Near-Real-Time Nonlinear Analysis of Free-Flying Flexible Aircraft on Modern Hardware Architectures

Alvaro Cea* and Rafael Palacios†
Imperial College London, SW7 2AZ, United Kingdom

Lucian Iorga‡
Airbus Operations Ltd., Filton, BS99 7AR, United Kingdom

This paper explores the use of graphical processing units for computationally-efficient aeroelastic dynamic analysis of free-flying vehicles. The numerical library JAX and an intrinsic description of the flexible aircraft dynamics are brought together into a highly vectorized codebase, thus enabling geometrically nonlinear simulations to run close to if not in real-time. The approach seamlessly integrates with conventional aeroelastic load packages to facilitate the analysis of complex aircraft configurations, which opens new possibilities for industrial-scale applications including geometric nonlinearities. In addition, a multi-body formulation is presented that shows the advantages of casting the nonlinear equations with velocities and strains as state variables. Verification studies on a double pendulum are first presented. The computational edge provided by the new implementation is then shown in a free-flying, very flexible structure in vacuum that undergoes strongly coupled rigid body and elastic deformations. Finally, the aeroelastic response of an industrial-scale aircraft configuration is studied. Results include a comparison of the trim equilibrium against linear simulations, a recovery of the full 3D geometry, and the response of the trimmed aircraft under 1-cos gusts. Wing loads are finally evaluated that confirm that linear assumptions may be non-conservative for high-aspect-ratio wing aircraft.

Nomenclature

$\boldsymbol{\alpha}_x$	=	maneuver steady-state input vector
$\boldsymbol{\mathcal{A}}_i$	=	Roger's approximation matrices
c	=	chord
$\boldsymbol{\eta}$	=	external loads in modal coordinates

*Research Associate, Department of Aeronautics, CAGB 308, South Kensington Campus. (alvaro.cea-esteban15@imperial.ac.uk)
†Professor in Computational Aeroelasticity, Department of Aeronautics and Brahmal Vasudevan Institute for Sustainable Aviation, CAGB 310, South Kensington Campus. AIAA Associate Fellow (r.palacios@imperial.ac.uk)
‡Wing Airframe Integrator

$\mathbf{f}_{ext}$ = external forces

$\mathbf{\Gamma}_{1,2}$ = nonlinear coupling coefficients in modal coordinates

$\boldsymbol{\gamma}(s,t)$ = load-paths sectional strains

$\boldsymbol{\kappa}(s,t)$ = load-paths sectional moment strains

$\boldsymbol{K}_a$ = FE stiffness matrix after condensation

L_g = gust length

$\boldsymbol{\lambda}_{12}$ = Lagrange multipliers for multibody system

$\boldsymbol{M}_a$ = Condensed FE mass matrix

$\boldsymbol{\mathcal{M}}$ = apparent mass matrix

N_m = Number of mode-shapes in the solution

N_n = Number of spatial-nodes in the solution

N_t = Number of time steps in the solution

$\mathbf{v}(s,t)$ = load-paths translational velocities in material coordinates

$\boldsymbol{\omega}_x(s,t)$ = load-paths angular velocities in material coordinates

ω = natural angular frequencies

$\mathbf{\Phi}_0(s)$ = natural mode shape j

$\mathbf{\Phi}_1(s)$ = intrinsic velocity modes

$\mathbf{\Phi}_2(s)$ = intrinsic force modes

$\mathbf{\Psi}_1(s)$ = intrinsic momentum modes

$\mathbf{\Psi}_2(s)$ = intrinsic strain modes

$\boldsymbol{q}_0(t)$ = modal coordinates (displacement component)

$\boldsymbol{q}_1(t)$ = intrinsic modal coordinates (velocity component)

$\boldsymbol{q}_2(t)$ = intrinsic modal coordinates (stress resultant component)

$\boldsymbol{q}_c$ = control surface input vector

$\boldsymbol{r}(s,t)$ = load-paths position vector

$\boldsymbol{R}(s,t)$ = load-paths local coordinate transformation matrix

ρ_∞ = free stream air density

s = curvilinear coordinate along main load path

t = time

U_∞ = free stream flow velocity

$\boldsymbol{v}_g(t)$ = gust input vector

$\boldsymbol{x}_a$ = Condensed structure coordinates

$\boldsymbol{x}_0(s,t)$ = Linear displacement vector

$\boldsymbol{x}_1(s,t)$ = local translational/angular velocity state vector

$\boldsymbol{x}_2(s,t)$ = internal force/moment state vector

$\boldsymbol{x}_3(s,t)$ = strains and curvatures state vector

$\boldsymbol{\zeta}(s,t)$ = quaternions parameterizing rotations

I. Introduction

Most countries have made net-zero commitments to mitigate the adverse effects of anthropogenic climate change. They include mandates for the introduction of aviation fuels of non-fossil origin, which are however much more costly to produce that conventional kerosene. The response of the aviation sector is to seek large additional fuel efficiencies in next-generation aircraft that partially compensate for those extra operating costs, even if they also result in overall higher development costs. In particular, this new capital vs. operational trade-off could make economically viable the development of civil transport aircraft with very high aspect ratio wings (i.e., over 50% higher than current designs). They would leverage the latest advancements in manufacturing with lighter materials to give large reductions in induced drag, but they also would result in much higher wing flexibility, which brings the need for more complex aeroelastic analysis methods to support their design. The new modeling approaches would need to capture with large, potentially nonlinear, wing deformations, while maintaining an acceptable computational cost. This was already highlighted in [1], which called for mathematical formulations that include nonlinear effects that not only enhance current modeling techniques but that also allow rapid data turnaround for the industry. Real-time, hardware-in-the-loop flight simulators would also benefit actively controlled, deformable airplane models. This implies a switch in aeroelastic simulation paradigm, where 1) the overall aerodynamic performance is calculated around the instantaneous in-flight shape[2]; 2) control laws accounts for both steady state and nonlinear dynamics [3]; and 3) the loads ultimately sizing the wings are atmospheric disturbances computed in the time-domain [4]. This may also be the case for more radical configurations with non-conventional aeroelastic behavior, even if not large deformations.

Aircraft certification requires the analysis of 100,000s of load cases [5], including steady maneuvers and dynamic loads across the flight envelope and for multiple mass cases. This poses a major challenge in the introduction of new methods with improved physical models, as they can quickly demand extremely large computer resources. For roll maneuvers, for instance, Ref. [6] has recently shown the importance of fully coupled nonlinear aeroelastic-flight dynamics solutions. Several computational strategies have been proposed to address this, typically using projection-based reduced order models[7], sometimes enhanced using machine learning [8]. Lastly, the mathematical representation of the airframe, embodied in 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 [9], or using a modal-based approximation [10].

Those previous considerations set the goals for the present work: 1) to be able to perform geometrically nonlinear aeroelastic analysis of the full vehicle, 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. The first two points have been demonstrated using the formulation developed in [11], [12], and [13], and we tackle the third point herein with a new implementation that we have recently introduced in [14] and for which we will explore here the computational performance on a modern hardware. Moreover, while in [14] the application was restricted to clamped structures, here we also include the flight dynamics and a trim algorithm to initialize the simulations of the transient response to atmospheric gusts.

The recently developed numerical library JAX [15] provides the computational environment for our implementation. JAX-based software can be deployed on CPUs, GPUs and TPUs (Central, Graphical and Tensor Processing Units), and features powerful parallelization capabilities across devices. In our flexible aircraft dynamics simulator, `FENIAX` (Finite-Element models for Intrinsic Aeroelastics in JAX) [14], models are built using standard aeroelastic descriptions, namely a linear (arbitrarily complex) FE model, and frequency-dependent unsteady aerodynamic influence coefficient matrices in either modal or in physical coordinates [16]. The latter are obtained here using the Doublet Lattice Method (DLM), followed by a rational function approximation (RFA) [17] 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 [18] and which is suitable for Computational Fluid Dynamics (CFD). Condensation of the 3D FE model on nodes along the main load paths is used to identify a modal description of the aircraft dynamics in intrinsic variables (velocities and strains) [19]. This gives a direct reduction process from the full-aircraft model that is already available in many industrial aeroelastic frameworks. This is unlike stick models which need of various post-processing steps to build the equivalent stiffness and mass models. Furthermore, in this paper 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. The result is an extremely efficient nonlinear aeroelastic solution approach that will be referred to as *Nonlinear Modal Reduced Order Model* (NMROM).

In what follows, section II will outline the main formulation behind `FENIAX`, the high performance software that we have recently introduced in [14]. In particular, the mathematical formulation of the NMROM is extended here to include general flexible multibody systems. This will be followed by several numerical examples in section III. First, we will verify the newly developed multibody formulation to investigate the dynamics of a double pendulum. Second, our coupled rigid-body/flexible solution in JAX will be also verified in on a very flexible, free-flying structure. Finally, the new trim solution will be exercised on a large aircraft model, followed by studies on gust response, for which we will investigate the reduction of computational effort obtained by exploiting modern hardware. An important novelty aspect

of this work is the exploration of the computational performance of the NMROM simulation strategy on graphical processing units (GPUs) to assess their exploitation for industrial-scale usage of the proposed approach. Lastly, section IV will summarize the main outcomes of this work.

II. Theory and implementation

The nonlinear aeroelastic model used in this work were first introduced in Refs. [12, 13] and are briefly outlined next. Moreover, the structural description is extended herein with a new multibody formulation. This will be followed by some details of implementation that are of particular relevance to our discussion.

A. Geometrically nonlinear description with rigid-body dynamics

In the construction of our NMROM, the deformed state of a structural model is approximated by a deformable curve in 3-D space, s, – identified with the aircraft major load-paths – whose dynamics are described by a system of nonlinear equations written in terms of inertial velocities, $\mathbf{x}_1(s,t)$, and stresses, $\mathbf{x}_2(s,t)$, both written in their components in the local material frame [16, Ch. 8], as

$$\begin{aligned} \mathcal{M}\dot{\mathbf{x}}_1 - \mathbf{x}_2' - \mathbf{E}\boldsymbol{x}_2 + \mathcal{L}_1(\mathbf{x}_1)\mathcal{M}\mathbf{x}_1 + \mathcal{L}_2(\mathbf{x}_2)C\mathbf{x}_2 &= \mathbf{f}_{ext} \\ C\dot{\mathbf{x}}_2 - \mathbf{x}_1' + \mathbf{E}^\top\mathbf{x}_1 - \mathcal{L}_1^\top(\mathbf{x}_1)C\mathbf{x}_2 &= \mathbf{0} \end{aligned} \tag{1}$$

The sectional properties given in the mass $\mathcal{M}$ and compliance C matrices are typically not available when the starting point for the analysis is a Global Finite-Element Model (GFEM) of the airframe. They will be indirectly evaluated from a reduced model obtained from a static condensation, or Guyan reduction [20], of the GFEM. This captures well the stiffness and inertia properties of the aircraft in the condensed matrices $\boldsymbol{K}_a$ and $\boldsymbol{M}_a$, respectively, when the mass model features lumped masses along the main load paths that are attached to the local structure via interpolation elements, as is common practice for large-scale aeroelastic models. However, if the mass is given by a generic mass model the method is also valid and dynamic condensation can be employed for additional accuracy, as we have previously shown in [21]. The eigenvalue solution of the FEM yields the modal shapes, $\boldsymbol{\Phi}_{0j}$, and their associated frequencies ω_j, with $j = 1, \ldots, N_m$ and N_m the size of the condensed matrices (although some truncation is often introduced). A modal expansion of the state variables is a key step in seamlessly mapping the GFEM into the nonlinear description. Writing the vibration modes in those state variables results in velocity and internal force/moment fields approximated as $\mathbf{x}_1 = \boldsymbol{\Phi}_1\mathbf{q}_1 = \boldsymbol{\Phi}_{1j}\mathbf{q}_{1j}$ and $\mathbf{x}_2 = \boldsymbol{\Phi}_2\mathbf{q}_2 = \boldsymbol{\Phi}_{2j}\mathbf{q}_{2j}$, respectively, using Einstein's summation convention. More specifically, we can define four representations of the mode shapes on the condensed model:

1) Velocity modes, $\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_0$, obtained from the time derivatives of the displacements, $\mathbf{x}_0$, namely, $\mathbf{x}_1 = \dot{\mathbf{x}}_0$, and thus $\boldsymbol{\Phi}_1\mathbf{q}_1 = \boldsymbol{\Phi}_0\dot{\mathbf{q}}_0$.
2) Momentum modes, defined as $\boldsymbol{\Psi}_1 = \boldsymbol{M}_a\boldsymbol{\Phi}_0$.

