally incur in prohibitively expensive computational times. To tackle this, projection based reduced order models have been employed even for high fidelity analysis [9], and are also blended with machine learning models [10]. Lastly, the mathematical representation of the airframe, embodied in the complex Finite-Element Models (FEMs) built by organizations, encompasses a level of knowledge that is to be preserved when including the new physics mentioned above. This has led to recent efforts geared towards building robust methods that incorporate nonlinear effects to those FEMs, either via stick models constructed for aeroelastic analysis [11], or using a modal-based approximation [12].

Those previous considerations set the goals for the present work: 1) to be able to perform geometrically nonlinear aeroelastic analysis, 2) to work with generic FEMs in a non-intrusive manner, and 3) to achieve a computational efficiency that is equivalent to present linear methods (if not faster). Grounded on previous developments where the first two points were demonstrated [13], [14], [15] we tackle the third point herein with a new implementation that achieves remarkable computational performance. The numerical library JAX [16] was leveraged to produce highly vectorised, automatically differentiated routines that are managed by a modular, object-oriented approach in Python. The power of JAX for scientific computation has been proved recently in fluid dynamics [17] and solid mechanics [18] applications. It provides a unified computational framework that can be seamlessly deployed on CPUs, GPUs and TPUs without needing to resort to Domain Specific Languages (DSL) [19]. In addition it also features powerful parallelisation capabilities across devices, a hot research topic given the new era we enter where software does not outlive hardware but potentially the other way around.

Our proposed method has two main inputs for the analysis: a linear (arbitrarily complex) FE model, and frequency-dependent aerodynamic influence coefficient matrices that provide the mapping between FE states and the corresponding aerodynamic forces (either in modal or in physical coordinates). The latter are obtained herein from the Doublet Lattice Method (DLM) and a rational function approximation (RFA) [20] to transform to the time domain. We have also presented a more efficient data-driven approach that circumvents the lag selection process of the RFA in [21] and which would also be suitable for more accurate Computational Fluids Aerodynamics (CFD). Using the 3D FE model, a skeleton-like substructure along the main load paths is derived, on which modal shapes and nonlinear couplings are evaluated in intrinsic variables (velocities and strains) [22]. They conform a basis of a Galerkin-projection of the geometrically-nonlinear 1D description [23] after which the projected equations are solved in time-domain. Advantages of the approach are its direct and accurate map between the 3D and 1D domains, as it only requires of a modal condensation that is already available in many industrial aeroelastic models to link the structural model to the aerodynamic loading. This is unlike stick models which need of various post-processing steps to build the equivalent stiffness and

mass models. Furthermore, we show how the full 3D solution using the nonlinear 1D solution is computed to a good accuracy by reconstructing the cross-sectional elements and applying a Radial Basis Function (RBF) interpolation to the remaining nodes in the domain. A well established formulation effectively applied to large-scale aeroelastic models and now combined with a highly vectorised implementation in JAX results in an extremely efficient nonlinear aeroelastic solver. The overall procedure has been implemented in what we have named as *Nonlinear Modal Reduced Order Model* (NMROM).

The structure of the rest of the paper is as follows. Sec. 2 presents a summary of the mathematical description that conforms the backbone behind the computational implementation of `FENIAX` (Finite-Element models for Intrinsic Aeroelastics in JAX), the high performance software for aeroelasticity we have built. Sec. 3 shows the verification cases that cover a very flexible, free-flying structure, the static and dynamic structural response of a simplified aircraft model, and the aeroelastic response to gusts of a full aircraft configuration. The performance edge of the implementation is highlighted in all of the examples. Lastly, sec. 4 summarises the the achievements and further developments planned for future work.

2 THEORY AND IMPLEMENTATION

In this section we briefly describe the backbone theory of the proposed methods for nonlinear aeroelastic modelling as continuation of the work in [14, 15]. A summary of implementation details and computational algorithms are presented next.

2.1 Nonlinear aeroelastic system

We start with a global FE model of the airframe as illustrated in Fig. 1.

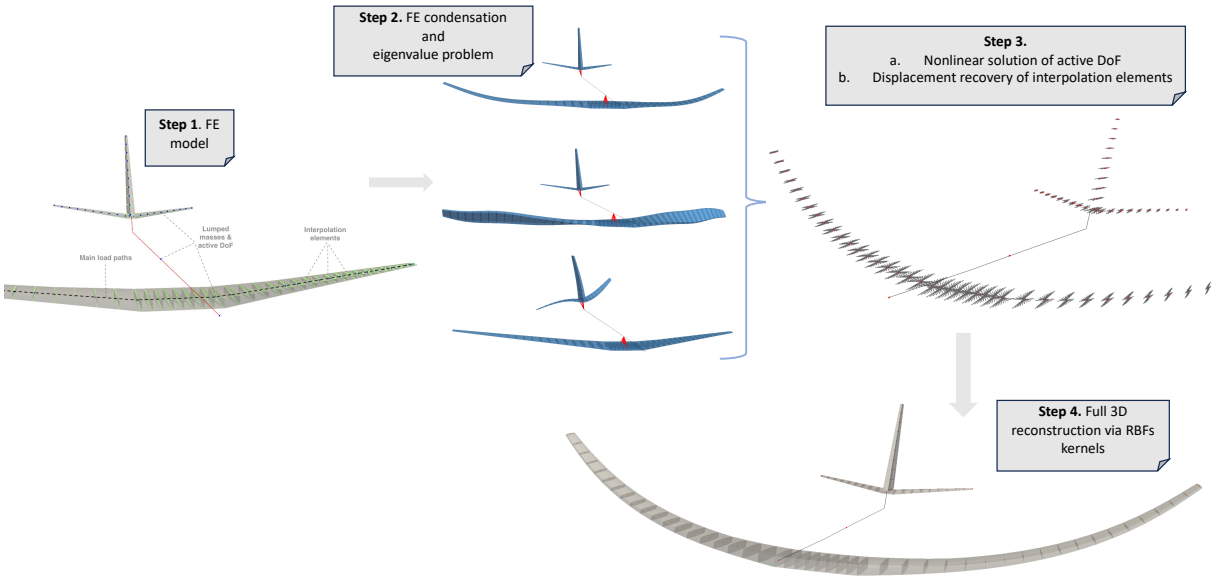


Figure 1: Workflow of the solution process

It is common practice for large-scale aeroelastic models to feature lumped masses along a load path axis that are attached to their corresponding cross-sectional nodes via interpolation elements. With those characteristics a reduced model can be obtained from a static condensation, or Guyan reduction [24], that captures well the stiffness and inertia properties in the condensed matrices, $\boldsymbol{K}_a$ and $\boldsymbol{M}_a$ (Step 1 in Fig. 1). In the case where the mass is given by a generic mass model the method is also valid and dynamic condensation can be employed for additional ac-

curacy, as demonstrated in [25]. The eigenvalue solution of the FEM yields the modal shapes, $\boldsymbol{\Phi}_0$, and frequencies $\boldsymbol{\omega}$ (Step 2, however, $\boldsymbol{\Phi}_0$ is defined on the master nodes and the figure shows the full reconstructed modal shapes). The dynamics of this reduced model are described by a system on nonlinear equations [23] written in material velocities, $\boldsymbol{x}_1$, and stresses, $\boldsymbol{x}_2$, as state variables. A modal expansion of those is a key step in seamlessly mapping the global FEM into the nonlinear description. The intrinsic modes are introduced and the projection of the state variables is such $\boldsymbol{x}_1 = \boldsymbol{\Phi}_1 \boldsymbol{q}_1$ and $\boldsymbol{x}_2 = \boldsymbol{\Phi}_2 \boldsymbol{q}_2$. A resulting set of four intrinsic modal shapes are directly linked to the displacement modal shapes coming from the global FEM:

1. Velocity modes, $\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_0$, which follow after the linear relation with displacements: $\boldsymbol{x}_1 = \dot{\boldsymbol{x}}_0$, $\boldsymbol{\Phi}_1 \boldsymbol{q}_1 = \boldsymbol{\Phi}_0 \dot{\boldsymbol{q}}_0$.
2. Momentum modes, $\boldsymbol{\Psi}_1 = \boldsymbol{M}_a \boldsymbol{\Phi}_0$. Note from this definition that, for arbitrary distributed mass models, the dynamic condensation technique will produce a fully-populated mass matrix, and the various couplings will be captured after the matrix multiplication.
3. Force/moment modes, $\boldsymbol{\Phi}_2 = \mathcal{S}(\boldsymbol{K}_a \boldsymbol{\Phi}_0)$, represent the internal stress resultants in the structure as the sum, $\mathcal{S}$, along the main load-paths of equilibrium forces and moments produced by the modal deformations. As a consequence, results are presented in the mid-point between nodes because more information cannot be extracted in terms of linear stresses from one node to the other.
4. Strain modes, $\boldsymbol{\Psi}_2 = -\boldsymbol{\Phi}_{0d} + \boldsymbol{E}^\top \boldsymbol{\Phi}_{0m}$, with $\boldsymbol{\Phi}_{0d}$ the approximate derivative along s: $\boldsymbol{\Phi}^i_{0d} = \frac{\boldsymbol{\Phi}^{i+1}_0 - \boldsymbol{\Phi}^i_0}{\Delta s_i}$; and $\boldsymbol{\Phi}_{0m} = \frac{\boldsymbol{\Phi}^{i+1}_0 + \boldsymbol{\Phi}^i_0}{2}$, the displacement modal shape in between nodes. $\boldsymbol{E}^\top$ is a constant matrix as defined in [13].

Details of their computational implementation in JAX can be found in Algorithm 1 below. Using the computed modal shapes, a dynamic system is obtained after a Galerkin projection of the equations of motion [26, Ch. 8]:

$$
\begin{aligned}
\dot{\boldsymbol{q}}_1 &= \boldsymbol{\omega} \odot \boldsymbol{q}_2 - \boldsymbol{\Gamma}_1 : (\boldsymbol{q}_1 \otimes \boldsymbol{q}_1) - \boldsymbol{\Gamma}_2 : (\boldsymbol{q}_2 \otimes \boldsymbol{q}_2) + \boldsymbol{\eta} \\
\dot{\boldsymbol{q}}_2 &= -\boldsymbol{\omega} \odot \boldsymbol{q}_1 + \boldsymbol{\Gamma}_2^\top : (\boldsymbol{q}_2 \otimes \boldsymbol{q}_1)
\end{aligned}
\tag{1}
$$

where $\odot$ is the Hadamard product (element-wise multiplication), $\otimes$ is the tensor product operation and $:$ is the double dot product[1]. The equations have been written herein in compact tensorial notation, which is in fact the way they have been implemented and vectorised. This description is geometrically-exact, with nonlinearities encapsulated in the modal couplings of the third-order tensors $\boldsymbol{\Gamma}_1$ and $\boldsymbol{\Gamma}_2$ (the former introduces the gyroscopic terms in the dynamics and the latter introduces the strain-force nonlinear relation). $\boldsymbol{\eta}$ is the modal projection of the external forcing terms. They are computed as integrals along the load-paths as an inner product: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \int_\Gamma \boldsymbol{u}^\top \boldsymbol{v} ds$, for any $\boldsymbol{u} \in \mathbb{R}^6$ and $\boldsymbol{v} \in \mathbb{R}^6$, as

[1]The double dot product represents a contraction of the last two indexes of the first tensor with the first two indexes of the second one; it however needs further specification as two alternative definitions can be adopted and here we opt for the following: $\boldsymbol{a} : \boldsymbol{b} = \sum_i \sum_j a_{..ij} b_{ij..}$. This has implications on the definition of the transpose of $\boldsymbol{\Gamma}_2$ in the second equation since for high order tensors multiple transpose operators can be defined. Consistency is achieved by ensuring the dot product operation satisfies the following: $\boldsymbol{x} \cdot (\boldsymbol{\Gamma} : (\boldsymbol{y} \otimes \boldsymbol{z})) = \boldsymbol{y} \cdot \left(\boldsymbol{\Gamma}^\top : (\boldsymbol{z} \otimes \boldsymbol{x})\right)$, which leads to the transpose of the third order tensor, $\boldsymbol{\Gamma} = \Gamma^{ijk}$, as $\boldsymbol{\Gamma}^\top = \Gamma^{jki}$.