3) Force/moment modes, $\mathbf{\Phi}_2 = \mathcal{S}(\boldsymbol{K}_a\mathbf{\Phi}_0)$, which correspond to the sectional stress resultants associated to the modal deformation. The operator $\mathcal{S}$ indicates integration of forces and moments along load paths [21], with modes computed in the mid-point between the condensation nodes in the GFEM to avoid discontinuities.
4) Strain modes, $\mathbf{\Psi}_2(s_i) = -\frac{\mathbf{\Phi}_0(s_{i+1})-\mathbf{\Phi}_0(s_i)}{\Delta s_i} + \mathsf{E}^\top \frac{\mathbf{\Phi}_0(s_{i+1})+\mathbf{\Phi}_0(s_i)}{2}$, with E a constant matrix as defined in [11].

Details of their computational implementation in `FENIAX` can be found in Algorithm 1 below. Using the computed modal shapes, a dynamic system is obtained after a Galerkin projection of the geometrically-exact beam equations of motion [18, Ch. 8]. Using again indicial notation, it can be written as

$$\begin{aligned}\dot{q}_{1i} &= \delta_{ij}\omega_i q_{2j} - q_{1k}\Gamma^k_{1ij}q_{1j} - q_{2k}\Gamma^k_{2ij}q_{2j} + \eta_j \\ \dot{q}_{2i} &= -\delta_{ij}\omega_i q_{1j} + q_{2k}\Gamma^k_{2ji}q_{1j}\end{aligned} \tag{2}$$

In Ref. [14] we have written the expressions above using tensor notation, which is in fact the way they have been implemented and vectorized, but this is not necessary for the discussion here. The description in (2) is geometrically nonlinear with the modal couplings given by the third-order tensors $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ and with $\boldsymbol{\eta}$ the modal projection of the external forces. They are defined as

$$\begin{aligned}\Gamma^k_{1ij} &= \langle \mathbf{\Phi}_{1i}, \mathcal{L}_1(\mathbf{\Phi}_{1j})\mathbf{\Psi}_{1k}\rangle, \\ \Gamma^k_{2ij} &= \langle \mathbf{\Phi}_{1i}, \mathcal{L}_2(\mathbf{\Phi}_{2j})\mathbf{\Psi}_{2k}\rangle, \\ \eta_i &= \langle \mathbf{\Phi}_{1i}, \mathbf{f}_{ext}\rangle\end{aligned} \tag{3}$$

with $\mathcal{L}_1$ and $\mathcal{L}_2$ linear operators defined as $\mathcal{L}_1([\boldsymbol{u},\boldsymbol{v}]) = [\tilde{\boldsymbol{v}}\ \mathbf{0}; \tilde{\boldsymbol{u}}\ \tilde{\boldsymbol{v}}]$ and $\mathcal{L}_2([\boldsymbol{u},\boldsymbol{v}]) = [\mathbf{0}\ \tilde{\boldsymbol{u}}; \tilde{\boldsymbol{u}}\ \tilde{\boldsymbol{v}}]$, and $\tilde{\bullet}$ is the skew-symmetric operator[19]; the inner product over the whole load path $\langle \boldsymbol{u},\boldsymbol{v}\rangle = \int_s \boldsymbol{u}^\top \boldsymbol{v} ds$, for any $\boldsymbol{u} \in \mathbb{R}^6$ and $\boldsymbol{v} \in \mathbb{R}^6$. The forcing terms can include gravity, $\boldsymbol{\eta}_g$, propulsion and aerodynamic forces, $\boldsymbol{\eta}_a$, including the effect of gust disturbances, $\boldsymbol{v}_g$. Control states can also be included [21], but they are not necessary for this work.

Calculation of the deformed geometry is obtained in a postprocessing step. Firstly, we define an Earth global reference system a used to determine the trajectory and instantaneous orientation of the aircraft, and then a local material frame b at each point along the load path. The relative position of the origin of S is $\mathbf{r}_a$, where the subindex implies that it is written on the global frame, and the the relative orientation is defined by the coordinate transformation matrix, $\mathbf{R}_{ab}$. Local rotations are needed, however, within the solution process when gravity, or other dead forces, are active. Recall that in an intrinsic formulation, forces are naturally given in the material frame of reference and so gravity forces need to be expressed in a local deformed frame. Quaternions $\boldsymbol{\zeta} = [\zeta_0, \boldsymbol{\zeta}_x]$ are used to parameterize the rotation, $\mathbf{R}_{ab}$, such that

given the angular velocity, $\boldsymbol{\omega}_x$, which is part of the velocity state, $\mathbf{x}_1$, it is

$$\dot{\boldsymbol{\zeta}} = \begin{bmatrix} \dot{\zeta}_0 \\ \dot{\boldsymbol{\zeta}}_x \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{\omega}_x^\top \boldsymbol{\zeta}_x \\ \frac{1}{2}(\zeta_0 \boldsymbol{\omega}_x - \tilde{\boldsymbol{\omega}}_x \boldsymbol{\zeta}_x) \end{bmatrix} \tag{4}$$

The quaternions, one per node, can be added to Eqs. (2) and march in time; the rotations can be extracted at every step as $\mathbf{R}_{ab} = \left(2\boldsymbol{\zeta}_x \otimes \boldsymbol{\zeta}_x + (\zeta_0^2 - \boldsymbol{\zeta}_x \cdot \boldsymbol{\zeta}_x)\boldsymbol{I}_3\right) + 2\zeta_0 \tilde{\boldsymbol{\zeta}}_x$, where the first parenthesis in 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 $\boldsymbol{\gamma}$ and curvatures $\boldsymbol{\kappa}$, which can be obtained from the strain modes defined above, we have

$$\begin{aligned} \mathbf{R}'_{ab} &= \mathbf{R}_{ab}\tilde{\boldsymbol{\kappa}} \\ \mathbf{r}'_a &= \mathbf{R}_{ab}(\boldsymbol{\gamma} + \boldsymbol{e}_1) \end{aligned} \tag{5}$$

where $\boldsymbol{e}_1 = [1, 0, 0]$. Analytical solutions to Eq. (5) are obtained when the strain is assumed constant between nodes, using a piecewise constant integration. 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

$$\begin{aligned} \mathbf{R}_{ab}(s) &= \mathbf{R}_{ab}(s_{n-1})\boldsymbol{\mathcal{H}}^0(\boldsymbol{\kappa}, s) \\ \mathbf{r}_a(s) &= \mathbf{r}_a(s_{n-1}) + \boldsymbol{R}_{ab}(s_{n-1})\boldsymbol{\mathcal{H}}^1(\boldsymbol{\kappa}, s)\,(\boldsymbol{e}_1 + \boldsymbol{\gamma}_n) \end{aligned} \tag{6}$$

with the operators $\boldsymbol{\mathcal{H}}^0(\boldsymbol{\kappa}, s)$ and $\boldsymbol{\mathcal{H}}^1(\boldsymbol{\kappa}, s)$ obtained from integration of the exponential function, as in [22]. 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.

Finally, 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.

B. General multi-body formulation in intrinsic modal variables

In this section we show how the presented formulation can also accommodate multibody constraints. These are imposed at the velocity level using the state variable $\mathbf{x}_1 = [\mathbf{v}, \boldsymbol{\omega}_x]$, which leads to highly compact and simple expressions, as shown next. The GFEM is split in multiple bodies, $B^{(\mathfrak{i})}$, although we restrict the description for two bodies in what follows. Each body is considered in isolation and the linear normal modes are obtained with no supports.

A reformulation of some variables is needed to denote the connecting nodes of the different bodies. The components of the intrinsic velocity modes, $\boldsymbol{\Phi}_1$, at the body $B^{(\mathfrak{i})}$ and the connecting node j are written as $\boldsymbol{\Phi}_1^{(\mathfrak{i})}(s_j) \in \mathbb{R}^{6\times N_{m_i}}$, with N_{m_i} the number of modes used in the solution of body $\mathfrak{i}$. The corresponding modal velocity coordinates are $\mathbf{q}_1^{(\mathfrak{i})}$. Therefore constraints will be imposed on translational local velocities, $\mathbf{v}^{(\mathfrak{i})}(s_j) = \boldsymbol{\Phi}_{1u}^{(\mathfrak{i})}(s_j)\mathbf{q}_1^{(\mathfrak{i})}$, their projection on the body-fixed reference frame, $\mathbf{v}_a^{(\mathfrak{i})} = \mathbf{R}_{ab}^{(\mathfrak{i})}\mathbf{v}^{(\mathfrak{i})}$, and the local angular velocities, $\boldsymbol{\omega}_x^{(1)} = \boldsymbol{\Phi}_{1\theta}^{(1)}\mathbf{q}_1^{(1)}$: Note how the angular velocity in the global reference frame, $\boldsymbol{\omega}_a$ is not needed to impose the constraints, as velocities due to rotations in the points of solid bodies are better described from the local reference. Whenever constraints are given in the global reference frame, the rotation matrix at each connecting node, $\mathbf{R}_{ab}^{(\mathfrak{i})}$, is needed. Quaternions, $\boldsymbol{\zeta}^{(\mathfrak{i})}$ are employed to parametrize rotations, Eqs. (4), and they will be part of the DAE system for the nodes at each joint. In order to simplify the formulation we restrict the description to two bodies, $B^{(1)}$ and $B^{(2)}$. Up to three algebraic equations, $\boldsymbol{g}^v$ are added to the intrinsic equations to restrain the movement of the 2 connecting points in their translational DoF. Similarly, for the general multibody case, rotations need to be constrained with up to three algebraic equations, $\boldsymbol{g}^\omega$. The Lagrange multipliers take the form, $\boldsymbol{\lambda}_{12} = [\boldsymbol{\lambda}_{12}^{v\top}\ \boldsymbol{\lambda}_{12}^{\omega\top}]^\top$. The modal velocity component of Eqs. (2) is written using the right-hand side functional $\boldsymbol{F}()$ for the $B^{(\mathfrak{i})}$ body as $\dot{\mathbf{q}}_1^{(\mathfrak{i})} = \boldsymbol{F}_{1\mathfrak{i}}(\mathbf{q}_1^{(\mathfrak{i})}, \mathbf{q}_2^{(\mathfrak{i})})$. And similarly for the quaternions, $\dot{\boldsymbol{\zeta}}^{(\mathfrak{i})} = \boldsymbol{F}_{\zeta\mathfrak{i}}(\mathbf{q}_1^{(\mathfrak{i})}, \boldsymbol{\zeta}^{(\mathfrak{i})})$. The resulting DAE system for two bodies is,

$$
\begin{aligned}
\dot{\mathbf{q}}_1^{(1)} &= \boldsymbol{F}_{11}(\mathbf{q}_1^{(1)}, \mathbf{q}_2^{(1)}) - \boldsymbol{G}_{1q}^{v\top}\boldsymbol{\lambda}_{12}^{v} - \boldsymbol{G}_{1q}^{\omega\top}\boldsymbol{\lambda}_{12}^{\omega} \\
\dot{\mathbf{q}}_1^{(2)} &= \boldsymbol{F}_{12}(\mathbf{q}_1^{(2)}, \mathbf{q}_2^{(2)}) - \boldsymbol{G}_{2q}^{v\top}\boldsymbol{\lambda}_{12}^{v} - \boldsymbol{G}_{2q}^{\omega\top}\boldsymbol{\lambda}_{12}^{\omega} \\
\dot{\mathbf{q}}_2^{(1)} &= \boldsymbol{F}_{21}(\mathbf{q}_1^{(1)}, \mathbf{q}_2^{(1)}) \\
\dot{\mathbf{q}}_2^{(2)} &= \boldsymbol{F}_{22}(\mathbf{q}_1^{(2)}, \mathbf{q}_2^{(2)}) \\
\dot{\boldsymbol{\zeta}}^{(1)} &= \boldsymbol{F}_{\zeta 1}(\mathbf{q}_1^{(1)}, \boldsymbol{\zeta}^{(1)}) - \boldsymbol{G}_{1\zeta}^{v\top}\boldsymbol{\lambda}^{v} - \boldsymbol{G}_{1\zeta}^{\omega\top}\boldsymbol{\lambda}_{12}^{\omega} \\
\dot{\boldsymbol{\zeta}}^{(2)} &= \boldsymbol{F}_{\zeta 2}(\mathbf{q}_1^{(2)}, \boldsymbol{\zeta}^{(2)}) - \boldsymbol{G}_{2\zeta}^{v\top}\boldsymbol{\lambda}^{v} - \boldsymbol{G}_{2\zeta}^{\omega\top}\boldsymbol{\lambda}_{12}^{\omega}
\end{aligned}
\tag{7}
$$

The $\boldsymbol{G}^v$ matrices are the Jacobians of the constraint equations $\boldsymbol{g}^v$, with respect to the modal velocity of the connecting node in the first body, $\boldsymbol{G}_{1q}^v = \frac{\partial \boldsymbol{g}^v}{\partial \mathbf{q}_1^{(1)}} \in \mathbb{R}^{3\times N_{m_1}}$, in the second body, $\boldsymbol{G}_{2q}^v = \frac{\partial \boldsymbol{g}^v}{\partial \mathbf{q}_1^{(2)}} \in \mathbb{R}^{3\times N_{m_2}}$, and with respect to the quaternion variables defining the rotations of the same nodes, $\boldsymbol{G}_{1\zeta}^v = \frac{\partial \boldsymbol{g}^v}{\partial \boldsymbol{\zeta}^{(1)}} \in \mathbb{R}^{3\times 4}$ and $\boldsymbol{G}_{2\zeta}^v = \frac{\partial \boldsymbol{g}^v}{\partial \boldsymbol{\zeta}^{(2)}} \in \mathbb{R}^{3\times 4}$ respectively. In the same way the $\boldsymbol{G}^\omega$ matrices are the Jacobians of the constraint equations $\boldsymbol{g}^\omega$. The Lagrange multipliers $\boldsymbol{\lambda}_{12}$

associated to the constraints are calculated by differentiating the constraints with respect to time:

$$\begin{aligned}\dot{\boldsymbol{g}}^v &= \boldsymbol{G}^v_{1q}\dot{\mathbf{q}}^{(1)}_1 + \boldsymbol{G}^v_{2q}\dot{\mathbf{q}}^{(2)}_1 + \boldsymbol{G}^v_{1\zeta}\dot{\boldsymbol{\zeta}}^{(1)} + \boldsymbol{G}^v_{2\zeta}\dot{\boldsymbol{\zeta}}^{(2)} + \partial_t \boldsymbol{g}^v \\ \dot{\boldsymbol{g}}^\omega &= \boldsymbol{G}^\omega_{1q}\dot{\mathbf{q}}^{(1)}_1 + \boldsymbol{G}^\omega_{2q}\dot{\mathbf{q}}^{(2)}_1 + \boldsymbol{G}^\omega_{1\zeta}\dot{\boldsymbol{\zeta}}^{(1)} + \boldsymbol{G}^\omega_{2\zeta}\dot{\boldsymbol{\zeta}}^{(2)} + \partial_t \boldsymbol{g}^\omega\end{aligned} \tag{8}$$

where $\partial_t \boldsymbol{g}^v$ and $\partial_t \boldsymbol{g}^\omega$ are the partial time derivatives of the constraints if they are of the rheonomic type. Since $\dot{\boldsymbol{g}}^v = \dot{\boldsymbol{g}}^\omega = \mathbf{0}$, Eq. (8) can be arranged in matrix form as follows,

$$\begin{bmatrix} \boldsymbol{G}^v_{1q} & \boldsymbol{G}^v_{1\zeta} \\ \boldsymbol{G}^\omega_{1q} & \boldsymbol{G}^\omega_{1\zeta} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^{(1)}_1 \\ \dot{\boldsymbol{\zeta}}^{(1)} \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}^v_{2q} & \boldsymbol{G}^v_{2\zeta} \\ \boldsymbol{G}^\omega_{2q} & \boldsymbol{G}^\omega_{2\zeta} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^{(2)}_1 \\ \dot{\boldsymbol{\zeta}}^{(2)} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\partial}_t \boldsymbol{g}^v \\ \boldsymbol{\partial}_t \boldsymbol{g}^\omega \end{bmatrix} = \mathbf{0} \tag{9}$$

Combining this expression with Eqs. (7) gives the equations for the Lagrange multipliers for the general case as,

$$\begin{aligned}&\begin{bmatrix} \boldsymbol{G}^v_{1q} & \boldsymbol{G}^v_{1\zeta} \\ \boldsymbol{G}^\omega_{1q} & \boldsymbol{G}^\omega_{1\zeta} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}_{11} \\ \boldsymbol{F}_{\zeta 1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}^v_{2q} & \boldsymbol{G}^v_{2\zeta} \\ \boldsymbol{G}^\omega_{2q} & \boldsymbol{G}^\omega_{2\zeta} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}_{12} \\ \boldsymbol{F}_{\zeta 2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\partial}_t \boldsymbol{g}^v \\ \boldsymbol{\partial}_t \boldsymbol{g}^\omega \end{bmatrix} - \\ &- \left(\begin{bmatrix} \boldsymbol{G}^v_{1q} & \boldsymbol{G}^v_{1\zeta} \\ \boldsymbol{G}^\omega_{1q} & \boldsymbol{G}^\omega_{1\zeta} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}^{v\top}_{1q} & \boldsymbol{G}^{\omega\top}_{1q} \\ \boldsymbol{G}^{v\top}_{\zeta 1} & \boldsymbol{G}^{\omega\top}_{1\zeta} \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}^v_{2q} & \boldsymbol{G}^v_{2\zeta} \\ \boldsymbol{G}^\omega_{2q} & \boldsymbol{G}^\omega_{2\zeta} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}^{v\top}_{2q} & \boldsymbol{G}^{\omega\top}_{2q} \\ \boldsymbol{G}^{v\top}_{2\zeta} & \boldsymbol{G}^{\omega\top}_{2\zeta} \end{bmatrix} \right) \boldsymbol{\lambda}_{12} = \mathbf{0}\end{aligned} \tag{10}$$

Defining $\boldsymbol{G}_1 = \begin{bmatrix} \boldsymbol{G}^v_{1q} & \boldsymbol{G}^v_{1\zeta} \\ \boldsymbol{G}^\omega_{1q} & \boldsymbol{G}^\omega_{1\zeta} \end{bmatrix}$ and $\boldsymbol{G}_2 = \begin{bmatrix} \boldsymbol{G}^v_{2q} & \boldsymbol{G}^v_{2\zeta} \\ \boldsymbol{G}^\omega_{2q} & \boldsymbol{G}^\omega_{2\zeta} \end{bmatrix}$, Lagrange multipliers are written as

$$\boldsymbol{\lambda}_{12} = \left(\boldsymbol{G}_1 \boldsymbol{G}_1^\top + \boldsymbol{G}_2 \boldsymbol{G}_2^\top\right)^{-1} \left(\boldsymbol{G}_1 \begin{bmatrix} \boldsymbol{F}_{11} \\ \boldsymbol{F}_{\zeta 1} \end{bmatrix} + \boldsymbol{G}_2 \begin{bmatrix} \boldsymbol{F}_{12} \\ \boldsymbol{F}_{\zeta 2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\partial}_t \boldsymbol{g}^v \\ \boldsymbol{\partial}_t \boldsymbol{g}^\omega \end{bmatrix} \right) \tag{11}$$

In relation with the inversion of the first parenthesis we note that the matrix inside is positive semi-definite as it is formed from a sum of matrices multiplied by its transpose. Therefore ensuring that each $\boldsymbol{G}_{(\mathrm{i})}$ is invertible makes the overall expression determinant larger than zero, which might require removal of linearly dependent rows and columns on these matrices as shown below in the specific examples of Appendix B. The process to time-march the multi-body system is to first compute all the $\boldsymbol{F}_{(\bullet)}$ terms which depend on the current state variables, $\mathbf{q}^{(\mathrm{i})}_1$ and $\mathbf{q}^{(\mathrm{i})}_2$; afterwards the problem-specific constraint Jacobians, $\boldsymbol{G}_{(\mathrm{i})}$, are obtained from the current state variables; then $\boldsymbol{\lambda}_{12}$ can be calculated from Eqs. (11); finally all the terms on the right-hand side of Eqs. (7) are computed to advance the ODE system to the next time step. Note that because generalized coordinates represent velocities instead of displacements, the differential index of the resulting equations is two (one derivative of $\boldsymbol{g}^v$ and $\boldsymbol{g}^\omega$, and substitution of $\boldsymbol{\lambda}_{12}$ to convert the DAE to ODE),

as opposed to classical formulations of flexible bodies which exhibit an index of three.

C. Aeroelastic system and aircraft trim flight

The structural description described above is coupled with an aerodynamic formulation to build the aeroelastic system of the GFEM. We describe the coupling for a single body for simplicity. For a set of reduced frequencies and a given Mach number, the DLM (or a higher fidelity aerodynamic method) yields the Generalized Aerodynamic Forces (GAFs). The current implementation uses Roger's rational function approximation to those GAFs [17], which results in the follower modal forces:

$$\begin{aligned}\boldsymbol{\eta}_a = \tfrac{1}{2}\rho_\infty U_\infty^2 \Bigg(& \boldsymbol{\mathcal{A}}_0\mathbf{q}_0 + \frac{c}{2U_\infty}\boldsymbol{\mathcal{A}}_1\mathbf{q}_1 + \left(\frac{c}{2U_\infty}\right)^2 \boldsymbol{\mathcal{A}}_2\dot{\mathbf{q}}_1 \\ & + \boldsymbol{\mathcal{A}}_{g0}\boldsymbol{v}_g + \frac{c}{2U_\infty}\boldsymbol{\mathcal{A}}_{g1}\dot{\boldsymbol{v}}_g + \left(\frac{c}{2U_\infty}\right)^2 \boldsymbol{\mathcal{A}}_{g2}\ddot{\boldsymbol{v}}_g + \sum_{p=1}^{N_p}\boldsymbol{\lambda}_p \Bigg)\end{aligned} \tag{12}$$

where $\boldsymbol{\mathcal{A}}_i$ are real matrices of aerodynamic influence coefficients, c is the reference chord, $\frac{1}{2}\rho_\infty U_\infty^2$ is the reference dynamic pressure, $\boldsymbol{\lambda}_p$ the aerodynamic states and N_p the number of lags. $\boldsymbol{v}_g$ represent external disturbances such as gust. Coupling of the structure and aerodynamic equations, rewritten in time-domain using aerodynamic lags gives the final ODE system [14]. A very important part in the calculation of dynamic loads is including the rigid body DoF in the analysis and thus having the aircraft in a state of equilibrium between lift, gravity and control devices forces. This trim condition implies a solution to the nonlinear system of algebraic equations formed by setting velocities and accelerations to zero in (2), namely $\mathbf{q}_1 = \dot{\mathbf{q}}_1 = \mathbf{0}$.