$$
\begin{aligned}
\Gamma_1^{ijk} &= \langle \boldsymbol{\Phi}_{1i}, \mathcal{L}_1(\boldsymbol{\Phi}_{1j})\boldsymbol{\Psi}_{1k} \rangle, \\
\Gamma_2^{ijk} &= \langle \boldsymbol{\Phi}_{1i}, \mathcal{L}_2(\boldsymbol{\Phi}_{2j})\boldsymbol{\Psi}_{2k} \rangle, \\
\eta_i &= \langle \boldsymbol{\Phi}_{1i}, \boldsymbol{f}_1 \rangle
\end{aligned}
\tag{2}
$$

with $\mathcal{L}_1$ and $\mathcal{L}_2$ linear operators. The solution of Eqs. 4 correspond to Step 3 in Fig. 1, and can be extended to form the full aeroelastic system with gravity forces, $\boldsymbol{\eta}_g$, aerodynamic forces, $\boldsymbol{\eta}_a$, and gust disturbances, $\boldsymbol{v}_g$. Control states can also be included [25], but they are not necessary for this work. For a set of reduced frequencies and a given Mach number, the DLM (or a higher fidelity aerodynamic method) yields the Generalised Aerodynamic Forces (GAFs). The current implementation uses Roger's rational function approximation to those GAFs [20], which results in the follower modal forces:

$$
\begin{aligned}
\boldsymbol{\eta}_a = Q_\infty \Bigg(& \boldsymbol{\mathcal{A}}_0 \boldsymbol{q}_0 + \frac{c}{2U_\infty} \boldsymbol{\mathcal{A}}_1 \boldsymbol{q}_1 + \left(\frac{c}{2U_\infty} \right)^2 \boldsymbol{\mathcal{A}}_2 \dot{\boldsymbol{q}}_1 \\
& + \boldsymbol{\mathcal{A}}_{g0} \boldsymbol{v}_g + \frac{c}{2U_\infty} \boldsymbol{\mathcal{A}}_{g1} \dot{\boldsymbol{v}}_g + \left(\frac{c}{2U_\infty} \right)^2 \boldsymbol{\mathcal{A}}_{g2} \ddot{\boldsymbol{v}}_g + \sum_{p=1}^{N_p} \boldsymbol{\lambda}_p \Bigg)
\end{aligned}
\tag{3}
$$

where the $\boldsymbol{\mathcal{A}}_i s$ are real matrices, c is the reference chord, $Q_\infty = \frac{1}{2}\rho_\infty U_\infty^2$ is the dynamic pressure, $\boldsymbol{\lambda}_p$ the aerodynamic states and N_p the number of lags. The coupling of the structure and aerodynamic equations combined with the aerodynamic lags gives the final ODE system:

$$
\begin{aligned}
\dot{\boldsymbol{q}}_1 &= \hat{\boldsymbol{\Omega}} \boldsymbol{q}_2 - \hat{\boldsymbol{\Gamma}}_1 : (\boldsymbol{q}_1 \otimes \boldsymbol{q}_1) - \hat{\boldsymbol{\Gamma}}_2 : (\boldsymbol{q}_2 \otimes \boldsymbol{q}_2) + \hat{\boldsymbol{\eta}} \\
\dot{\boldsymbol{q}}_2 &= -\boldsymbol{\omega} \odot \boldsymbol{q}_1 + \boldsymbol{\Gamma}_2^\top : (\boldsymbol{q}_2 \otimes \boldsymbol{q}_1) \\
\dot{\boldsymbol{\lambda}}_p &= Q_\infty \boldsymbol{\mathcal{A}}_{p+2} \boldsymbol{q}_1 + Q_\infty \boldsymbol{\mathcal{A}}_{p+2} \dot{\boldsymbol{v}}_g - \frac{2U_\infty \gamma_p}{c} \boldsymbol{\lambda}_p
\end{aligned}
\tag{4}
$$

the aerodynamic added-mass effect has been moved to the left-hand side such that $\mathbf{A}_2 = (\boldsymbol{I} - \frac{\rho_\infty c^2}{8} \boldsymbol{\mathcal{A}}_2)^{-1}$, and it couples all DoF in $\boldsymbol{q}_1$. Thus the natural frequency terms become $\hat{\boldsymbol{\Omega}} = \mathbf{A}_2 \text{diag}(\boldsymbol{\omega})$ and the nonlinear terms $\hat{\boldsymbol{\Gamma}} = \mathbf{A}_2 \boldsymbol{\Gamma}$. The effect of all external forces, aero, $\boldsymbol{\eta}_a$, gravity, $\boldsymbol{\eta}_g$, and others, $\boldsymbol{\eta}_f$, are combined in such that $\hat{\boldsymbol{\eta}} = \mathbf{A}_2 \left(\left(\boldsymbol{\eta}_a - \frac{\rho c^2}{8} \boldsymbol{\mathcal{A}}_2 \dot{\boldsymbol{q}}_1 \right) + \boldsymbol{\eta}_g + \boldsymbol{\eta}_f \right)$. The calculation of nodal position vectors, $\boldsymbol{r}_a$, and rotation matrices, $\boldsymbol{R}_{ab}$ is a postprocessing step. The rotations are needed, however, within the solution process when gravity, or other dead forces, are active (forces are naturally given in the material frame of reference and so those forces need to be brought back to the inertial frame). Quaternions $\boldsymbol{\zeta} = [\zeta_0, \zeta_1, \zeta_2, \zeta_3](s,t) = [\zeta_0, \boldsymbol{\zeta}_x](s,t)$ can be used to parameterize the rotation, $\boldsymbol{R}_{ab}$, such that given the angular velocity, $\boldsymbol{\omega}_x$, which is part of the velocity main variable, $\boldsymbol{x}_1 = [\boldsymbol{v}_x, \boldsymbol{\omega}_x]$,

$$
\dot{\boldsymbol{\zeta}} = \begin{bmatrix} \dot{\zeta}_0 \\ \dot{\boldsymbol{\zeta}}_x \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \boldsymbol{\omega}_x^\top \boldsymbol{\zeta}_x \\ \frac{1}{2} (\zeta_0 \boldsymbol{\omega}_x - \tilde{\boldsymbol{\omega}}_x \boldsymbol{\zeta}_x) \end{bmatrix}
\tag{5}
$$

the quaternions, one per node, would be added to Eqs. 4 and march in time; the rotations can be extracted at every step as

$$
\begin{aligned}
\boldsymbol{R}_{ab} =& \boldsymbol{\zeta}_x \otimes \boldsymbol{\zeta}_x + \zeta_0^2 \boldsymbol{I}_3 + 2\zeta_0 \tilde{\boldsymbol{\zeta}}_x + (-(\boldsymbol{\zeta}_x \cdot \boldsymbol{\zeta}_x) \boldsymbol{I}_3 + \boldsymbol{\zeta}_x \otimes \boldsymbol{\zeta}_x) \\
=& \left(2\boldsymbol{\zeta}_x \otimes \boldsymbol{\zeta}_x + (\zeta_0^2 - \boldsymbol{\zeta}_x \cdot \boldsymbol{\zeta}_x) \boldsymbol{I}_3 \right) + 2\zeta_0 \tilde{\boldsymbol{\zeta}}_x
\end{aligned}
\tag{6}
$$

note that the first parenthesis in the second equality of this equation is the symmetric part of the rotation and the last term the antisymmetric part.
Alternatively, the rotation and position in the inertial reference system can be calculated by integration of strains along the domain, as in the Frenet-Serret formulas of differential geometry. Following definition of strains and curvatures, packed in the variable $\boldsymbol{x}_3 = [\boldsymbol{\gamma}, \boldsymbol{k}]$, we have

$$\begin{aligned} \boldsymbol{R}'_{ab} &= \boldsymbol{R}_{ab}\tilde{\boldsymbol{k}} \\ \boldsymbol{r}'_a &= \boldsymbol{R}_{ab}(\boldsymbol{\gamma} + \boldsymbol{e}_x) \end{aligned} \tag{7}$$

where $\boldsymbol{e}_x = [1, 0, 0]$. Analytical solutions to Eq. (7) can be obtained when the strain is assumed constant between nodes and a piecewise constant integration is carried out, as is the case in the current implementation. If a component in the load-path is discretized in n+1 points, strain and curvatures are defined in the mid-points of the spatial discretization (n in total). γ_n and κ_n are constant within the segment $s_{n-1} \leq s \leq s_n$, and the position and rotation matrix after integration are

$$\begin{aligned} \boldsymbol{R}_{ab}(s) &= \boldsymbol{R}_{ab}(s_{n-1})\boldsymbol{\mathcal{H}}^0(\boldsymbol{k}, s) \\ \boldsymbol{r}_a(s) &= \boldsymbol{r}_a(s_{n-1}) + \boldsymbol{R}_{ab}(s_{n-1})\boldsymbol{\mathcal{H}}^1(\boldsymbol{k}, s)\,(\boldsymbol{e}_x + \boldsymbol{\gamma}_n) \end{aligned} \tag{8}$$

with the operators $\boldsymbol{\mathcal{H}}^0(\boldsymbol{k}, s)$ and $\boldsymbol{\mathcal{H}}^1(\boldsymbol{k}, s)$ obtained from integration of the exponential function as defined in [27].

Note that when position and rotations are recovered from strain integration, there is still one point that is either clamped or needs to be tracked from integration of its local velocity. In the next section an optimized implementation of this algorithm is shown in JAX.
Lastly, once the nonlinear solution of the condensed model is computed, the corresponding full 3D state is calculated via a two postprocessing steps: firstly the displacements of the cross-sectional nodes linked to the reduced model via the interpolation elements are computed using the positions and rotations of the latter; secondly, Radial Basis Functions (RBFs) kernels are placed on those cross-sections, thus building an intermediate model that is utilised to extrapolate the positions of the remaining nodes in the full model. This paves the way for a broader multidisciplinary analysis where CFD-based aerodynamic loading could be used for the calculation of the nonlinear static equilibrium, and also with the transfer of the full deformed state back to the original FE solver to study other phenomena such as local buckling.

2.2 Computational implementation

One of the main contribution of this work is a new computational implementation that achieves accelerations of over 2 orders of magnitude with respect to its predecessor[2]. In addition, a highly modular, flexible architecture based on software design patterns has been put in place, which was further described in [28]. Moreover, the resulting nonlinear aeroelastic framework is suitable for modern hardware architectures and able to compute sensitivities via algorithmic differentiation (AD), as will be demonstrated herein. The key enabler was moving from standard Python to a highly vectorised, JAX-based numerical implementation. JAX is a Python library designed for high-performance numerical computing with focus on machine learning activities [16]. It combines XLA (accelerated linear algebra) and Autograd, the former being a compiler that optimises models for different hardware platforms, the latter is an Automatic Differentiation (AD) tool in Python. Moreover, its extensible system of composable function transformations provides a set of important features for Computational Science as illustrated in Fig. 2. For

[2]Both the new implementation and the examples of this paper can be found at `https://github.com/ACea15/FENIAX`

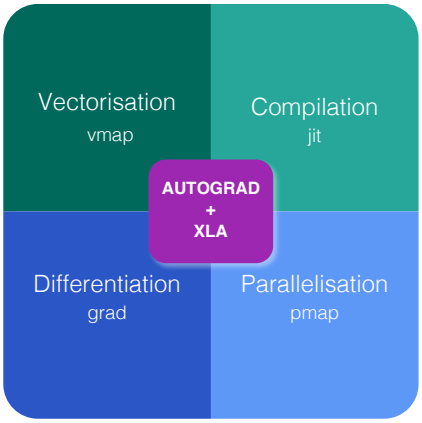


Figure 2: JAX capabilities for modern scientific computing

instance, the `vmap` function allows for complex vectorisation operations and the `pmap` function for Single-Program Multiple-Data (SPMD) parallelisation. Both forward and reverse mode automatic differentiation are supported. Finally the just-in-time compilation (`jit`) relies on the XLA engine to compile and execute functions on CPUs but also on accelerators such as GPUs and TPUs, offering a versatile solution for seamlessly connecting the software to various types of hardware without requiring extra CUDA code, or a Domain Specific Language.
The tensor structure of the main components in the solution process is illustrated in Fig. 3 in the sequential order they are computed, together with the asymptotic time and space complexities. The discretization of the problem comprises N_n number of condensed nodes, N_m modes used in the reduced order model and N_t time steps in the solution (if the problem is static, N_t represents a ramping load stepping scheme). The intrinsic modes, $\Phi, \Psi \in \mathbb{R}^{N_m \times 6 \times N_n}$ are computed from the eigenvalue solution and the coordinates $\boldsymbol{X}_a \in \mathbb{R}^{3 \times N_n}$ of the active nodes. The nonlinear couplings, $\boldsymbol{\Gamma} \in \mathbb{R}^{N_m \times N_m \times N_m}$ are calculated next, from which the system in Eq. 4 is assembled and solved to yield the solution states $\boldsymbol{q} \in \mathbb{R}^{N_t \times N_s}$. Local velocities, internal forces and strain fields $\boldsymbol{X}_{1,2,3} \in \mathbb{R}^{N_t \times 6 \times N_n}$ are computed as a product of the corresponding intrinsic modes and states, and their integration leads to the position tensor, r_a with similar structure. In some cases, such as when gravity forces are included, the evolution of the rotational matrix, $\boldsymbol{R}_{ab}$, needs to be solved for too.