The modal component corresponding to the internal forces, $\mathbf{q}_2$, has a length equal to the total number of modes selected in the solution. However the entries corresponding to the rigid-body modes are zero as those do not carry internal forces or strains, similar to the natural frequencies vector, $\boldsymbol{\omega}$. Instead, the non-zero unknowns corresponding to the rigid-body states come into play in the vector $\hat{\mathbf{q}}_0$. The system of equations is

$$\begin{aligned} & F_i(\mathbf{q}_2, \hat{\mathbf{q}}_0, \mathbf{q}_c) = \delta_{ij}\omega_i q_{2j} - q_{2k}\Gamma^k_{2ij}q_{2j} + \tfrac{1}{2}\rho_\infty U_\infty^2 \left(\mathcal{A}_{0ij}q_{0j} + \hat{\mathcal{A}}_{0ij}\hat{q}_{0j} + \mathcal{A}_{cij}q_{cij}\right) + \eta_{gi} = 0 \\ & \boldsymbol{\mathcal{B}}_r\hat{\mathbf{q}}_0 + \boldsymbol{\mathcal{B}}_0\mathbf{q}_0 = \mathbf{0} \\ & \qquad q_{0i} = q_{2(i+r)}/\omega_{(i+r)} \end{aligned} \tag{13}$$

with $\mathbf{q}_2, \hat{\mathbf{q}}_0, \mathbf{q}_c$ the elastic, rigid-body perturbations and control states. The fist equation in (13), which describes the equilibrium between rigid-body, elastic DoF and external forces, includes n_c unknowns corresponding to the control states more than the number of equations. To make the static problem well posed, the second equation in Eq. (13)

prescribes a 0-displacement point in the structure via the combination of elastic displacements in $\mathbf{q}_0$ and the rigid-body components $\hat{\mathbf{q}}_0$. The last equation defines the linear relation between strain and the displacement modes seen by the aerodynamics, with the constant r corresponding to rigid-body DoF that are filter-out from $\mathbf{q}_2$ and $\boldsymbol{\omega}$ as they are just 0. This is, $\mathbf{q}_0$ represents only elastic displacements. A straight forward example is a trim where only two rigid-body DoF are allowed, the pitch and the plunge, which are determined by a single tail elevator deflection. The added constraining equation is then set for the z-component of the initial node to be zero, thus $\boldsymbol{\mathcal{B}}_r$ and $\boldsymbol{\mathcal{B}}_0$ become just the rigid-body and elastic vectors corresponding to z-component of the modal shapes, $\boldsymbol{\Phi}_0$, at the first node. A somewhat more advanced trim would also allow a forward direction of flight and include thrust as another control state variable, with the added equation to constrain the forward displacement. However for this a good estimate of the aerodynamic drag forces would be required, which is not the case for the DLM aerodynamics so we choose the first option as a suitable strategy for loads. This is equivalent to the model being placed in a wind tunnel on a support that is free to pitch and plunge.

D. Computational implementation

This work uses a new computational implementation of the theory above, recently introduced in [14], that achieves accelerations of over 2 orders of magnitude with respect to its predecessor*. For that purpose, a highly modular, flexible architecture based on software design patterns has been put in place, which results in the nonlinear aeroelastic framework that is suitable for modern hardware architectures. The key enabler was moving from standard Python to a highly vectorized JAX-based numerical implementation. JAX is a Python library designed for high-performance numerical computing with focus on machine learning activities [15]. It combines XLA (accelerated linear algebra) and Autograd, the former being a compiler that optimizes 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 modern Computational Science. These include the `vmap` function which allows for complex vectorization operations and the `shmap` 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. 1 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, $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{N_m \times 6 \times N_n}$ are computed from the eigenvalue solution and the coordinates $\mathbf{x}_a \in \mathbb{R}^{3 \times N_n}$ of the active nodes. The nonlinear couplings,

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

$\mathbf{\Gamma} \in \mathbb{R}^{N_m \times N_m \times N_m}$ are calculated next, from which the system in Eq. 2 is assembled and solved to yield the solution states $\mathbf{q} \in \mathbb{R}^{N_t \times N_s}$ with N_s the number of states in the system that is proportional to the number of modes. Local velocities, internal forces and strain fields $\mathbf{x}_{1,2,3} \in \mathbb{R}^{N_t \times 6 \times N_n}$ 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.

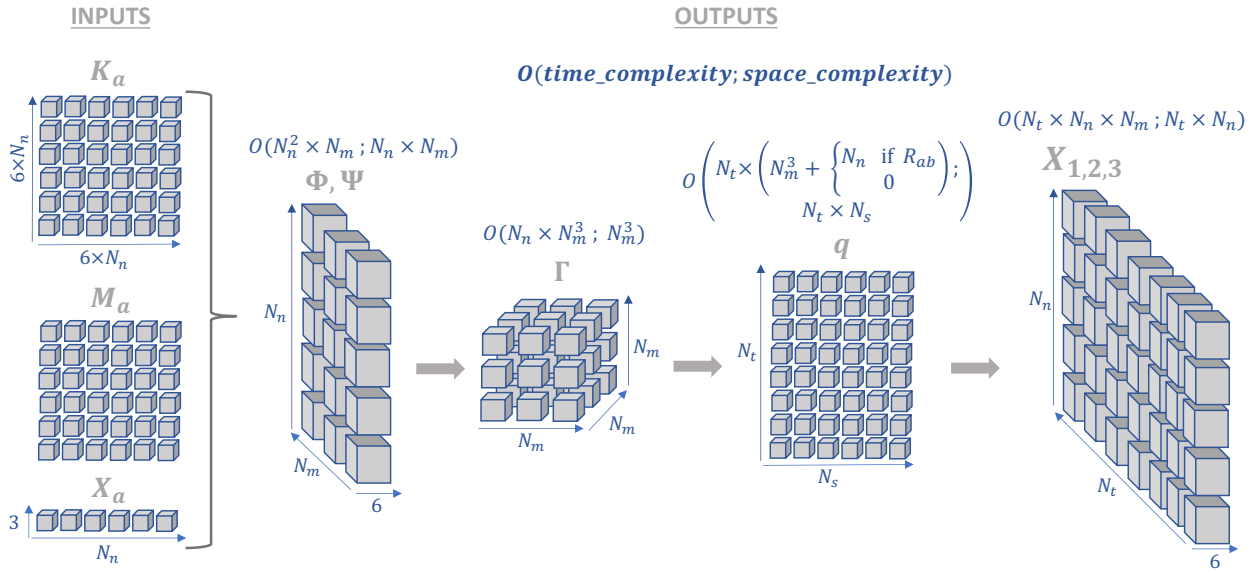


Fig. 1 Main data components in the sequential solution process together with their associated time and space complexities

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 the JAX `vmap` function to vectorize 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, $O(N_n^2)$ 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.

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 to compute the nonlinear couplings in 3 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 constrain the implementation to comply with a nonconventional functional paradigm. Some of these characteristics are pure functions, i.e. functions that have no side effects, as well as input/output stream

Algorithm 1: JAX-based computation of intrinsic modal shapes

input : FEM and eigenvalue solution: $\mathbf{x}_a, \boldsymbol{K}_a, \boldsymbol{M}_a, \boldsymbol{\Phi}_0, \boldsymbol{\omega}$, 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;
```

management that needs to be placed outside the numerical algorithms and immutability of arrays. These very constraints allow the capabilities described above to be achieved 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. 6 is a good example of a recursive operation that requires more than vectorized operations to solve, as opposed to the previous algorithms. The function `jax.lax.scan`, which can be differentiated

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)
```

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 such components make up the whole aircraft.

III. Results

Three different cases are set up to validate and showcase the potential of the proposed methods: a rigid double-pendulum falling under gravity, a very flexible structure undergoing large displacements coupled with rigid-body dynamics, and a full aircraft model whose trimmed nonlinear equilibrium is first computed, to then perturb it with a gust.

A. Double rigid pendulum

A spherical joint is utilized to combine two rigid, 2-noded beams of length 60 and 20 respectively, and with a discrete mass of 80 at the tip of each one of them. Note that this dynamical system can become chaotic, a situation after which comparisons are not reliable. In fact, computations have been performed over a period of 30 seconds, when chaotic dynamics begin to appear. The system is released to fall under the force of gravity acting in the y-direction. It can be posed as an analytical set of nonlinear equations that are time marched to obtain the solution, as shown in the

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)
```

Appendix of [23]. Moreover, a Nastran constrained model is built using MPCs entries and solved with the nonlinear solver in the Solution 400. The NMROM is constructed using only the linear rigid-body modes of the two bodies with free-free conditions. The converged time step used in the three simulations is $\Delta t = 0.003$. Comparisons of the three approaches are depicted in Fig. 2.

Excellent agreement is found between the three solutions. The time evolution of the tip of the two bodies is shown in Fig. 3. These results validate the multibody approach with a spherical type of connection, and also the implementation of gravity forces, which are more involved to include in the dynamical system as dead forces need to be rotated from the global to local reference frame in which the formulation is written (on the other hand aerodynamic forces will naturally rotate with the structural deformations).

B. Unsupported dynamics of very flexible structure

This example exemplifies the ability of our solvers to turn a generic linear finite-element model with free-free boundary conditions into a fully geometrically nonlinear solution that accounts for the rigid-body dynamics coupled with large elastic deformations. The test cases have already been presented in [11], though the novelties introduced herein are the new optimized 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 [24] and has served to verify several implementations of nonlinear beam dynamics with rigid body motions [25]. A straight structure of constant square cross section of side 3 and wall thickness 3/10 is built consisting of 784 shell elements linked to 50

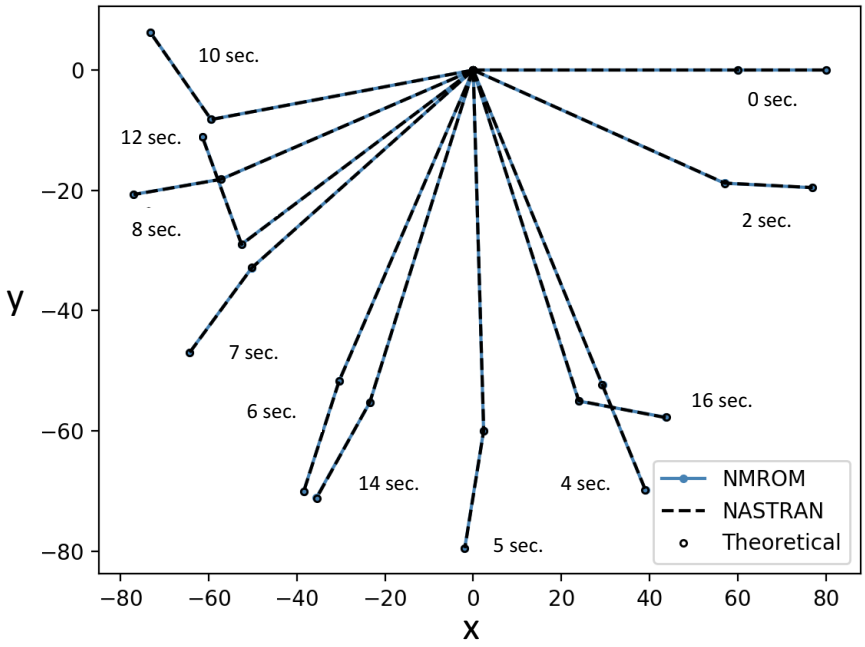
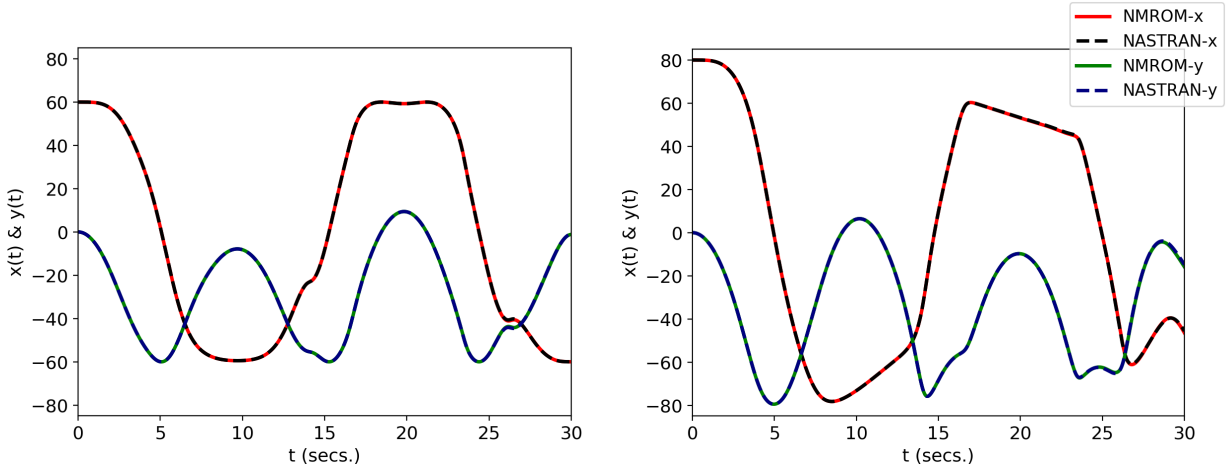


Fig. 2 Double-pendulum free falling under gravity



(a) First node (b) Second node

Fig. 3 Dynamic evolution of double-pendulum nodes

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.

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 (both using a Runge-Kutta 4 algorithm to time-march the equations of motion with $dt = 10^{-3}$). It is worth remarking the latter motion also exhibits large torsional deformations which are combined with the also large bending displacements and rigid-body modes.

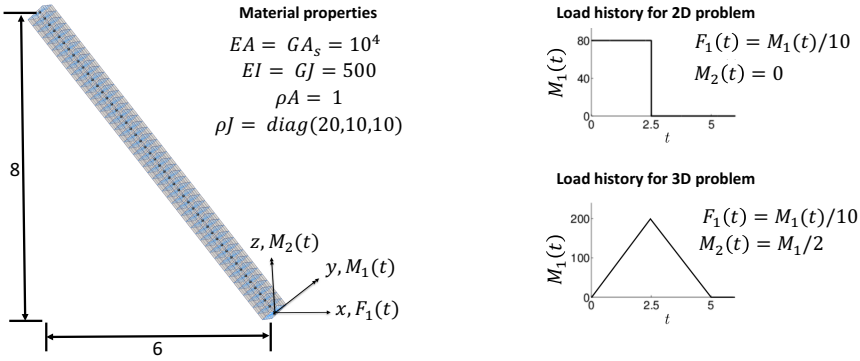


Fig. 4 FFS geometry, material properties and load cases

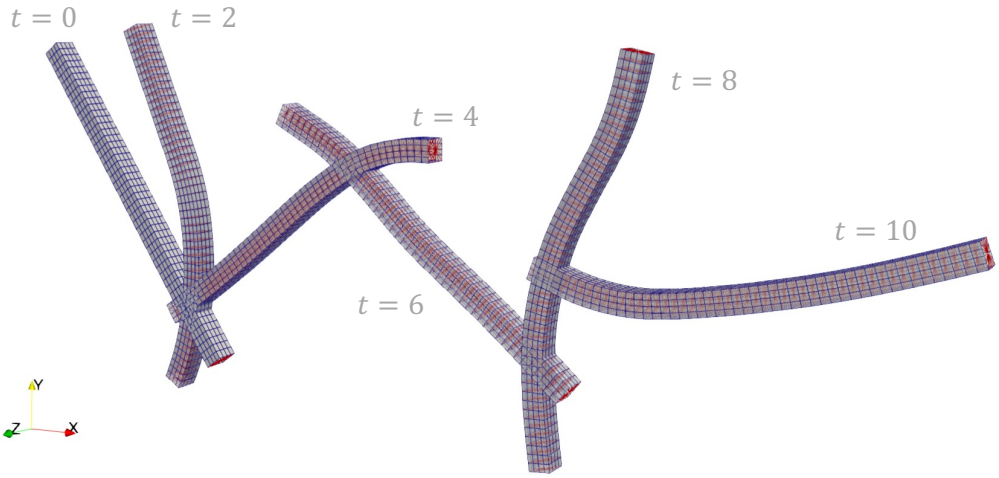


Fig. 5 Free-flying structure in the 2D-plane

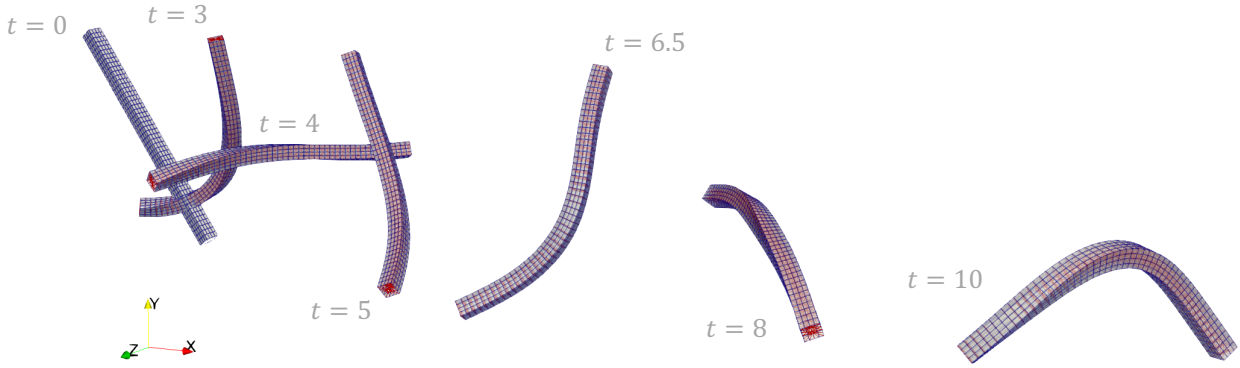


Fig. 6 Free-flying structure in the 3D-plane

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 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: 1) linking the integration time-step to the largest eigenvalue λ in the solution $dt = \lambda^{-0.5}$; and 2) 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 and Nvidia GPU RTX 6000 processors.

Table 1 FFB computational times in seconds and CG error

Nmodes	CPU-(1) (time/err)	GPU-(1)	CPU-(2)	GPU-(2)
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 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 6.9×10^{-5} versus the 2.54×10^{-5} of the planar case.

C. XRF1

The studies presented in this section are based on a reference aircraft configuration developed to industry standards known as XRF1 and representative of a long-range wide-body transport airplane. The version with a wing-tip extension described in [13] is employed to verify a gust response against NASTRAN linear solution. The FE model contains a total of around 177400 nodes, which are condensed into 176 active nodes along the reference load axes using linear interpolation elements. The structural FEM comprises of ~ 57000 CBEAM elements, ~ 55500 CQUAD4 shell elements, ~ 3800 CTRIA3 shell elements and ~ 800 rigid elements. A Guyan or static condensation approach is used for the

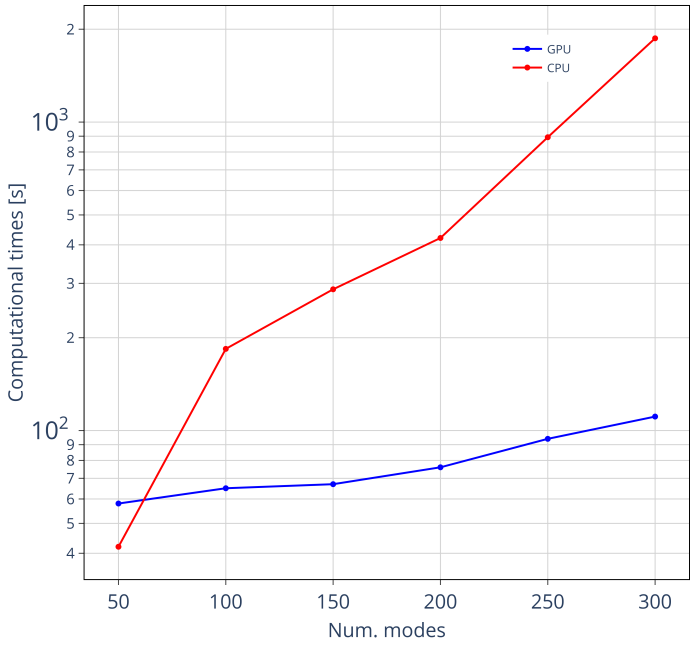


Fig. 7 Performance CPU vs GPU comparison in free-flying structure (fixed time step)

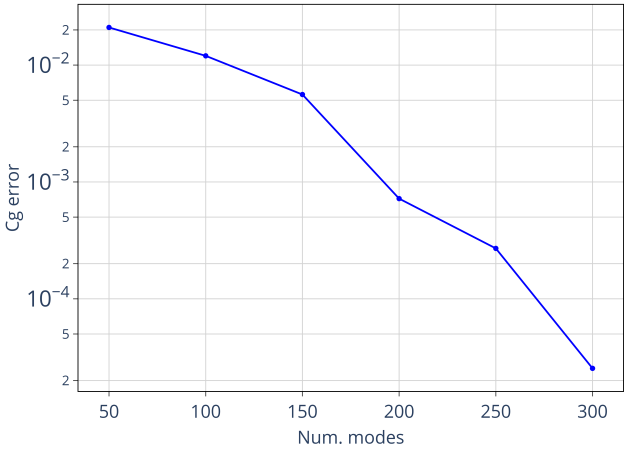


Fig. 8 Error metric in CG position for planar case

reduction. The aerodynamic model contains approximately 1500 aerodynamic panels. The simulations are carried out with a modal resolution of 70 modes following the convergence studies in [13].

1. Trim flight

The calculation of the trim equilibrium is carried out for increasing values of the gravity acceleration as to replicate a pull-up manoeuvre. The aircraft 3D trim equilibrium is shown in Fig. 9 for a range of pull-up manoeuvres from 1g to 3.5g. The reason to go beyond regulation requirements is to check the robustness of our solvers in large-deformations scenarios and also to appreciate better the differences between linear and nonlinear analysis. Moreover, by comparing with MSC Nastran linear static solution on the full FE model at small displacements, we verify our implementation of the trim. The airflow conditions are a density of $\rho = 0.778$ kg/m^3 and velocity $U_\infty = 180$ m/s. Shortening effects and follower forces become quite significant as the gravity loading increases.

Fig. 9 XRF1 3D trim for varying loading (green NMROM and red Linear Nastran solution)

Next in Fig. 10 we show the angle of attack and elevator angles at trim obtained from our nonlinear solver and the linear Nastran solution, which is performed on the full model without condensation. Note the follower effect of aerodynamic forces in the nonlinear model means they are less effective in counterbalancing the weight, which always points downwards, as larger deflections are present.

In addition to the static solution comparison, we verify our implementation by setting a dynamic simulation with gravity, angle of attack and the elevator angle from the static simulation as inputs. A good equilibrium state is attained with the vehicle flying straight over the course of 10 seconds dropping less that 10^{-4} m in altitude.

2. Gust response

Next we look at the gust response of the aircraft under trimmed conditions. Two high intensity 1-cos gusts are imposed on the 1-g trimmed flight: one of 67 m length, labeled (L=67), and another of 125 m length (L=125), both

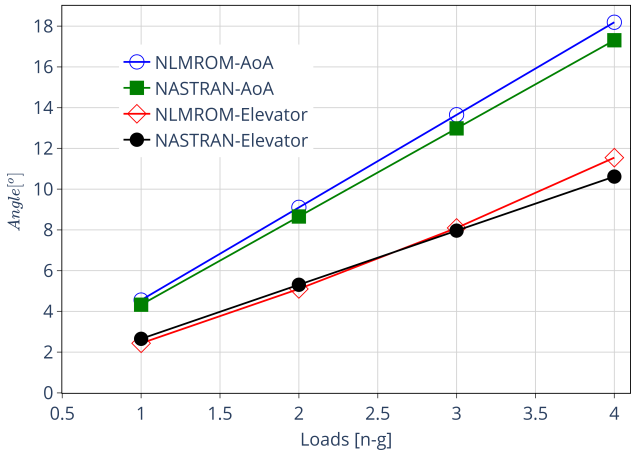


Fig. 10 XRF1 trim elevator (absolute value) and angle of attack

with intensity of 28.14 m/s. Similar to the trim case, these values are above regulations, and we use them to push our solvers into the region of deformations where the next generation of higher aspect ratio airplanes may operate. Fig. 11 presents the time evolution of the wing-tip normalized displacements (using the half-span metric) including the trim values at time zero and the differences between the linear and nonlinear analysis. They are not large on the overall tip displacements as expected for standard commercial-transport aircraft. Note displacements do not return to trim values as the aircraft includes rigid-body DoF and it is pushed up by the gust.

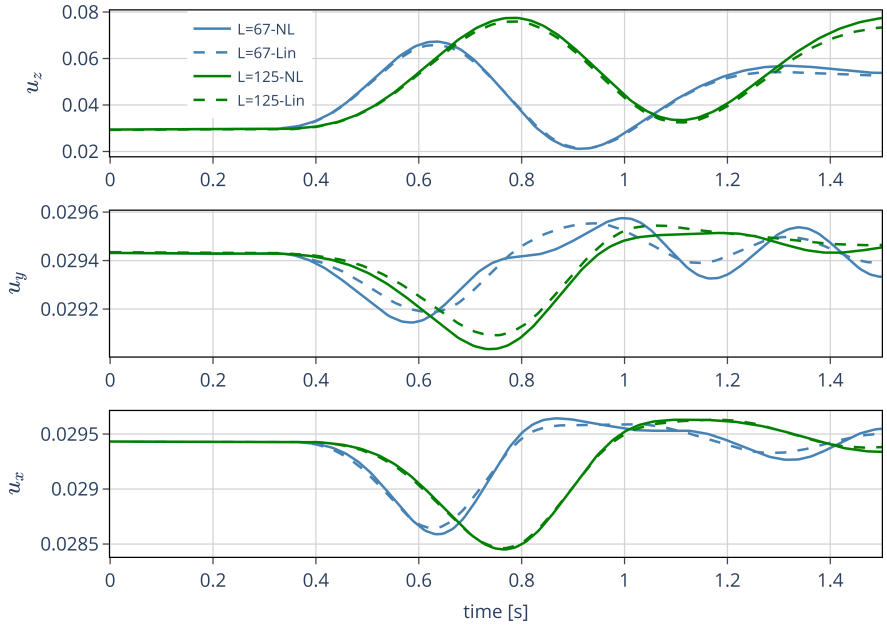


Fig. 11 Wing tip displacements for trim and gust analysis, `FENIAX` simulation

A more interesting metric to monitor is the loading at the root of the wing. As seen in Fig. 12, higher loads are found from the nonlinear analysis. This is also reported in [4] and means that the dynamic loads from the linear analysis

can be non-conservative. Safety factors are defined in part to account for those but their adequacy is not guaranteed in more flexible designs.

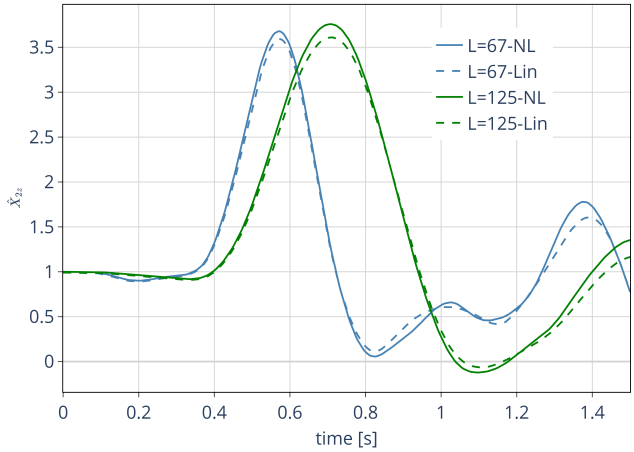


Fig. 12 Normalized wing vertical shear force time evolution with gust excitation, `FENIAX` simulation

Fig. 13 shows the load diagram for the normalized shear and torsional forces acting at the wing-root. This case illustrates the reason for building load envelopes with a large set of defining parameters: the sharper gust produces larger shear forces whereas the gust with a bigger length induces larger torsional moments. Similar behavior is seen for the bending-shear loads in Fig. 14.

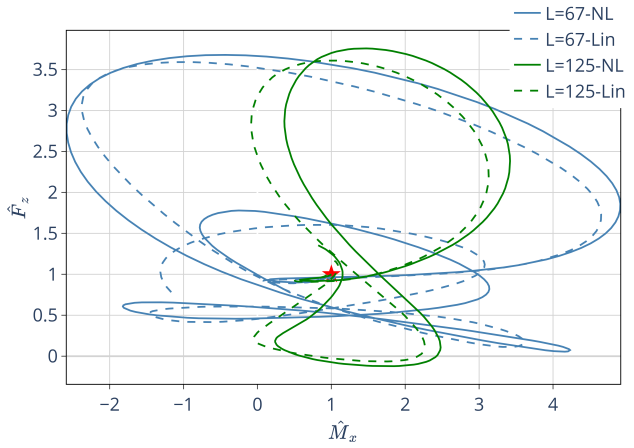


Fig. 13 Wing root torsion-shear load diagram in response to two 1-cos gust, `FENIAX` simulation

Finally the trim shape and the instantaneous shape at the peak of the gust are displayed in Fig. 15.