A simplified version of the intrinsic modes evaluation is given in Algorithm 1 with input the FE model, the corresponding eigenvalue solution, and a config object that encapsulates input details such as the matrices required for averaging, M_{avg}, or computing the differences between adjacent nodes, M_{diff}, which are built from the connectivities in the load-paths.

The various functions in the algorithm make heavy use of JAX `vmap` function to vectorise the contraction and expansion operations. For instance, the function `moment_force` is cast in Algorithm 2 and connects together two `vmap` operations. The asymptotic time complexity of Algorithm 1 is driven by the calculation of the internal moments associated with internal forces since for each node a sum through the path-loads is carried out. Arguably a $O(nlog(n))$ or even $O(n)$ depending on the graph rather than $O(n^2)$ could be achieved with standard for-loops and additional variables to describe the graph since there is repetition in the process; this part, however, is not performance critical and no further optimisation is deemed required.

The function created by `vmap` loops the inputs through the axes specified in `in_axes`, piping the resulting vector to the function given to `vmap` as the first argument, and the outputs are saved across the axes given in `out_axes`. With the intrinsic modes computed, the algorithm

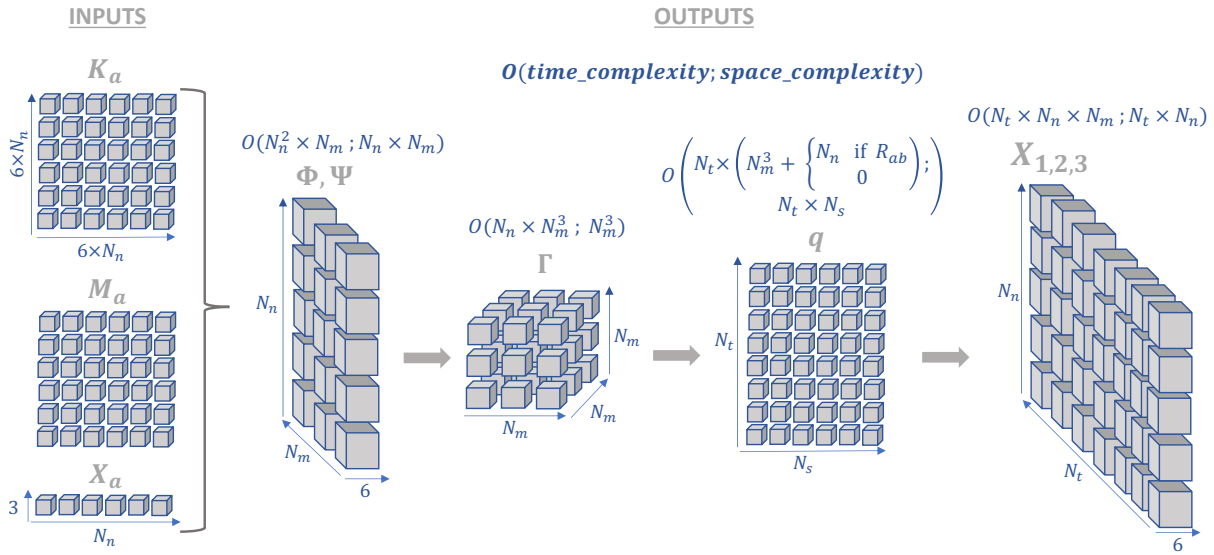


Figure 3: Main data components in the sequential solution process together with their associated time and space complexities

to compute the nonlinear couplings in 2 is shown below:

The new capabilities come at the expense of a higher restriction in the way the code is written. Compilation and transformations in JAX only work for functionally pure programs, which pushes the software to comply with a nonconventional functional paradigm. Some of these characteristics are pure functions, i.e. functions that have no side effects, input/output stream management needs to be placed outside the numerical algorithms or immutability of arrays. These very constraints allow to achieve the capabilities describe above via the many abstractions implemented internally in the library. An example of this restrictions is the effectively impossibility of jit-compile conventional for-loops (they are unrolled in the compilation therefore any input to the loop should always be small). The integration of strains to obtain position and rotation fields as defined in Eq. 8 is a good example of a recursive operation that requires more than vectorised operations to solve, as opposed to the previous algorithms. The function `jax.lax.scan`, which can be differentiated both in forward and backward modes, provides a fast solution for this operation where a previous computation in the loop has to be carried forward (in our case every new computed position and rotation matrix are passed to the next calculation). A greatly simplified version is shown in Algorithm 4 that has as inputs the position and rotation matrix of the first node (given as clamped or calculated via quaternion integration), the initial position of the nodes and the strain field, X_3. The `scan` function iterates through the first axis of the input tensor `xs` and also takes the initial carry state in the variable $init$. Such an abstraction usually imposes various concatenation and deconcatenation operations of the data in the process. In the real algorithm an additional for-loop wraps this function to account for the various components in the aircraft (wings, fuselage, tail, etc.) that do not have contiguous nodes to perform the integration. This for-loop is unrolled in the `jit` compilation but this is not problematic since a very small number of components made up the whole aircraft.

Algorithm 1: JAX-based computation of intrinsic modal shapes

input : FEM and eigenvalue solution: $\boldsymbol{X}_a, \boldsymbol{K}_a, \boldsymbol{M}_a, \boldsymbol{\Phi}_0, \boldsymbol{\omega}$, and $\boldsymbol{config}$ object
output: Intrinsic modal shapes
begin

```
import jax.numpy as jnp
X_m ← jnp.matmul(X_a^⊤, M_avg)    ▷ Get mid-node coordinates
X_d ← jnp.matmul(X_a, M_diff)     ▷ vector difference between
 contigous nodes
Δ_s = jnp.linalg.norm(X_d, axis=0)
Φ_1 ← Φ_0
Φ_1m ← jnp.tensordot(Φ_1, M_avg, axes=(2, 0))      ▷ Velocity
 modes at mid-node locations
ψ_1v ← jnp.matmul(M_a, Φ_0)          ▷ Momenta modes
ψ_1 ← reshape_modes(ψ_1v, Nmodes, Nnodes)     ▷ Nm×6×Nn
Φ_2fv ← jnp.matmul(K_a, Φ_0)    ▷ Internal forces and moments
 associated to modal shapes (Nm×6×Nn)
Φ_2fv ← reshape_modes(Φ_2fv, Nmodes, Nnodes)
Φ_2f ← jnp.tensordot(Φ_2fv, M_paths, axes=(2, 0))      ▷ Sum of
 internal forces and moments (Nm×6×Nn)
X_3 ← coordinates_difftensor(X, X_m)   ▷ mid-node vector to
 every other node in the reduced model (3×Nn×Nn)
X_3tilde ← -axis_tilde(X_3)     ▷ Cross-product in matrix form
 (6×6×Nn×Nn)
φ_2mn ← moment_force(φ_2v, X_3tilde)   ▷ Moment distribution due
 to nodal forces (Nm×6×Nn×Nn)
φ_2m ← moment_contraction(φ_2mn, M_paths)     ▷ Sum of internal
 moments due to forces (Nm×6×Nn)
Φ_2 ← Φ_2f + φ_2m  ▷ Total value internal forces and moments
E_φ ← ephi(E, φ_1m)                  ▷ E times
φ_1d ← jnp.tensordot(Φ_1, M_diff, axes=(2, 0))      ▷ Velocity
 mode variation across nodes (Nm×6×Nn)
ψ_2 ← - φ_1d/Δ_s + E_φ
```

Algorithm 2: Internal moments due to internal forces

Function `moment_force(`$\boldsymbol{\phi}_{2v}$, $\boldsymbol{X}_{3tilde}$`)`**:**

```
f1 ⟵ vmap(lambda u, v:  jnp.tensordot(u, v, axes=(1,1),
 in_axes=(None, 2), out_axes=2)) ;
f2 ⟵ vmap(f1, in_axes=(2, 3), out_axes=3) ;
f3 ⟵ f2(φ_2v, X_3tilde)              ▷ Nm×6×Nn×Nn ;
```

return f3;

3 RESULTS

Three different cases are presented to meet the following requirements: a) validate the current structural and aeroelastic implementation against results from theory and MSC Nastran; b)

Algorithm 3: Nonlinear Couplings implementation in JAX

input : Intrinsic modal shapes: $\mathbf{\Phi}_1$, $\mathbf{\Phi}_2$, $\mathbf{\Psi}_1$, $\mathbf{\Psi}_2$; nodal differences, Δ_s
output: Intrinsic nonlinear modal couplings $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$
begin

```
f1 ⟵ vmap(lambda u, v:  jnp.tensordot(𝓛1(u), v, axes=(1,
 1), in_axes=(1, 2), out_axes=2)) ▷ iterate through nodes
f2 ⟵ vmap(f1, in_axes=(0, None), out_axes=0)
L1 ⟵ f2(Φ1, ψ1)                  ▷ Nm×6×Nm×Nm
Γ1 ⟵ jnp.einsum(isn,jskn→ijk, Φ1, L1)
f3 ⟵ vmap(lambda u, v:  jnp.tensordot(𝓛2(u), v, axes=(1,
 1), in_axes=(1, 2), out_axes=2))     ▷ iterate nodes
f4 ⟵ vmap(f3, in_axes=(0, None), out_axes=0)
L2 ⟵ f4(Φ2, ψ2)                  ▷ Nm×6×Nm×Nm
Γ2 ⟵ jnp.einsum(isn,jskn,n→ijk, Φm1, L2, Δs)
```

Algorithm 4: Strain and curvature integration via JAX scan

input : ra(0), Rab(0), Xa, X3
output: positional, ra, and rotation, Rab, tensor fields
begin

```
Function integrate_strains(carry, x):
    f_Rab ⟵ vmap(lambda Rab_i, H0_i:  Rab_i H0_i)
    f_ra ⟵ vmap(lambda ra_i, Rab_i, H1_i, strain_i:  ra_i
     + Rab_i H1_i ([1,0,0] +   )
       strain_i)
    strain, kappa, ds ⟵ deconcatenate(x)
    Rab_carry, ra_carry ⟵ deconcatenate(carry)
    Rab ⟵ f_Rab(Rab_carry, H0(kappa, ds))
    ra ⟵ f_ra(ra_carry, Rab_carry, H1(kappa, ds), strain)

    y ⟵ concatenate([Rab, ra])
    carry ⟵ y
    return carry, y
ds ⟵ jnp.linalg.norm(Xa[1:]-Xa[:-1])
init ⟵ concatenate([Rab0, ra0])
xs ⟵ concatenate([X3, ds])
carry, y ⟵ jax.lax.scan(integrate_strains, init, xs)
Rab, ra ⟵ deconcatenate(y)
```

show the computational advantage of deploying the code on GPUs; c) showcase and verify the differentiable capabilities of the code. Starting with a canonical case of a free-flying structure, moving to a representative aircraft without fuselage or engines, and ending with a fully fledged aircraft model built by Airbus for research purposes.

3.1 Canonical case: unsupported dynamics of very flexible structure

This example exemplifies the ability of our solvers to turn a generic linear free-free finite-element model into a fully nonlinear solution that accounts for the rigid-body dynamics coupled with large elastic deformations. It has already been presented in [13], though the novelties introduced herein are the new optimised implementation that can run on accelerators and the approach to recover the full 3D state from the reduced model. The beam version of this structure was first studied by Simo and Vu-Quoc [29] and has served to verify several implementations of nonlinear beam dynamics with rigid body motions [30]. A straight structure of constant square cross section (side = 3, wall thickness = 3/10) is built consisting of 784 shell elements linked to 50 spanwise nodes via interpolation elements as depicted in Fig. 4 together with the material properties and two types of loading: firstly, a dead-force in the x-direction and dead-moment in the z-direction that yield a planar motion in the x-y plane; and secondly, the addition of a moment in the y-direction which produces a three dimensional motion.

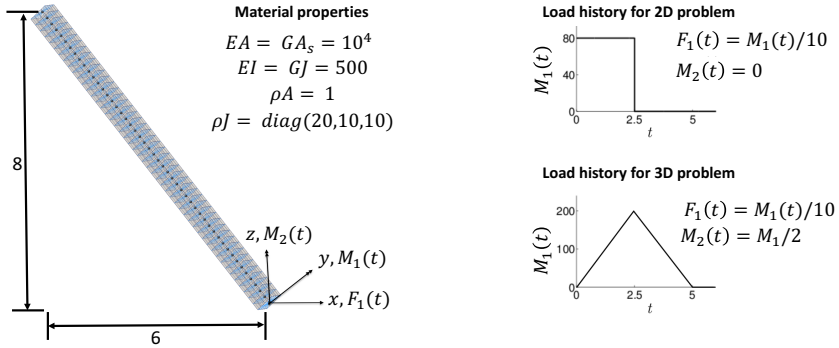


Figure 4: FFS geometry, material properties and load cases

The free-flying evolution of the 3D model is shown in Fig. 5 for the planar motion and Fig. 6 for the loads giving rise to the full 3D deformations. It worth remarking the latter motion also exhibits large torsional deformations which are combined with the also large bending displacements and rigid-body modes.

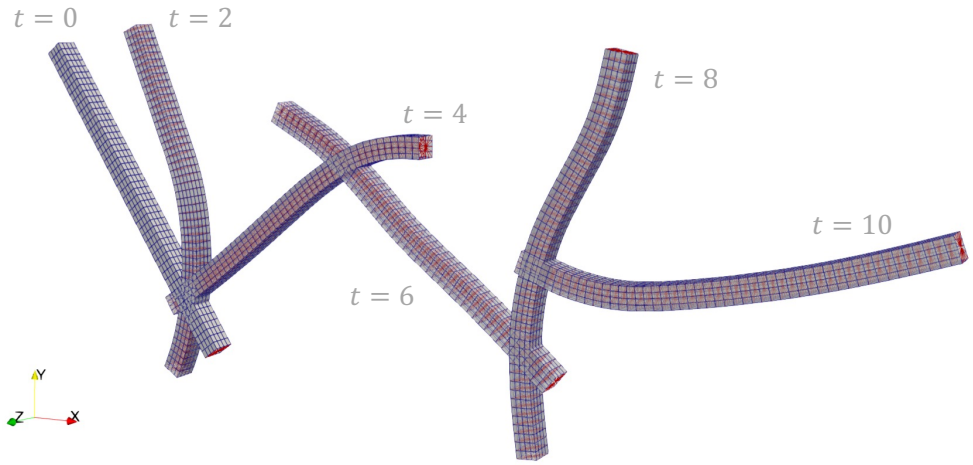


Figure 5: Free-flying structure in the 2D plane

Because the applied load is a dead force we can track the position of the center-of-gravity (CG) analytically as a verification exercise. Furthermore, the highly nonlinear nature of this problem makes it a good example to showcase the strength of accelerators for large problems and to gain insights as to when it might be better to deploy the codes in standard CPUs instead. Therefore

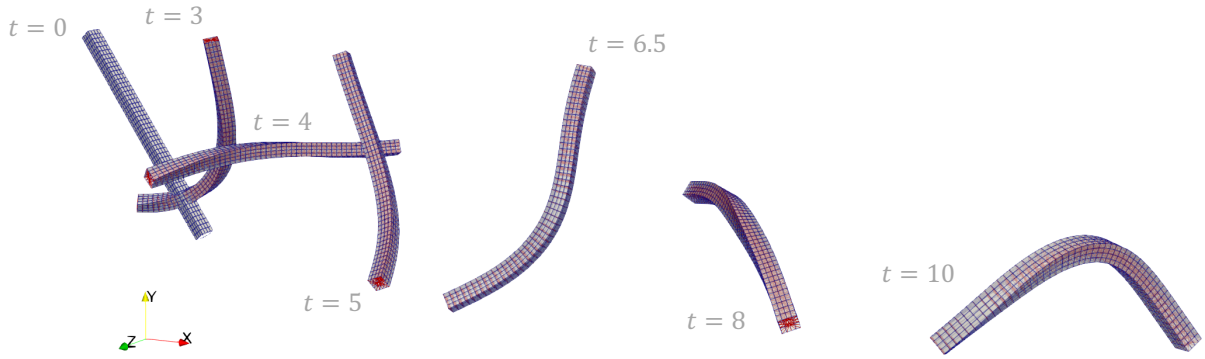


Figure 6: Free-flying structure in the 3D plane

we perform a sweep with the number of modes kept in the solution from 50 to 300, which determines the size of the system to be solved. The full modal basis is employed at 300 modes and due to the nonlinear cubic term this entails operations of the order of $O(10^7)$ at every time step of the solution, making it a good case for accelerators. The increase in the number of modes also restricts the incremental time step used in the explicit solver to preserve stability. Table 1 shows both computational time and CG error for the planar case in two scenarios: linking the integration time-step to the largest eigenvalue λ in the solution $dt = \lambda^{-0.5}$; and fixing it to $dt = 10^{-3}$. The error metric is defined as the L-2 norm divided by the time steps. Computations have been carried out in AMD EPYC 7742 CPU processors and Nvidia GPU RTX 6000 at the Imperial College cluster.

Table 1: FFB computational times in seconds and CG error

Nmodes	CPU (time/err)	GPU (time/err)	CPU (time/err)	GPU (time/err)
50	7/1.3e-1	9.9/1.3e-1	42/2.1e-2	58/2.1e-2
100	9.3/5.7e-2	10.4/5.7e-2	184/1.2e-2	65/1.2e-2
150	34/2.2e-2	14/2.2e-2	287/5.6e-3	67/5.6e-3
200	79/2e-3	22/2e-3	421/7.2e-4	76/7.2e-4
250	474/5.3e-4	38/5.3e-4	893/2.7e-4	94/2.7e-4
300	1869/2.54e-5	111/2.54e-5	1869/2.54e-5	111/2.54e-5

Fig. 7 and 8 illustrate the times and error results in the table for the second case with fixed time step. The gain in performance from the GPU is is more impressive the larger the system to solve, and for the full modal basis the CPU takes more than 31 minutes versus the less than 2 minutes in the GPU. Computational times in the 3D problem are similar and the error on the CG position is slightly higher: for the 300 modes case, the error is $6.9e-5$ versus the $2.54e-5$ of the planar case.

3.2 Representative aeroplane model: structural verification

A representative FE model of a full aircraft without engines is used to demonstrate a versatile solution that accounts for geometric nonlinearities in a very efficient manner and only needs modal shapes and linear FE matrices from a generic FE solver as inputs. Another of the goals set for this work was to achieve an equally flexible strategy in the automatic calculation of derivatives across the various solvers in the code. The structural static and dynamic responses and their sensitivities with respect to input parameters are verified against MSC Nastran and finite differences respectively.

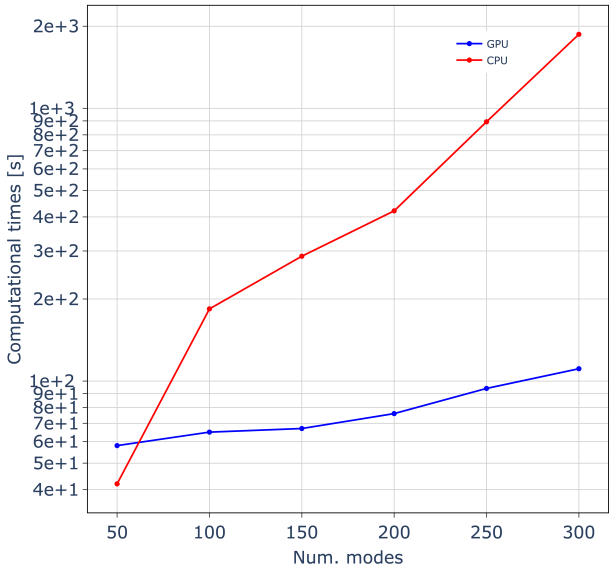


Figure 7: Performance CPU vs GPU comparison in free-flying structure (fixed time step)

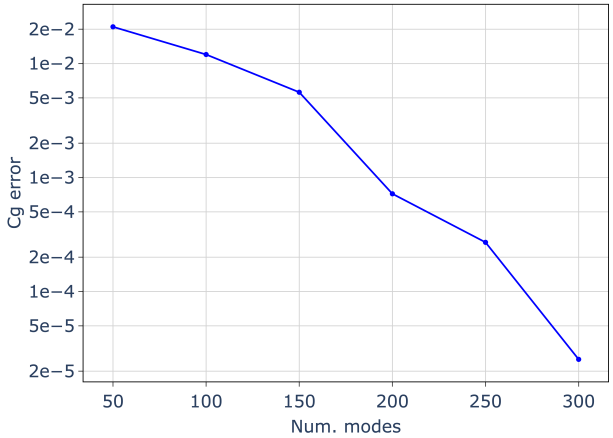


Figure 8: Error metric CG position for planar case

The aircraft's main wing is composed of wing surfaces, rear and front spars, wing box and ribs with composite materials employed in the construction. Flexible tail and rear stabiliser are rigidly attached to the wing, as shown in Fig. 9. This aircraft was first shown in [25] and is a good test case as it is not very complex yet representative of aircraft FE models and it is available open source.

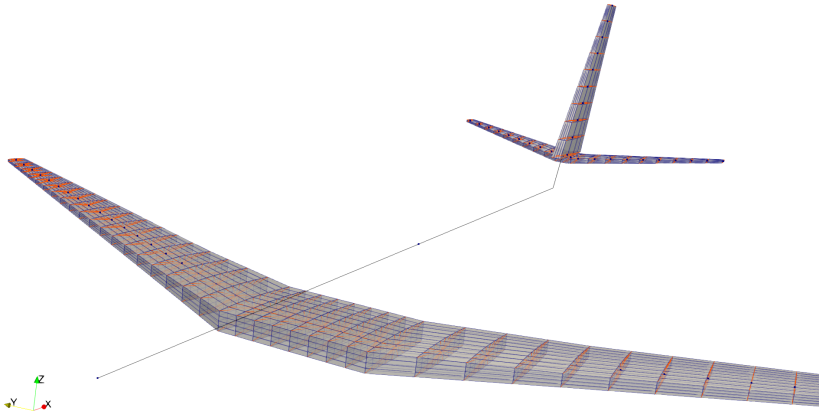
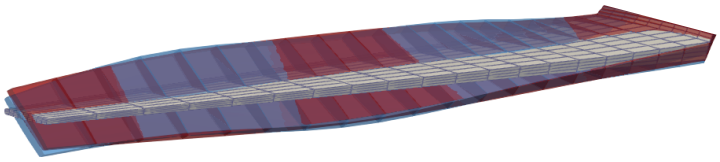
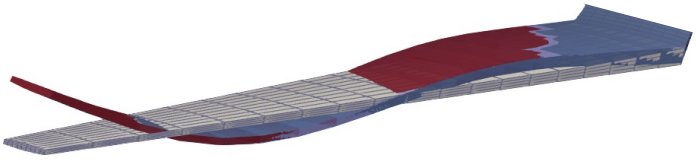


Figure 9: Representative aeroplane structural FE model

A Guyan reduction is employed in the reduction process and Fig. 10 illustrates the accuracy of the condensed model by comparing the 3D modal shapes. No differences can be appreciated for the first few modes (the lowest frequency corresponding to a bending mode agrees in both models at $\omega_1 = 4.995$ rads/s) so we show higher frequency modes: a high order bending mode ($\omega_{10} = 60.887/60.896$ rads/s in full versus reduced models) and a torsional mode ($\omega_{20} = 107.967/107.969$ rads/s). This very good preservation of the full model leads to an excellent accuracy in the static and dynamic results presented below. It is important to remark this aircraft model is very conventional without high-aspect ratio wings. Therefore while this modelling strategy would not be suitable for every engineering structure, as long as there is a dominant dimension and deformations in the other two remain small (as is the case in high level descriptions of aircraft, bridges or wind turbines) it has been found to produce very good approximations when compared with full dimensional solutions. The computations in this section were carried out on a standard CPU with a i7-6700 processor.



(a) Torsional mode



(b) Bending mode

Figure 10: Full VS reduced order models modal shape

3.2.1 Geometrically nonlinear static response

The static equilibrium of the aircraft under prescribed loads is first studied with follower loads normal to the wing applied at the tip of each wing. The response for an increasing load stepping of 200, 300, 400, 480 and 530 KN is computed. Nonlinear static simulations on the original full model (before condensation) are also carried out in MSC Nastran and are included. The interpolation elements in Nastran are used to output the displacements at the condensation nodes for direct comparison with the NMROM results. Geometric nonlinearities are better illustrated by representing a sectional view of the wing as in Fig. 11, where deformations in the z-direction versus the metric $S = \sqrt{x^2 + y^2}$ are shown. MSC Nastran linear solutions (Solution 101) are also included to appreciate more clearly the shortening and follower force effects in the nonlinear computations.