The simulation times encompassing both the trim and and the gust response are finally presented in table 2. The GPU performs nearly twice as fast as the CPU on these simulations with 70 modes resolution; as seen in the previous example this difference increases with the number of modes.

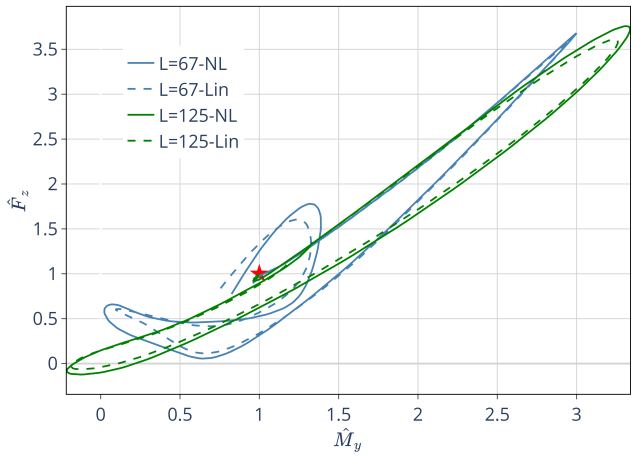


Fig. 14 Wing root bending-shear load diagram in response to two 1-cos gust, `FENIAX` simulation

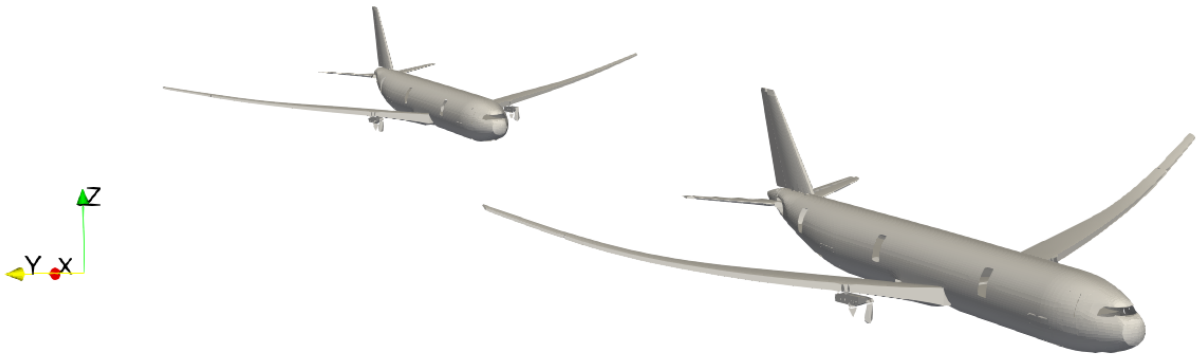


Fig. 15 XRF1 flying at trim equilibrium perturbed by gust (L=67)

Table 2 Computational times of the XRF1 trim-gust response (in seconds)

	Gust L=67		Gust L=125	
	Nonlin.	Lin.	Nonlin.	Lin.
CPU	79.3	31.2	82.2	33.1
GPU	49.9	17.8	46.3	17.6

IV. Conclusions

A modal-based formulation has been presented for the dynamics of flexible aircraft that accounts for geometrically nonlinear deformations, rigid-body motions, and the nonlinear interaction between them, including when multibody configurations are present. The approach is built on a non-intrusive reduction order process combined with a nonlinear description of the dominant dimension in slender structures. It achieves a nonlinear representation of linear aeroelastic models of arbitrary complexity in a very efficient manner and without losing the characteristics of the linear model. While the underlying theory had already been introduced, here we have first extended it to multibody problems, and we have introduced new implementation that achieves both high performance and software modularity. This is achieved by means of the numerical library JAX, which becomes the engine powering the computations and has allowed time-domain computations in near real-time with two orders of magnitude speed-ups compared to the previous state-of-the-art. In addition to this, a new algorithm that efficiently computes the aircraft trim equilibrium was also devised.

Three test cases have been presented herein to illustrate the aforementioned features in the code. Firstly, the multibody dynamics capabilities have been demonstrated on a double pendulum problem. Next, a very flexible, unsupported shell structure is shown to fly in vacuum undergoing large, coupled rigid and flexible motions. The previous two results have validated the implementation and produced some first insight on the computational advantages of the proposed approach. Finally, the aeroelastic response of a full aircraft model built to industry standards is analyzed. The trim equilibrium for various accelerations is computed and compared against the linear response provided by MSC Nastran. Next the gust response of the aircraft flying at trimmed conditions is studied, with the nonlinear analysis leading to slightly higher loads than its linear counterpart. On all of these cases we have shown how the program running the computations can be deployed on standard CPUs but also on modern hardware architectures such as GPUs that can lead to performance gains of over 30 times faster.

Acknowledgements

This work has received funds from Innovate UK, under project 10002372, managed by the UK Aerospace Technology Institute. The benchmark examples for the GPU computations were carried out at the Imperial College cluster. Code and examples are open-source except for the XRF1 model which is proprietary.

Appendices

A. But how fast is JAX?

In this section we assess the performance of JAX versus a lower level programming language. We set out to build a solution that is simple and transparent in terms of the computations involved, yet representative of the type of operations encountered along this work. A Monte Carlo simulation to solve a stochastic differential equation (SDE) was chosen

as the problem to benchmark JAX against C++. The paths generated follow a geometric Brownian motion stochastic process according to the SDE $dS_t = \mu S_t dt + \sigma dW_t$ with μ and σ modeled as constants and W_t being a Wiener process. One loop is required to go over the number of paths, N_p, and inside another loop for the total number of time steps, N_t, discretizing each path, with the application of one exponential function, a couple of floating point operations, and one random number generation per iteration. The time complexity of the algorithm is therefore $O(N_p \times N_t)$. The solution in JAX involves the combination of a `vmap` operation for the paths (since they are independent of each other) and one `scan` to generate the path, which is a recursive operation. This is similar, though much simpler, to the recovery of positions given the strain field over time as explained in Sec. II.D. An example of the paths being generated is given in Fig. 16.

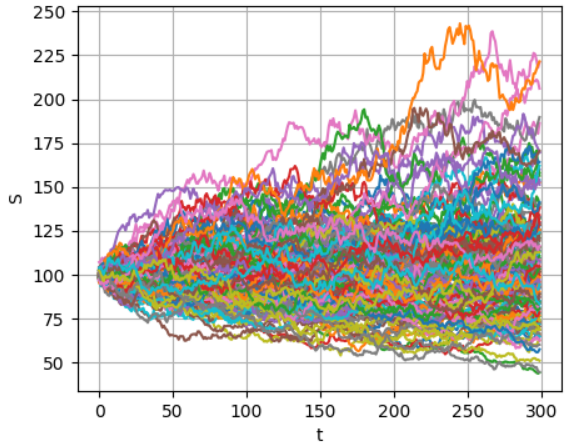


Fig. 16 Paths generated from GBM

Computations were run on one single CPU with an i7-6700 processor of 3.4 GHz clock speed. A JAX array and an `std::valarray` container are used to build the paths, both continuous arrays in memory. 5×10^5 paths were generated each with 300 time steps and the benchmark is shown in Table 3 (results include the average of 5 simulations with the corresponding standard deviation). Initially the better performance of JAX stood out compared to a standard C++ compilation [f0]; aggressive compilation with -O3 flag more than halve the speed and tailored compilation to the running architecture (-O3 -march=native [f2]) further improved the simulation time by nearly 2 seconds; finally (-O3 -march=native -ftree-vectorize -flto [f3]) brought the time slightly close to JAX, but still almost two times slower. The reason for this disparity is that JAX automatically activates multi-threading in some of its operations while the C++ implementation runs on a single thread by default. Thus we built a concurrent implementation using the std library `thread`, which led to the fastest results in the benchmark but surpassing JAX by only 4%. Given the ability to work closer to the hardware, C++ will always be capable of achieving higher computational speeds; the trade-off is how much effort is involved in doing so. This gives an idea of the power of JAX for Scientific Computing.

Since JAX is optimized for GPU deployment, we went one step further and wrote a CUDA kernel to solve the same

Table 3 Montecarlo simulation benchmarks on CPU

	Time [s]
JAX	2.57 ± 0.03
CPP-Threading	2.46 ± 0.03
CPP-f3	5.11 ± 0.05
CPP-f2	5.98 ± 0.08
CPP-f1	7.71 ± 0.09
CPP-f0	17.6 ± 0.6

computational problem, in this case increasing the number of paths to 4×10^6, nearly one order of magnitude larger, but leading to speeds similar to the CPU runs. Results in Table 4 show almost identical computational speeds between both implementations, further highlighting JAX strength.

Table 4 Montecarlo simulation benchmarks on GPU

	Time [s]
JAX	2.58 ± 0.07
CUDA	2.56 ± 0.02

B. Lower kinematic pairs for multibody mechanisms

In section II.B we have shown the mathematical framework in order to couple, via a point connection, two independent flexible bodies. Displacement and rotational DoF are to be restricted as to build the required kinematic pairs. The simplicity of the resulting Jacobians is one the strengths of the formulation. Fig. 17 presents some of the most common types of connections that will be modeled within the intrinsic nonlinear formulation. A spherical pair or joint restricts the three positional DoF of the connecting node of the second-body to be equal to those of the first-body, and rotations are independent in the two bodies; therefore out of the twelve DoF in the two nodes, three are constrained in this link. In the hinge or revolute joint, similar to the spherical pair, the second-body is bound to move with the first-body, however, only rotation along a hinge-axis is permitted in the second-body; thus there is only one independent DoF in the second-body. The cylindrical joint allows independent movement and rotation in the second-body only through one axis. The prismatic joint is similar to the cylindrical joint, adding to it a constraint in the free rotational DoF, such that only a linear movement in one direction with respect the first-body is allowed in the second one. Fig. 17 shows with arrows the independent DoF of the second-body for these four joints. Other types of joints can be derived as a combination of these ones, such as the screw joint; and even higher kinematic pairs can be put as a combination of these joints, the universal joint is an example of it, consisting of two hinges in orthogonal planes.