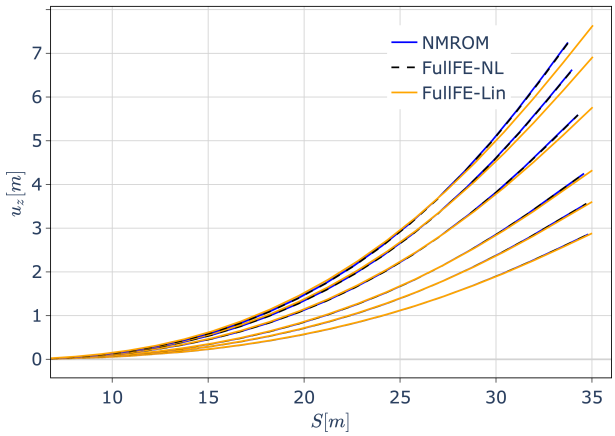


Figure 11: Static geometrically-nonlinear effects on the aircraft main wing

The tolerance in the Newton solver was set to 10^{-6} in all cases. A convergence analysis with the number of modes in the solution is presented in Fig. 12. 5, 15, 30, 50, 100 modes are used to build the corresponding NMROMs. The error metric is defined as the L-2 norm divided by the total number of nodes (only the condenses ones in this case): $\epsilon = ||u_{NMROM} - u_{NASTRAN}||/NumNodes$. It can be seen the solution with 50 modes already achieves a very good solution even for the largest load which produces a 25.6% tip deformation of the wing semi-span, $b = 28.8$ m. The displacement difference with the full FE solution at the tip in this case is less than 0.2%.

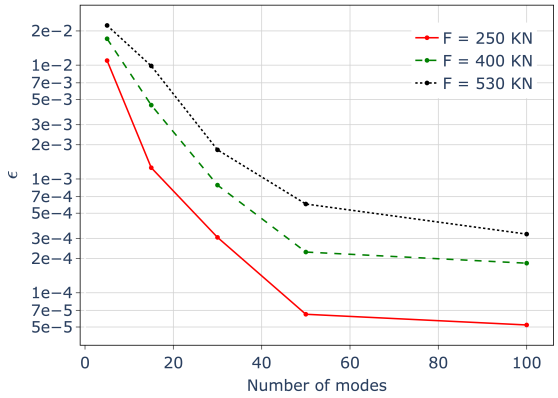


Figure 12: Modal convergence static solution of representative aircraft

The 3D structural response has been reconstructed using the approach in Fig. 1. The nodes connected by the interpolation elements (RBE3s) to the ASET solution are reconstructed first

and subsequently a model with RBFs kernels is used to extrapolate to the rest of the nodes in the full FE. A very good agreement is found against the geometrically-nonlinear Nastran solution (SOL 400). Fig. 14 shows the overlap in the Nastran solution (in red) and the NMROM (in blue) for the 530 KN loading.

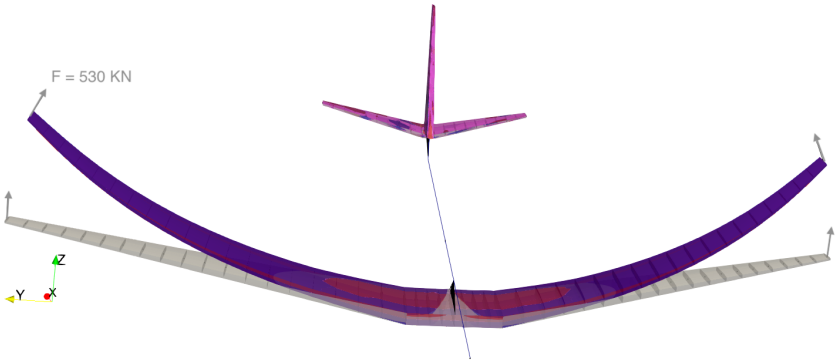


Figure 13: Static 3D solution for a solution with 50 modes and 530 KN loading (Full NASTRAN solution in red versus the NMROM in blue).

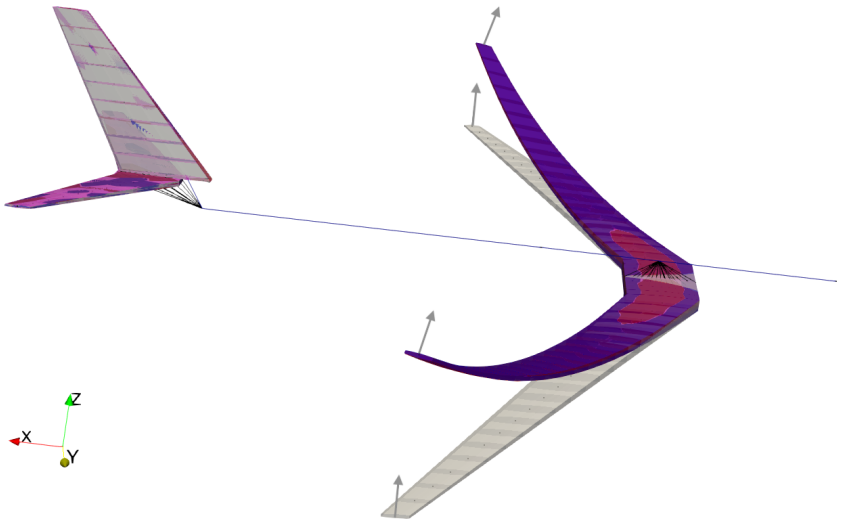


Figure 14: Static 3D solution for a solution with 50 modes and 530 KN loading (Full NASTRAN solution in red versus the NMROM in blue).

The error metric of this 3D solution is also assessed in Fig. 15, for the solution with 50 modes. The discrepancy metric is of the same order than the previously shown at the reduction points. This conveys an important point, that there is no significant accuracy loss in the process of reconstructing the 3D solution.

Next we compare the computational times for the various solutions presented in this section in Table 2. Computations of the six load steps in Fig. 11 are included in the assessment. A near 50 times speed-up is achieved with our solvers compared to Nastran nonlinear solution, which is one of the main strengths of the proposed method. As expected, the linear static solution in Nastran is the fastest of the results, given it only entails solving a linear, very sparse system of equations.

Table 2: Computational times static solution

	NMROM (modes: 5, 15, 30, 50, 100)	NASTRAN 400	NASTRAN 101
Time [s]	6.7, 6.63, 6.79, 7.06, 9.55	345	1.02

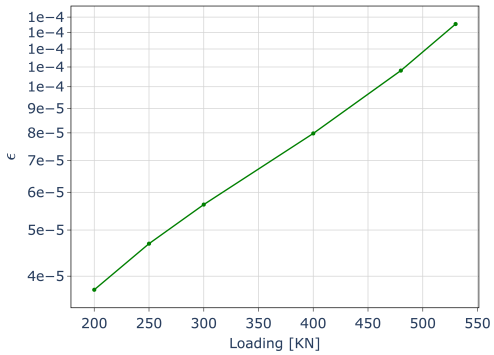


Figure 15: 3D discrepancy comparison between full FE and NMROM solutions

3.2.2 Differentiation of static response

The AD for the static solvers is first verified as follows: the load stepping shown above becomes a pseudo-time interpolation load such that a variable τ controls the amount of loading and we look at the variation of the wing-tip displacement as a function of this τ. If $f(\tau = [1, 2, 3, 4, 5, 6]) = [200, 250, 300, 400, 480, 530]$ KN, with a linear interpolation between points, the derivative of the z-component of the tip of the wing displacements is computed at $\tau = 1.5, 3.5, 5.5$, as show in Fig. 16 with red points. Table 3 shows a very good agreement

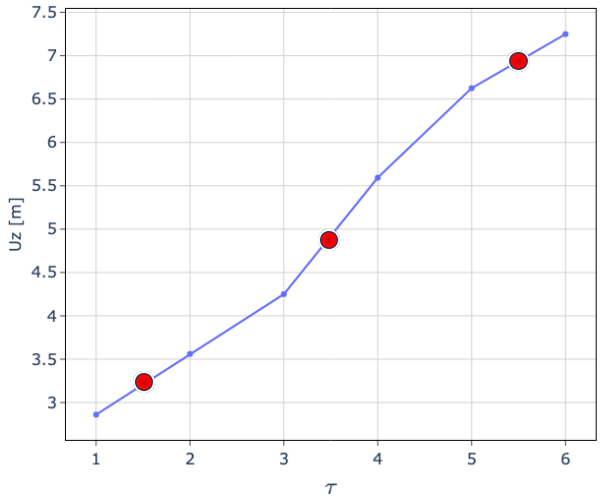


Figure 16: Static tip displacement with pseudo-time stepping load

against finite-differences (FD) with an epsilon of 10^{-3}. Note how the derivative at each of the marked points corresponds approximately to the slope in the graph at those very points. And the biggest slope occurs precisely in between tau of 4 and 5 when the prescribed loading undergoes the biggest change from 300 to 400 KN.

Table 3: AD verification in static problem

τ	$f(\tau)$ [m]	$f'(\tau)$ (AD)	$f'(\tau)$ (FD)
1.5	2.81	0.700	0.700
3.5	4.527	1.344	1.344
5.5	6.538	0.623	0.623

3.2.3 Large-amplitude nonlinear dynamics

This test case demonstrates the accuracy of the NMROM approach for dynamic geometrically-nonlinear calculations and was first introduced in [14]. The right wing of Fig. 9 is considered and dynamic nonlinear simulations are carried out and compared to MSC Nastran linear and nonlinear analysis (SOL 109 and 400, respectively) on the full FE model. A force is applied at the wing tip with a triangular loading profile, followed by a sudden release of the applied force to heavily excite the wing. The force profile is given in Fig. 17. The applied force is then $f_{tip} = \alpha \times \mathbf{f}_{max} f(0.05, 4) = [-2 \times 10^5, 0., 6 \times 10^5] f(0.05, 4)$ where α has been set to 1.

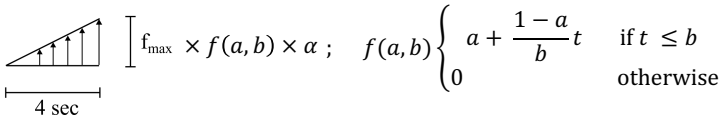


Figure 17: Ramping load profile for dynamic simulation of representative wing

The dynamic response is presented in Fig. 18, where results have been normalised with the wing semi-span ($l = 28.8$ m.). As expected, linear analysis over-predicts vertical displacements and does not capture displacements in the x and y directions. NMROMs were built with 5, 15, 30, 50 and 100 modes. A Runge-Kutta four is used to march the equation in time with time steps corresponding to the inverse of the largest eigenvalue in the NMROM, i.e. $dt = [27.34, 6.62, 2.49, 1.27, 0.575] \times 10^{-3}$ s.

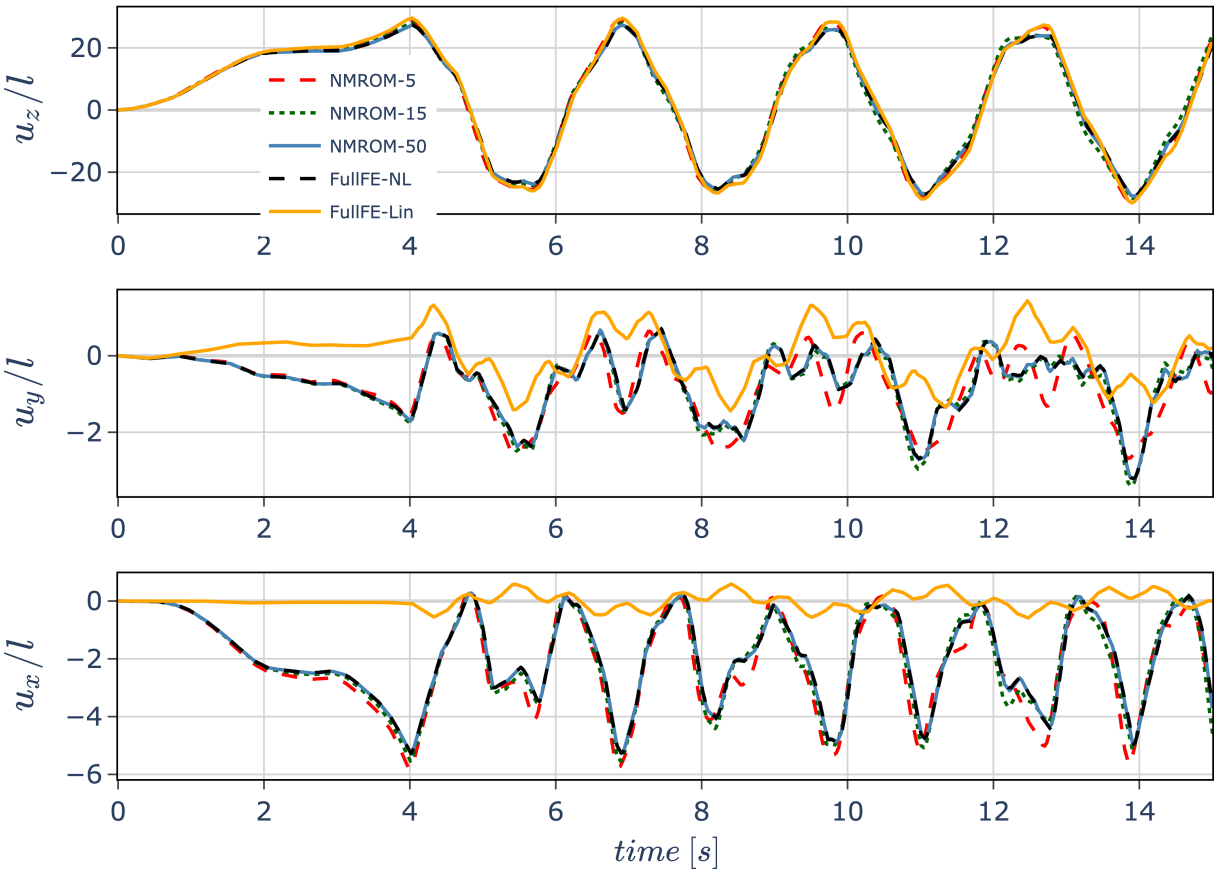


Figure 18: Span-normalised wing-tip displacements

As in the previous example, the 3D shape of the model is retrieved and compared against the full nonlinear dynamic solution as illustrated in Fig. 19 (Nastran solution in yellow and NMROM

with 50 modes in blue). The times at positive and negative peaks are displayed. Even though a wing of such characteristics would never undergo this level of deformations, these results further support the viability of the methodology to solve highly geometrically nonlinear dynamics, on complex models and with minimal computational effort.

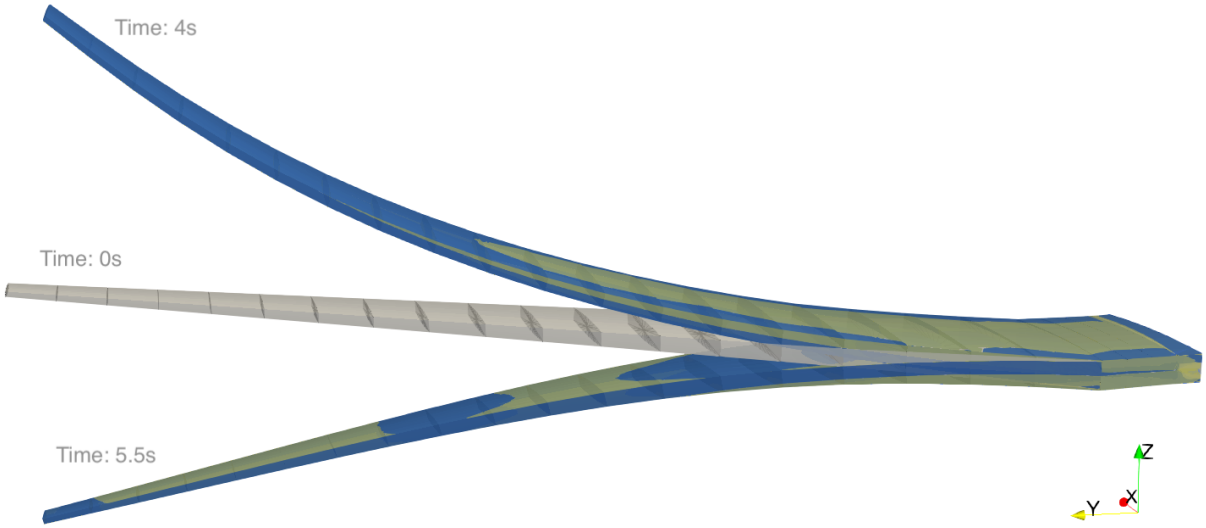


Figure 19: Snapshots of wing 3D dynamic response comparing NMROM (blue) and NLFEM3D (yellow)

Next we look at the differences of the dynamic simulations with the same metric employed above that now evolves in time. Integrator errors accumulate and discrepancies grow with time but still remain small. In fact the differences between Nastran and our dynamic solvers are comparable to the static example with the highest load (around the 5×10^{-5} mark), both cases inducing over 25 percent deformations of the wing semi-span.

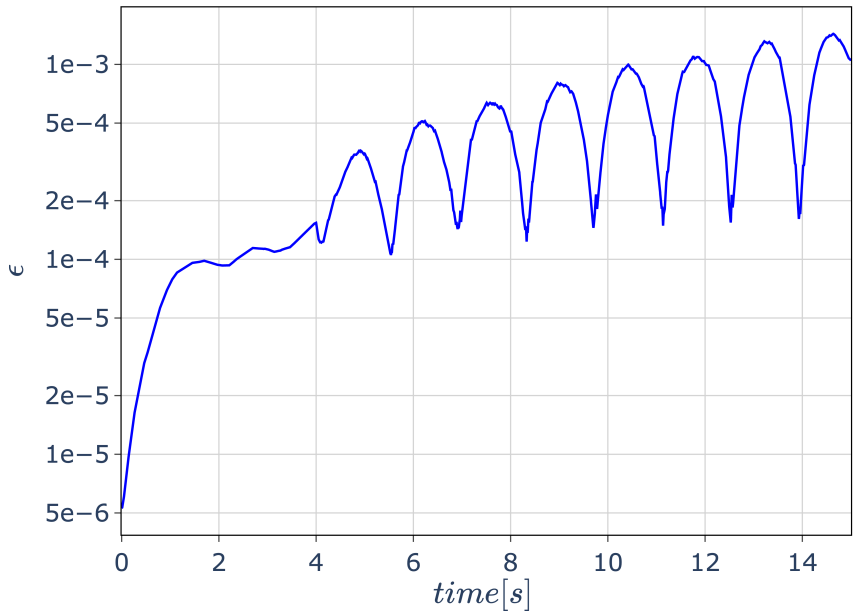


Figure 20: L-2 norm per node differences between Nastran full FE solution and NMROM with 50 modes

An impressive reduction of computational time is achieved by our solvers as highlighted in Table 4. The nonlinear response of the full model in Nastran took 1 hour 22 minutes, which is

over two orders of magnitude slower than the NMROM with 50 modes resolution, which proved very accurate. The significant increase in computational effort when moving from a solution with 50 modes to 100 modes is due to various factors: vectorised operations are limited and the quadratic nonlinearities ultimately lead to O(N_m^3) algorithms; the time-step needs to be decreased for the Runge-Kutta integration to remain stable; the additional overheads that come with saving and moving larger tensors, from the modal shapes, the cubic modal couplings, to the system states (note times shown account for all the steps from start to end of the simulation, including saving all the data for postprocessing).

Table 4: Computational times representative wing dynamic solution

	NMROM (modes: 5, 15, 30, 50, 100)	NASTRAN 400	NASTRAN 109
Time [s]	2.79, 2.92, 4.85, 12.14, 155.3	4920	33.6

3.2.4 Differentiation of dynamic response

We move now to one of the main highlights of this work, i.e. the ability to compute gradients via automatic differentiation in geometrically nonlinear dynamic problems. The maximum root loads occurring in a wing subjected to dynamic loads is a good test case as it can be a critical metric in sizing the aircraft wings, especially high-aspect ratio ones. Thus we look at the variation of the maximum z-component of the vertical internal forces as a function of α in the loading profile of Fig. 17. Effectively, the slope of the loading increases with α. Table 5 shows the derivatives computed using FD with an epsilon of 10^{-4} and AD in reverse-mode on the example with 50 modes resolution. In this case the FD needed tweaking of epsilon while application of AD was straight forward with no need for checkpoints and took around three times the speed of a single calculation.

Table 5: Automatic differentiation in dynamic problem

α	$f(\alpha)$ [KN/m]	$f'(\alpha)$ (AD)	$f'(\alpha)$ (FD)
0.5	1706.7	3587.71	3587.77
1.0	3459.9	3735.26	3735.11
1.5	5398.7	3957.81	3958.31

The increase in the internal loading, $f(\alpha)$, above the linear correlation with α is a consequence of very high frequency dynamics being excited after the ramping load is suddenly released. In fact in the z-component of the wing-tip evolution in Fig. 21 we can see a maximum tip displacement of 4.36m, 7.91m and 10.83m, for $\alpha = 0.5, 1, 1.5$ i.e smaller than the proportional linear response. On the contrary, in Fig. 22 the evolution of the root loads show a response with much higher frequencies and the maximum occurs in the free dynamical response of the wing, which is higher as we increase α.

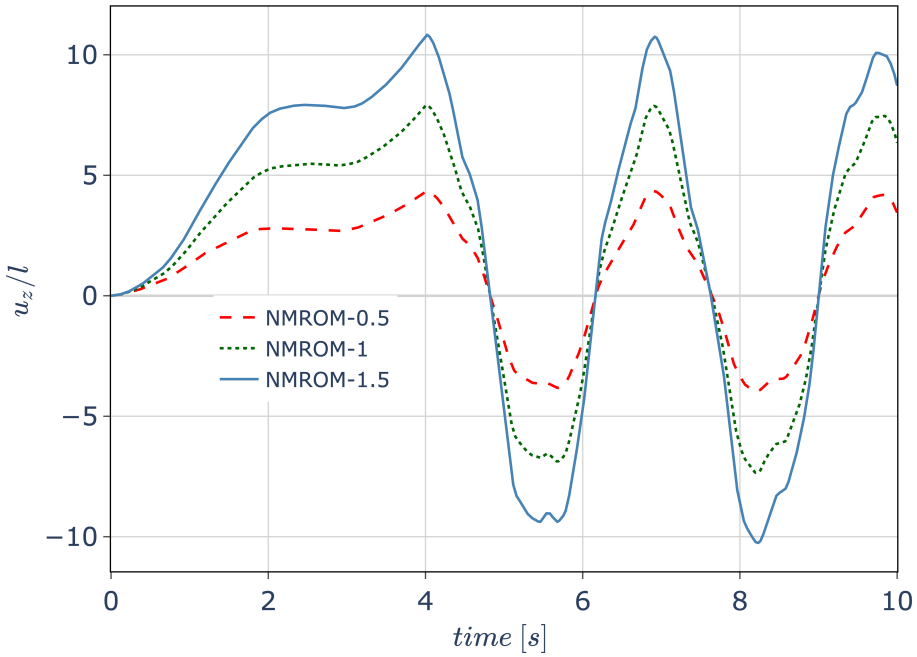


Figure 21: Span-normalised wing-tip z-displacement for the various load profiles

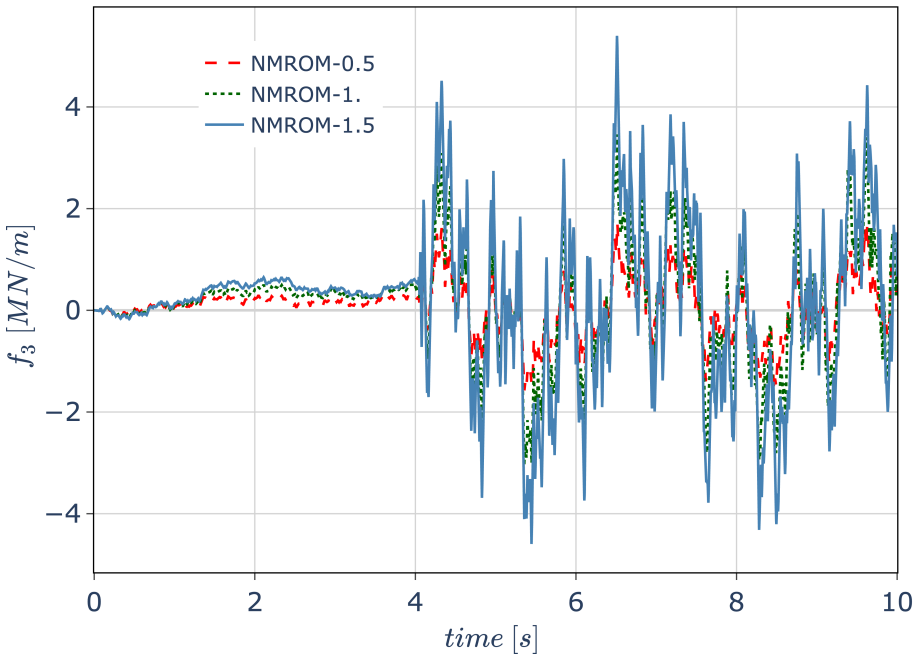


Figure 22: Wing root loads, z-component

3.3 Industrial-level case: full-aircraft gust response

The studies presented in this section are based on a reference configuration developed to industry standards known as XRF1, which is representative of a long-range wide-body transport airplane. The version with a wing-tip extension in [15] is employed to verify a gust response against NASTRAN linear solution. Fig. 23 shows the reference FE model with three modal shapes. The FE model contains a total of around 177400 nodes, which are condensed into 176 active nodes along the reference load axes through interpolation elements. A Guyan or static condensation approach is used for the reduction. The aerodynamic model contains $\sim 1,500$ aerodynamic panels. The simulations are carried out with a modal resolution of 70 modes and a time step in the Runge-Kutta solver of 0.005.