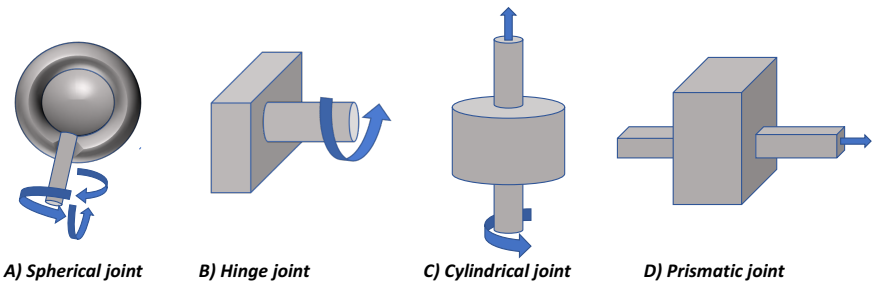


Fig. 17 Typical examples of lower kinematic pairs

1. Spherical joint

The three positional constraints in the spherical joint are modelled by equating the global velocities at each connecting node, giving three algebraic equations:

$$\boldsymbol{g}^{v} = \dot{\boldsymbol{r}}_G^{(1)} - \dot{\boldsymbol{r}}_G^{(2)} = \boldsymbol{0} \tag{14}$$

where $\boldsymbol{r}_G^{(\mathrm{i})}$ is the position in the global reference frame and $\dot{\boldsymbol{r}}_G^{(\mathrm{i})} = \boldsymbol{R}^{(\mathrm{i})}\mathbf{v}^{(\mathrm{i})} = \boldsymbol{R}^{(\mathrm{i})}\boldsymbol{\Phi}_{1u}^{(\mathrm{i})}\mathbf{q}_1^{(\mathrm{i})}$. The Jacobian of $\dot{\boldsymbol{r}}_a$ with respect to the modal velocities is given by the function $\boldsymbol{G}_q^v$,

$$\boldsymbol{G}_q^v(\boldsymbol{\zeta}) = \frac{\partial \dot{\boldsymbol{r}}_G}{\partial \mathbf{q}_1} = \boldsymbol{R}(\boldsymbol{\zeta})\boldsymbol{\Phi}_{1u} \tag{15}$$

similarly the function $\boldsymbol{G}_\zeta^v$ gives the Jacobian of $\dot{\boldsymbol{r}}_G$ with respect to the rotational quaternions,

$$\boldsymbol{G}_\zeta^v(\mathbf{v},\boldsymbol{\zeta}) = \frac{\partial \dot{\boldsymbol{r}}_G}{\partial \boldsymbol{\zeta}} = \frac{\partial R^{ij}}{\partial \zeta^k}\mathrm{v}_j \boldsymbol{e}_i \otimes \boldsymbol{e}_k =$$

$$\begin{bmatrix} 2\zeta_0\mathrm{v}_1 + 2\zeta_2\mathrm{v}_3 - 2\zeta_3\mathrm{v}_2 & 2\zeta_1\mathrm{v}_1 + 2\zeta_2\mathrm{v}_2 + 2\zeta_3\mathrm{v}_3 & 2\zeta_0\mathrm{v}_3 + 2\zeta_1\mathrm{v}_2 - 2\zeta_2\mathrm{v}_1 & -2\zeta_0\mathrm{v}_2 + 2\zeta_1\mathrm{v}_3 - 2\zeta_3\mathrm{v}_1 \\ 2\zeta_0\mathrm{v}_2 - 2\zeta_1\mathrm{v}_3 + 2\zeta_3\mathrm{v}_1 & -2\zeta_0\mathrm{v}_3 - 2\zeta_1\mathrm{v}_2 + 2\zeta_2\mathrm{v}_1 & 2\zeta_1\mathrm{v}_1 + 2\zeta_2\mathrm{v}_2 + 2\zeta_3\mathrm{v}_3 & 2\zeta_0\mathrm{v}_1 + 2\zeta_2\mathrm{v}_3 - 2\zeta_3\mathrm{v}_2 \\ 2\zeta_0\mathrm{v}_3 + 2\zeta_1\mathrm{v}_2 - 2\zeta_2\mathrm{v}_1 & 2\zeta_0\mathrm{v}_2 - 2\zeta_1\mathrm{v}_3 + 2\zeta_3\mathrm{v}_1 & -2\zeta_0\mathrm{v}_1 - 2\zeta_2\mathrm{v}_3 + 2\zeta_3\mathrm{v}_2 & 2\zeta_1\mathrm{v}_1 + 2\zeta_2\mathrm{v}_2 + 2\zeta_3\mathrm{v}_3 \end{bmatrix} \tag{16}$$

with v_j and ζ_k the components of the local velocities and the rotational quaternions respectively. Using Eqs. (15) and (16), the multi-body system in Eqs. (7) is assembled with the Lagrange multipliers in (11) and the Jacobians of the system as,

$$\begin{aligned} \boldsymbol{G}_{1q}^v &= \boldsymbol{G}_q^v(\boldsymbol{\zeta}^{(1)}) = \boldsymbol{R}^{(1)}\boldsymbol{\Phi}_{1u}^{(1)} \\ \boldsymbol{G}_{2q}^v &= -\boldsymbol{G}_q^v(\boldsymbol{\zeta}^{(2)}) = -\boldsymbol{R}^{(2)}\boldsymbol{\Phi}_{1u}^{(2)} \\ \boldsymbol{G}_{1\zeta}^v &= \boldsymbol{G}_\zeta^v(\mathbf{v}^{(1)},\boldsymbol{\zeta}^{(1)}) \\ \boldsymbol{G}_{2\zeta}^v &= -\boldsymbol{G}_\zeta^v(\mathbf{v}^{(2)},\boldsymbol{\zeta}^{(2)}) \end{aligned} \tag{17}$$

Spherical joints can be found in joystick devices, the steering knuckle part of automotive suspensions or the human hip-joint, making it an important type of joint in the multibody modelling.

2. Hinged joint

For the revolute or hinge joint three equations, $\boldsymbol{g}^v$, represent the attachment of the two connecting nodes, exactly as in the spherical joint; and two $\boldsymbol{g}^\omega$ equations limit the rotation of the second-body with respect to the first-body to an axis of rotation as follows: let $\boldsymbol{p}_0$ be a unit vector along the hinge axis, the velocity (in the global frame) of the end-point of $\boldsymbol{p}_0$, $\mathbf{v}_{Gp}$, will be equal in both reference systems:

$$\mathbf{v}_{Gp} = \dot{\boldsymbol{r}}_G^{(1)} + \dot{\boldsymbol{p}}_{01} + \tilde{\boldsymbol{\omega}}^{(1)}\boldsymbol{p}_{01} = \dot{\boldsymbol{r}}_G^{(1)} + \tilde{\boldsymbol{\omega}}^{(1)}\boldsymbol{p}_{01} = \dot{\boldsymbol{r}}_G^{(2)} + \tilde{\boldsymbol{\omega}}^{(2)}\boldsymbol{p}_{02} \tag{18}$$

where $\boldsymbol{p}_{0i}$ is the hinge-axis vector $\boldsymbol{p}_0$ in the initial local reference system of the body i, and its derivative $\dot{\boldsymbol{p}}_{0i}$ is equal to $\mathbf{0}$ as to model a rigid hinge. The constraints will then be,

$$\begin{aligned} \boldsymbol{g}^v &= \dot{\boldsymbol{r}}_G^{(1)} - \dot{\boldsymbol{r}}_G^{(2)} = \mathbf{0} \\ \boldsymbol{g}^\omega &= \tilde{\boldsymbol{\omega}}^{(1)}\boldsymbol{p}_{01} - \tilde{\boldsymbol{\omega}}^{(2)}\boldsymbol{p}_{02} = \mathbf{0} \end{aligned} \tag{19}$$

The positional constraints $\boldsymbol{g}^v$ are similar to the spherical pair so the Jacobians, $\boldsymbol{G}^v_{q_1}$, $\boldsymbol{G}^v_{q_2}$, $\boldsymbol{G}^v_{\zeta_1}$ and $\boldsymbol{G}^v_{\zeta_2}$, are equal to those given in Eqs. (17). The Jacobians of the rotational constraints and the rest of the terms needed to complete the Lagrange multipliers in Eq. (11) are,

$$\begin{aligned} \left[\boldsymbol{G}^\omega_{1q}\ \boldsymbol{G}^\omega_{2q}\right] &= \left[\tilde{\boldsymbol{\Phi}}^{(1)}_\theta \boldsymbol{p}_{01} \quad -\tilde{\boldsymbol{\Phi}}^{(2)}_\theta \boldsymbol{p}_{02}\right] \\ \boldsymbol{G}^\omega_{1\zeta} &= \boldsymbol{G}^\omega_{2\zeta} = \mathbf{0} \\ \left[\partial_t \boldsymbol{g}^v\ \partial_t \boldsymbol{g}^\omega\right] &= \mathbf{0} \end{aligned} \tag{20}$$

Note that $\boldsymbol{G}^\omega_q$ are defined in a subspace which is the plane perpendicular to $\boldsymbol{p}_0$ and therefore $\boldsymbol{G}^\omega_{qi}\boldsymbol{G}^{\omega\top}_{qi}$ is not full rank – which is consistent with the five DoF that have to be constrained in this kinematic pair. One of the rows in $\boldsymbol{G}^\omega_q$ is zero or dependent on the other two and has to be removed, yielding 2 constraints for the rotation, as expected. The simplicity of the rotational constraints is also remarkable

Hinge joints appear in all sort of mechanisms, from engine systems to deployable wings.

3. Cylindrical joint

In the cylindrical pair problem the 2 bodies share an axis through which free rotations and displacements may occur. Constraining rotational DoF is done in a similar fashion to the hinge joint. Despite the reference systems moving

differently along the axis of rotation, the assumption by which the velocity of the end point of $\boldsymbol{p}_0$ is equal in both reference systems remains valid. For the positional constraints, the velocity component perpendicular to the axis of rotation has to be the same in both bodies (note that no distinction is made here for $\boldsymbol{p}_0$ in the reference systems as they can be initially defined such that $\boldsymbol{p}_0$ is a unit-normed vector in the x-direction of both bodies). And similarly to the rotational constraints, which are defined in a subspace, here the positional constraints are defined through the subspace of the velocities perpendicular to the axis of rotation. The 3 equations are not independent and can be reduced to 2, by removing one row in the Jacobians. These can be calculated in the same way as in the hinge problem. The constraints are,

$$\begin{aligned} \boldsymbol{g}^v &= \boldsymbol{R}^{(1)} \left(\mathbf{v}^{(1)} - (\mathbf{v}^{(1)} \cdot \boldsymbol{p}_0) \boldsymbol{p}_0 \right) - \boldsymbol{R}^{(2)} \left(\mathbf{v}^{(2)} - (\mathbf{v}^{(2)} \cdot \boldsymbol{p}_0) \boldsymbol{p}_0 \right) = \mathbf{0} \\ \boldsymbol{g}^\omega &= (\tilde{\boldsymbol{\omega}}^{(1)} - \tilde{\boldsymbol{\omega}}^{(2)}) \boldsymbol{p}_0 = \mathbf{0} \end{aligned} \tag{21}$$

and the resulting Jacobians,