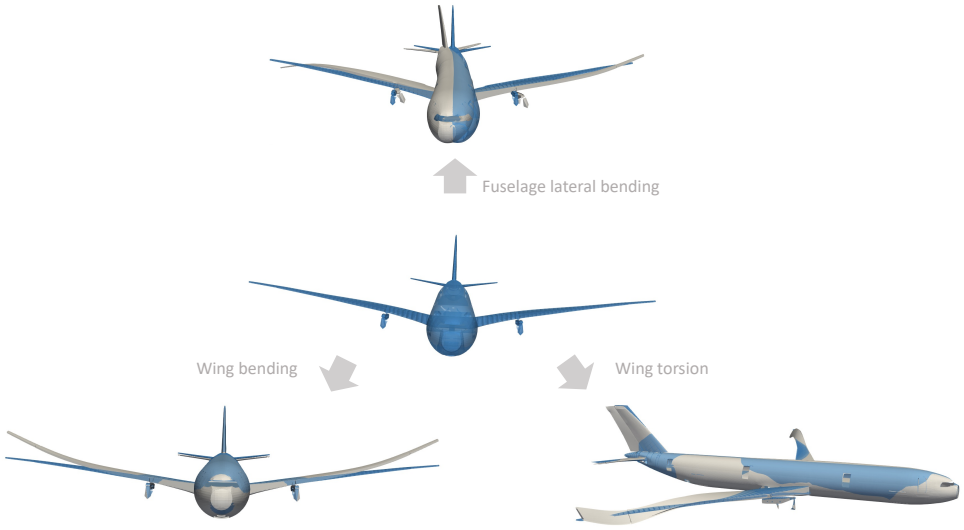


Figure 23: Modified XRF1 reference configuration with characteristic modal shapes

To verify the accuracy of the reduction method we show now a comparison of some of the most relevant natural frequencies in Table 6 (normalized with the lowest frequency). It is common practice in this industrial aeroelastic models to represent the mass model as lumped masses along the load axis (provided by the mass department which thoroughly tracks the multiple components and combines them into sectional lumped points with inertia). As explained above, this makes the Guyan reduction match an exact condensation and explains the accuracy of the results.

While the previous results where purely structural, now the dynamic response to an atmospheric disturbance or gust is computed. This aeroelastic analysis is a requirement for certification purposes and it is one of the main drivers in sizing the wings of high aspect ratio wings. Furthermore, the previous examples showed the advantage of our approach in terms of computational speed, but other than that results could be obtained with commercial software. The geometrically nonlinear aeroelastic response, however, it is not currently available in commercial solutions that are bounded to linear analysis in the frequency domain. Other research codes feature those additional physics, yet are limited to simple models. Thus the added value in the proposed approach comes at the intersection between the nonlinear physics arising from large integrated displacements, computational efficiency and the ability to enhance the models already built for industrial use.

Table 6: Normalised natural frequencies of the modified XRF1 clamped model

Mode Type	ω_{full}	$\omega_{condensation}$	**Error** (%)
First out-of-plane wing bending (1)	1.0	1.0	0.0001
First out-of-plane wing bending (2)	1.04948	1.04949	0.0004
First fuselage bending (3)	1.4285	1.4288	0.025
Fuselage lateral bending + wing out-of-plane bending (4)	1.5093	1.5099	0.038
Wing and pylon lateral roll bending (5)	2.1449	2.1449	0.0007
⋮			
First in-plane wing bending (13)	3.156	3.155	0.03
Second in-plane wing bending (14)	3.25655	3.2566	0.0016
⋮			
First wing torsion (28)	7.675	7.676	0.01
Second wing torsion (29)	7.7368	7.7366	0.002

3.3.1 Linear response for low intensity gust

A verification exercise is introduced first by applying two 1-cos gust shapes at a very low intensity, thus producing small deformations and a linear response. The flow Mach number is 0.81. A first gust is applied that we name as G1 of length 67 m and peak velocity 0.141 m/s, and a second gust, G2, of 165 m and peak velocity of 0.164 m/s. Fig. 24 shows the normalised wing-tip response with our NMROM that accurately reproduces the Nastran 146 solution based on the full FE model.

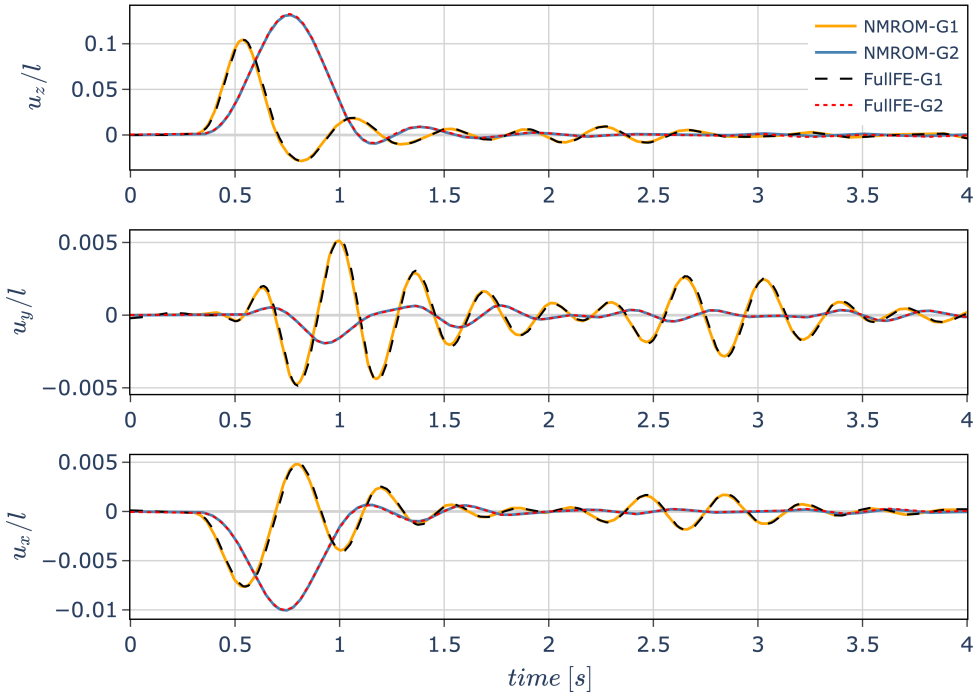


Figure 24: Wing-tip response to low intensity gust

3.3.2 Nonlinear response for high intensity gust

Next we increase the gust intensity by a factor of 200 in order to show the effects of geometric nonlinearities that are only captured by the nonlinear solver. As seen in Fig. 25, there are major

differences in the x and y components of the response due to follower and shortening effects, and a slight reduction in the z-component. These are well known geometrically nonlinear effects that are added to the analysis with no significant overhead.

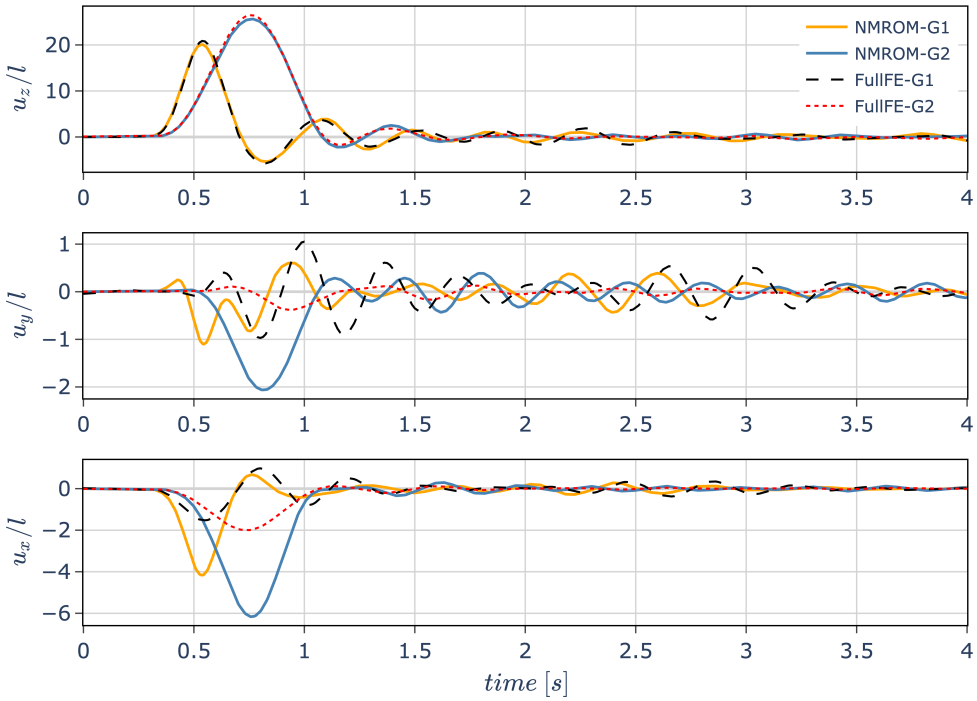


Figure 25: Wing-tip response to high intensity gust

Snapshots of the 3D response are reconstructed for the G1 gust using the method verified above at the time points where tip displacement are maximum and minimum, i.e. 0.54 and 0.84 seconds. The front and side views together with the aircraft reference configuration are shown in Fig. 26.

In terms of simulation times, the nonlinear solution with 70 modes has taken around 17 seconds, while the Nastran solution took around 81 seconds. These 81 seconds do not account for the whole simulation time because that would include the time to sample the DLM aerodynamics which are input into the NMROM as a post-processing step. Instead, the increase in time when adding an extra gust subcase to an already existing analysis is reported, i.e. the difference between one simulation that only computes one gust response and another with two. It is remarkable that the explicit solvers are faster on the nonlinear solution than the linear solution by a commercial software. Besides our highly efficient implementation, the main reason for this might be the Nastran solution involves first a frequency domain analysis and then an inverse Fourier transform to obtain the time-domain results.

3.3.3 Aeroelastic AD verification

Similarly to the examples above, we now validate the AD implementation for the nonlinear aeroelastic response to the gust $G1$. The sensitivity of the six components of the wing root loads are computed with respect to the gust parameters w_g and L_g, and the flow parameter ρ_{inf}. The results are presented in 7. A very good agreement with the finite differences is found with $\epsilon = 10^{-4}$.

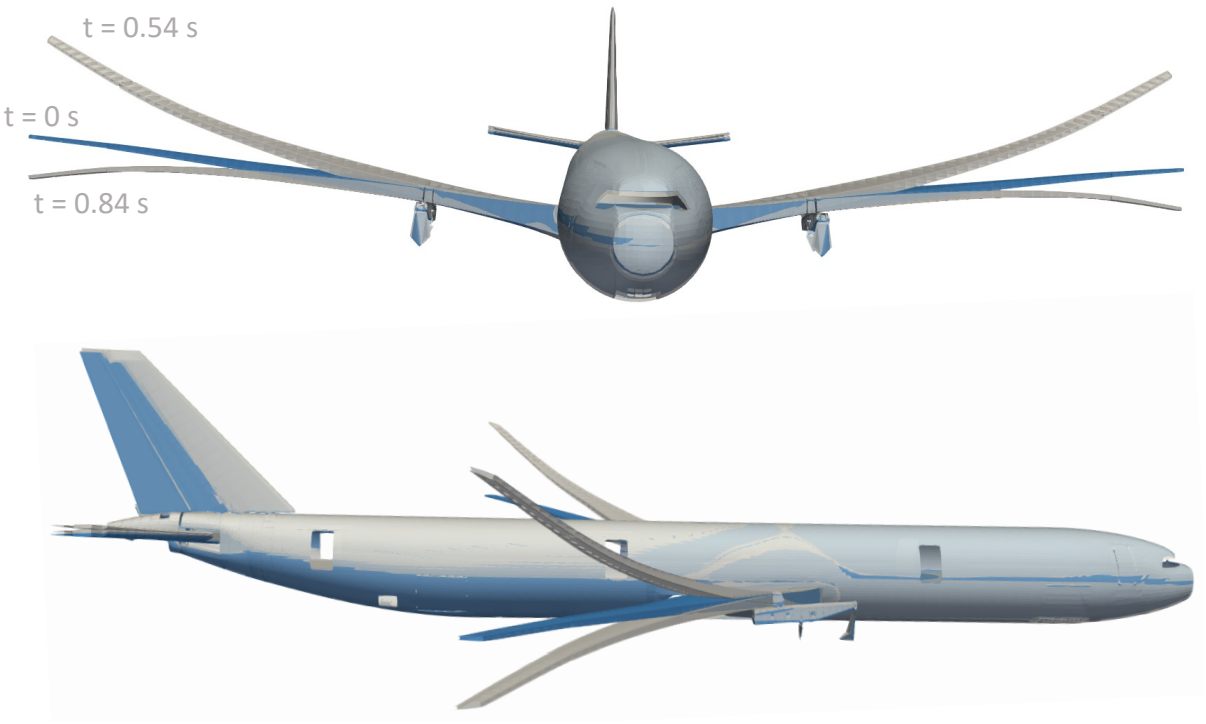


Figure 26: Snapshots of XRF1 Nonlinear gust response

Table 7: AD verification for the response to gust

	w_g	L_g	ρ_{inf}
f_1 (AD)	12.180	-0.690	477.208
f_1 (FD)	12.180	-0.690	477.198
$f_2(AD)$	19.088	-1.015	712.485
$f_2(FD)$	19.088	-1.015	712.514
$f_3(AD)$	65.574	18.764	1464.910
$f_3(FD)$	65.574	18.764	1464.909
$m_1(AD)$	126.648	6.961	2883.370
$m_1(FD)$	126.648	6.961	2883.371
$m_2(AD)$	330.759	84.098	5931.723
$m_2(FD)$	330.759	84.099	5930.027
m_3 (AD)	252.128	24.212	7179.735
m_3 (FD)	252.128	24.211	7180.023

4 CONCLUSIONS

This paper has presented a modal-based description that incorporates geometrically nonlinear effects due to structural slenderness onto generic FE models initially built for linear analysis. This nonlinear aeroelastic framework accounts for follower aerodynamic forces, geometric stiffening and the coupling between elastic and rigid-body DoF. While the underlying theory had already been introduced, a new implementation was put in-place for both high-performance and software modularity, with the numerical library JAX as the engine powering the computations. This has allowed time-domain computations in near real-time with two orders of magnitude

speed-ups compared to conventional implementations. Three major highlights about the new implementation have been introduced in this work: 1) the propagation of derivatives in the solution process via the AD tool embedded in JAX with structural and aeroelastic examples that include static and dynamic responses; 2) show how the program running the computations can be deployed on standard CPUs but also on modern hardware architectures such as GPUs, and demonstration of performance gains with accelerations of over 30 times with respect to the CPU for large problems; 3) the ability to recover the full 3D state from the reduced-order-model was verified against MSC Nastran full nonlinear solution. This completes a differentiated aeroelastic framework that can run very efficiently in modern hardware architectures while enhancing traditional FE models that can be very complex by construction but lack the physics of geometrically nonlinear effects.

A relevant amount of test cases accompany the software, of which a subset has been presented herein to illustrate the aforementioned features in the code. They have have been arranged in order of model complexity. Firstly, a very flexible, unsupported shell structure undergoing coupled rigid and flexible motions in vacuum. Secondly, a representative model of an aircraft without engines and fuselage is employed to verify the 3D structural static and dynamic nonlinear response against MSC Nastran and their corresponding sensitivities against finite-differences. Lastly, the dynamic aeroelastic response to a gust on a full aircraft model built to industry standards is shown together with the derivatives with respect to flow and gust parameters.

As for future work, a strategy to compute manoeuvre and dynamic load envelopes that can also be differentiated via AD will be built. This will be enabled using parallelisation of distributed accelerators such that the thousands of loads cases are computed fast. Increasing the fidelity in the load calculations to consider CFD-based aerodynamics would be an additional necessary step in order to achieve a more accurate nonlinear aeroelastic methodology.

ACKNOWLEDGEMENTS

This work has received funds from Innovate UK, under project 10002372, managed by the UK Aerospace Technology Institute. Support and feedback from Lucian Iorga at AIRBUS is greatly acknowledged.

5 REFERENCES

[1] Livne, E. (2018). Aircraft Active Flutter Suppression: State of the Art and Technology Maturation Needs. *Journal of Aircraft*, 55(1), 410–452. ISSN 0021-8669, 1533-3868. doi:10.2514/1.C034442.

[2] Gray, A. C. and Martins, J. R. (2021). Geometrically Nonlinear High-fidelity Aerostructural Optimization for Highly Flexible Wings. In *AIAA Scitech 2021 Forum*. ISBN 978-1-62410-609-5. doi:10.2514/6.2021-0283.

[3] Artola, M., Goizueta, N., Wynn, A., et al. (2021). Aeroelastic control and estimation with a minimal nonlinear modal description. *AIAA Journal*, 59(7), 2697–2713. ISSN 0001-1452. doi:10.2514/1.j060018.

[4] Cesnik, C. E., Palacios, R., and Reichenbach, E. Y. (2014). Reexamined Structural Design Procedures for Very Flexible Aircraft. *Journal of Aircraft*, 51(5), 1580–1591. ISSN 0021-8669, 1533-3868. doi:10.2514/1.C032464.

[5] Jonsson, E., Riso, C., Monteiro, B. B., et al. (2023). High-Fidelity Gradient-Based Wing Structural Optimization Including Geometrically Nonlinear Flutter Constraint. *AIAA Journal*, 61(7), 3045–3061. ISSN 0001-1452, 1533-385X. doi:10.2514/1.J061575.

[6] Gray, A. C., Riso, C., Jonsson, E., et al. (2023). High-Fidelity Aerostructural Optimization with a Geometrically Nonlinear Flutter Constraint. *AIAA Journal*, 61(6), 2430–2443. ISSN 0001-1452, 1533-385X. doi:10.2514/1.J062127.

[7] Kier, T. M. (2017). An integrated model for lateral gust loads analysis and dutch roll flight dynamics using a 3d panel method. International Forum on Aeroelasticity and Structural Dynamics 2013. ISBN 978-88-97576-28-0. ISSN 18379664. doi:10.7150/jca.24838.

[8] Sanghi, D., Cesnik, C. E. S., and Riso, C. (2024). Roll Maneuverability of Transonic High-Aspect-Ratio-Wing Aircraft with Flared Folding Wingtips. *Journal of Aircraft*, 1–14. ISSN 0021-8669, 1533-3868. doi:10.2514/1.C037470.

[9] Choi, Y., Boncoraglio, G., Anderson, S., et al. (2020). Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, 423, 109787. ISSN 00219991. doi:10.1016/j.jcp.2020.109787.

[10] Swischuk, R., Mainini, L., Peherstorfer, B., et al. (2019). Projection-based model reduction: Formulations for physics-based machine learning. *Computers & Fluids*, 179, 704–717. ISSN 00457930. doi:10.1016/j.compfluid.2018.07.021.

[11] Riso, C. and Cesnik, C. E. S. (2023). Impact of Low-Order Modeling on Aeroelastic Predictions for Very Flexible Wings. *Journal of Aircraft*, 60(3), 662–687. ISSN 0021-8669, 1533-3868. doi:10.2514/1.C036869.

[12] Drachinsky, A. and Raveh, D. E. (2022). Nonlinear Aeroelastic Analysis of Highly Flexible Wings Using the Modal Rotation Method. *AIAA Journal*, 60(5), 3122–3134. ISSN 0001-1452, 1533-385X. doi:10.2514/1.J061065.

[13] Palacios, R. and Cea, A. (2019). Nonlinear Modal Condensation of Large Finite Element Models: Application of Hodges's Intrinsic Theory. *AIAA Journal*, 57(10), 4255–4268. ISSN 0001-1452, 1533-385X. doi:10.2514/1.J057556.

[14] Cea, A. (2021). *A Geometrically Nonlinear Approach for the Aeroelastic Analysis of Commercial Transport Aircraft*. Ph.D. thesis, Imperial College London.

[15] Cea, A. and Palacios, R. (2023). Geometrically Nonlinear Effects on the Aeroelastic Response of a Transport Aircraft Configuration. *Journal of Aircraft*, 60(1), 205–220. ISSN 0021-8669, 1533-3868. doi:10.2514/1.C036740.

[16] Bradbury, J., Frostig, R., Hawkins, P., et al. (2018). JAX: Composable transformations of Python+NumPy programs.

[17] Bezgin, D. A., Buhendwa, A. B., and Adams, N. A. (2023). JAX-Fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *Computer Physics Communications*, 282, 108527. ISSN 00104655. doi:10.1016/j.cpc.2022.108527.

[18] Xue, T., Liao, S., Gan, Z., et al. (2023). JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science. *Computer Physics Communications*, 291, 108802. ISSN 00104655. doi:10.1016/j.cpc.2023.108802.

[19] Lusher, D. J., Jammy, S. P., and Sandham, N. D. (2021). OpenSBLI: Automated code-generation for heterogeneous computing architectures applied to compressible fluid dynamics on structured grids. *Computer Physics Communications*, 267, 108063. ISSN 00104655. doi:10.1016/j.cpc.2021.108063.

[20] Roger, K. L. (1977). Airplane math modeling methods for active control design. *Structural Aspects of Active Controls, AGARD CP-228*, 4.1–4.11.

[21] Palacios, R. and Cea, A. (2023-10-03/2023-10-04). Polynomial preconditioning in unsteady aerodynamic models for aeroelasticity. In *RAeS 8th Aircraft Structural Design Conference*.

[22] Wang, Y., Palacios, R., and Wynn, A. (2015). A method for normal-mode-based model reduction in nonlinear dynamics of slender structures. *Computers and Structures*, 159, 26–40. ISSN 00457949. doi:10.1016/j.compstruc.2015.07.001.

[23] Hodges, D. H. (2003). Geometrically Exact, Intrinsic Theory for Dynamics of Curved and Twisted Anisotropic Beams. *AIAA Journal*, 41(6), 1131–1137. ISSN 0001-1452, 1533-385X. doi:10.2514/2.2054.

[24] Guyan, R. J. (1965). Reduction of stiffness and mass matrices. *AIAA Journal*, 3(2), 380–380. doi:10.2514/3.2874.

[25] Cea, A. and Palacios, R. (2021). A non-intrusive geometrically nonlinear augmentation to generic linear aeroelastic models. *Journal of Fluids and Structures*, 101, 103222. ISSN 08899746. doi:10.1016/j.jfluidstructs.2021.103222.

[26] Palacios, R. and Cesnik, C. E. S. (2023). *Dynamics of Flexible Aircraft: Coupled Flight Mechanics, Aeroelasticity, and Control*. Cambridge Aerospace Series. Cambridge, UK: Cambridge University Press. doi:10.1017/9781108354868.

[27] Palacios, R., Murua, J., and Cook, R. (2010). Structural and aerodynamic models in nonlinear flight dynamics of very flexible aircraft. *AIAA Journal*, 48(11), 2648–2659. ISSN 0001-1452. doi:10.2514/1.J050513.

[28] Cea, A. and Palacios, R. (2024). A Nearly-Real Time Nonlinear Aeroelastic Simulation Architecture Based on JAX. In *AIAA SCITECH 2024 Forum*. Orlando, FL. ISBN 978-1-62410-711-5. doi:10.2514/6.2024-0613.

[29] Simo, J. C. and Vu-Quoc, L. (1988). On the dynamics in space of rods undergoing large motions - A geometrically exact approach. *Computer Methods in Applied Mechanics and Engineering*, 66(2), 125–161. ISSN 00457825. doi:10.1016/0045-7825(88)90073-4.

[30] Hesse, H., Palacios, R., and Murua, J. (2014). Consistent structural linearization in flexible aircraft dynamics with large rigid-body motion. *AIAA Journal*, 52(3), 528–538. ISSN 0001-1452. doi:10.2514/1.J052316.

COPYRIGHT STATEMENT