$$\begin{aligned} \left[\boldsymbol{G}^v_{1q} \; \boldsymbol{G}^v_{2q} \right] &= \left[\boldsymbol{R}^{(1)} \left(\boldsymbol{I}_3 - \boldsymbol{p}_0 \otimes \boldsymbol{p}_0 \right) \boldsymbol{\Phi}^{(1)}_{1u} \quad - \boldsymbol{R}^{(2)} \left(\boldsymbol{I}_3 - \boldsymbol{p}_0 \otimes \boldsymbol{p}_0 \right) \boldsymbol{\Phi}^{(2)}_{1u} \right] \\ \left[\boldsymbol{G}^v_{1\zeta} \; \boldsymbol{G}^v_{2\zeta} \right] &= \left[\boldsymbol{G}^v_{\zeta} \left(\left(\boldsymbol{I}_3 - \boldsymbol{p}_0 \otimes \boldsymbol{p}_0 \right) \mathbf{v}^{(1)}, \boldsymbol{\zeta}^{(1)} \right) \quad - \boldsymbol{G}^v_{\zeta} \left(\left(\boldsymbol{I}_3 - \boldsymbol{p}_0 \otimes \boldsymbol{p}_0 \right) \mathbf{v}^{(2)}, \boldsymbol{\zeta}^{(2)} \right) \right] \\ \left[\boldsymbol{G}^\omega_{1q} \; \boldsymbol{G}^\omega_{2q} \right] &= \left[\tilde{\boldsymbol{\Phi}}^{(1)}_{1\theta} \boldsymbol{p}_0 \quad - \tilde{\boldsymbol{\Phi}}^{(2)}_{1\theta} \boldsymbol{p}_0 \right] \\ \left[\boldsymbol{G}^\omega_{1\zeta} \; \boldsymbol{G}^\omega_{2\zeta} \right] &= \left[\mathbf{0} \quad \mathbf{0} \right] \\ \left[\partial_t \boldsymbol{g}^v \; \partial_t \boldsymbol{g}^\omega \right] &= \mathbf{0} \end{aligned} \tag{22}$$

with the function $\boldsymbol{G}^v_{\zeta}$ given by Eq. (16).

An example with this pair that would also require geometrically-nonlinear capabilities is that of a flexible catenary formed by a cable and a mass attached to it through a cylindrical joint to model a zip-line.

4. Prismatic pair

For this joint, the second body has only one DoF unconstrained to slide through a common axis $\boldsymbol{p}_0$. If the initial reference frames are chosen to be equal, they will remain so as the three rotational DoF of the second body are constrained. This allows putting the positional constraints in the local frame:

$$\begin{aligned} \boldsymbol{g}^v &= \left(\mathbf{v}^{(1)} - \mathbf{v}^{(2)} \right) \times \boldsymbol{p}_0 = \mathbf{0} \\ \boldsymbol{g}^\omega &= \boldsymbol{\omega}^{(1)} - \boldsymbol{\omega}^{(2)} = \mathbf{0} \end{aligned} \tag{23}$$

yet again the three $\boldsymbol{g}^v$ equations are not independent. In fact, if $\boldsymbol{e}_1$ of the initial reference systems is chosen to be parallel to $\boldsymbol{p}_0$, the two resulting constraint equations will be just $\mathrm{v}_2^{(1)} = \mathrm{v}_2^{(1)}$ and $\mathrm{v}_3^{(2)} = \mathrm{v}_3^{(2)}$. The Jacobians are,

$$
\begin{aligned}
\left[\boldsymbol{G}_{q_1}^v \; \boldsymbol{G}_{q_2}^v\right] &= \left[\tilde{\boldsymbol{\Phi}}_{1u}^{(1)} \boldsymbol{p}_0 \quad -\tilde{\boldsymbol{\Phi}}_{1u}^{(2)} \boldsymbol{p}_0\right] \\
\left[\boldsymbol{G}_{q_1}^\omega \; \boldsymbol{G}_{q_2}^\omega\right] &= \left[\boldsymbol{\Phi}_{1\theta}^{(1)} \quad -\boldsymbol{\Phi}_{1\theta}^{(2)}\right] \\
\boldsymbol{G}_{\zeta_1}^\omega &= \boldsymbol{G}_{\zeta_2}^\omega = \boldsymbol{0} \\
\left[\partial_t \boldsymbol{g}^v \; \partial_t \boldsymbol{g}^\omega\right] &= \boldsymbol{0}
\end{aligned}
\tag{24}
$$

A slider-crank mechanism transforms linear into angular motion, and is formed with three revolute joints and one prismatic joint, which in the modelling of a combustion engine represents the piston motion.

References

[1] Livne, E., "Aircraft Active Flutter Suppression: State of the Art and Technology Maturation Needs," *Journal of Aircraft*, Vol. 55, No. 1, 2018, pp. 410–452. https://doi.org/10.2514/1.C034442.

[2] Gray, A. C., and Martins, J. R., "Geometrically Nonlinear High-fidelity Aerostructural Optimization for Highly Flexible Wings," *AIAA Scitech 2021 Forum*, 2021. https://doi.org/10.2514/6.2021-0283.

[3] Artola, M., Goizueta, N., Wynn, A., and Palacios, R., "Aeroelastic Control and Estimation with a Minimal Nonlinear Modal Description," *AIAA Journal*, Vol. 59, No. 7, 2021, pp. 2697–2713. https://doi.org/10.2514/1.j060018.

[4] Cesnik, C. E., Palacios, R., and Reichenbach, E. Y., "Reexamined Structural Design Procedures for Very Flexible Aircraft," *Journal of Aircraft*, Vol. 51, No. 5, 2014, pp. 1580–1591. https://doi.org/10.2514/1.C032464.

[5] Kier, T. M., "An Integrated Model for Lateral Gust Loads Analysis and Dutch Roll Flight Dynamics Using a 3d Panel Method," 2017. https://doi.org/10.7150/jca.24838.

[6] Sanghi, D., Cesnik, C. E. S., and Riso, C., "Roll Maneuverability of Transonic High-Aspect-Ratio-Wing Aircraft with Flared Folding Wingtips," *Journal of Aircraft*, 2024, pp. 1–14. https://doi.org/10.2514/1.C037470.

[7] Choi, Y., Boncoraglio, G., Anderson, S., Amsallem, D., and Farhat, C., "Gradient-Based Constrained Optimization Using a Database of Linear Reduced-Order Models," *Journal of Computational Physics*, Vol. 423, 2020, p. 109787. https://doi.org/10.1016/j.jcp.2020.109787.

[8] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K., "Projection-Based Model Reduction: Formulations for Physics-Based Machine Learning," *Computers & Fluids*, Vol. 179, 2019, pp. 704–717. https://doi.org/10.1016/j.compfluid.2018.07.021.

[9] Riso, C., and Cesnik, C. E. S., "Impact of Low-Order Modeling on Aeroelastic Predictions for Very Flexible Wings," *Journal of Aircraft*, Vol. 60, No. 3, 2023, pp. 662–687. https://doi.org/10.2514/1.C036869.

[10] Drachinsky, A., and Raveh, D. E., "Nonlinear Aeroelastic Analysis of Highly Flexible Wings Using the Modal Rotation Method," *AIAA Journal*, Vol. 60, No. 5, 2022, pp. 3122–3134. https://doi.org/10.2514/1.J061065.

[11] Palacios, R., and Cea, A., "Nonlinear Modal Condensation of Large Finite Element Models: Application of Hodges's Intrinsic Theory," *AIAA Journal*, Vol. 57, No. 10, 2019, pp. 4255–4268. https://doi.org/10.2514/1.J057556.

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

[13] Cea, A., and Palacios, R., "Geometrically Nonlinear Effects on the Aeroelastic Response of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 60, No. 1, 2023, pp. 205–220. https://doi.org/10.2514/1.C036740.

[14] Cea, A., and Palacios, R., "JAX-based Aeroelastic Simulation Engine for Differentiable Aircraft Dynamics," *Computer Physics Communications*, Vol. 311, No. 109547, 2025. https://doi.org/10.1016/j.cpc.2025.109547.

[15] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., "JAX: Composable Transformations of Python+NumPy Programs," , 2018.

[16] Palacios, R., and Cesnik, C. E. S., *Dynamics of Flexible Aircraft: Coupled Flight Mechanics, Aeroelasticity, and Control*, Cambridge University Press, 2023. https://doi.org/10.1017/9781108354868.

[17] Roger, K. L., "Airplane Math Modeling Methods for Active Control Design," *Structural Aspects of Active Controls, AGARD CP-228*, 1977, pp. 4.1–4.11.

[18] Palacios, R., and Cea, A., "Polynomial Preconditioning in Unsteady Aerodynamic Models for Aeroelasticity," *RAeS 8th Aircraft Structural Design Conference*, 2023-10-03/2023-10-04.

[19] Wang, Y., Palacios, R., and Wynn, A., "A Method for Normal-Mode-Based Model Reduction in Nonlinear Dynamics of Slender Structures," *Computers and Structures*, Vol. 159, 2015, pp. 26–40. https://doi.org/10.1016/j.compstruc.2015.07.001.

[20] Guyan, R. J., "Reduction of Stiffness and Mass Matrices," *AIAA Journal*, Vol. 3, No. 2, 1965, pp. 380–380. https://doi.org/10.2514/3.2874.

[21] Cea, A., and Palacios, R., "A Non-Intrusive Geometrically Nonlinear Augmentation to Generic Linear Aeroelastic Models," *Journal of Fluids and Structures*, Vol. 101, 2021, p. 103222. https://doi.org/10.1016/j.jfluidstructs.2021.103222.

[22] Palacios, R., Murua, J., and Cook, R., "Structural and Aerodynamic Models in Nonlinear Flight Dynamics of Very Flexible Aircraft," *AIAA Journal*, Vol. 48, No. 11, 2010, pp. 2648–2659. https://doi.org/10.2514/1.J050513.

[23] Cheng, K. C.-W., Cea, A., Palacios, R., Castrichini, A., and Wilson, T., "Nonlinear Multibody Modeling of Flexible Aircraft with Flared Hinged Wingtips," *Journal of Aircraft*, 2025, p. 1–20. https://doi.org/10.2514/1.c038094.

[24] Simo, J. C., and Vu-Quoc, L., "On the Dynamics in Space of Rods Undergoing Large Motions - A Geometrically Exact Approach," *Computer Methods in Applied Mechanics and Engineering*, Vol. 66, No. 2, 1988, pp. 125–161. https://doi.org/10.1016/0045-7825(88)90073-4.

[25] Hesse, H., Palacios, R., and Murua, J., "Consistent Structural Linearization in Flexible Aircraft Dynamics with Large Rigid-Body Motion," *AIAA Journal*, Vol. 52, No. 3, 2014, pp. 528–538. https://doi.org/10.2514/1.J052316.