ENABLING EFFICIENT CRASH ANALYSES IN AIRCRAFT PRELIMINARY DESIGN: AN EXPLICIT NON-INTRUSIVE METHODOLOGY USING NEURAL DIFFERENTIAL EQUATIONS

H. Dahmen*, M. Haupt*, S. Heimbs*
* Technische Universität Braunschweig, Institute of Aircraft Design and Lightweight Structures (IFL),
Hermann-Blenk-Str. 35, 38108 Braunschweig, Germany

Abstract

Crash is a load case that is mandatory for the certification of aircrafts. To evaluate the influence of novel structural components, crash analyses should ideally be integrated into preliminary aircraft design, but the computational demands of detailed finite element simulations make this approach currently impractical. This paper presents a novel non-intrusive methodology that combines substructuring, mesh generalisation, proper orthogonal decomposition-based model order reduction, and neural differential equations to enable efficient crash analyses during early design stages.
The methodology decomposes complex aircraft fuselage structures into manageable substructures, which are represented by individual surrogate models trained on finite element simulations. Mesh generalisation using basis functions decouples surrogate models from specific mesh configurations, enabling representation of diverse geometries within a unified framework. Proper orthogonal decomposition-based dimensional reduction compresses the solution space, whilst neural differential equations learn the underlying crash physics without explicit time parameterisation in the latent space.
The methodology is demonstrated through a vertical strut substructure. The neural differential equation implementation successfully captures essential crash behaviour characteristics, though dynamic lag indicates areas for improvement in temporal response prediction.
The framework provides a foundation for incorporating crashworthiness considerations into iterative preliminary design processes, potentially reducing reliance on costly late-stage design modifications.

Keywords

Aircraft Crashworthiness; Surrogate Modelling; Neural Differential Equations; Model Order Reduction

1. INTRODUCTION

The aviation industry faces increasing pressure to develop sustainable aircraft technologies while maintaining the highest safety standards. Future aircraft designs will integrate novel propulsion systems, lightweight materials, and innovative structural configurations that significantly alter conventional system architectures. Crashworthiness assessment represents a critical design constraint that directly impacts both sustainability and performance objectives. In industrial processes crashworthiness is typically considered late in the development process, so that problems with new designs can only be remedied with major drawbacks in terms of performance and costs.
The primary barrier to early-stage crashworthiness assessment lies in the high computational expense of large-scale finite element (FE) crash simulations. Traditional crash analyses require detailed finite element models with millions of degrees of freedom, resulting in prohibitively long computation times that are incompatible with the iterative nature of preliminary design processes. To address this challenge, the LuFo VI-3 project HYFLIP is developing rapid analysis methods that enable crashworthiness and ditching load cases to be considered during the preliminary design stage of aircraft development.
This paper presents a novel methodology that combines a substructuring technique, mesh generalisation, proper orthogonal decomposition (POD)-based model order reduction, and neural ordinary differential equations (NeuralODEs) to achieve computationally efficient crash analyses.
The proposed methodology reduces computational complexity through a systematic approach: First, substructuring concepts minimise the number of design variables required for surrogate model development. Second, the surrogate model operates on a generalised mesh topology and employs POD as a linear reduction technique to further decrease the number of degrees of freedom. Finally, a NeuralODE implementation solves the reduced-order system in the temporal domain by utilising a neural network to calculate temporal derivatives of the system state, coupled with a standard ODE solver to capture temporal evolution during both prediction and training phases.
This paper demonstrates the methodology using a vertical strut from a typical fuselage section, which connects the frame and passenger crossbeam. Vertical struts are the focus of research into optimising crash kinematics, as they have the potential to incorporate crash absorber structures. For example, Heimbs [1] investigated the use of composite crash absorbers. This paper illustrates both the technical approach and training strategy for a surrogate model of a typical vertical strut design.

2. STATE OF THE ART

2.1. Crashworthiness Requirements

Crashworthiness requirements for large aircraft are specified in the certification specification for large aeroplanes (CS-25) under regulation 25.561 an 25.562 [2]. As summarised in by Guida et. al. [3], the survival of passengers depends on five essential criteria being met. A survival space for passengers and escape routes must remain clear

doi: 10.25967/650048

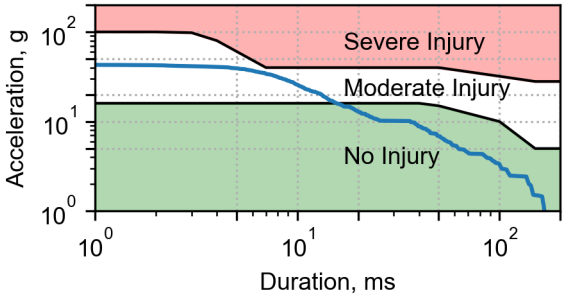
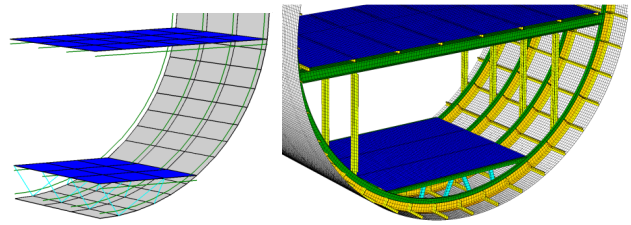


FIG 1. Eiband diagram for assessing human injury based on acceleration curve



(a) GFEM half model of PrADO **(b) DFEM of fuselage generator**

FIG 2. FE models with different levels of detail

in the event of an accident. Fire hazards must not arise and heavy objects must be restrained. Furthermore, the loads on passengers must remain within acceptable limits, which requires careful evaluation of the acceleration amplitude, duration, direction and impulse characteristics. Eiband's [4] diagram, shown in Figure 1, establishes a correlation between acceleration amplitude, duration and risk of injury. It thus provides a rapid assessment of the threat to passengers. The acceleration experienced by a passenger in a typical seat approval test simulation is plotted in blue on the diagram. Consequently, any rapid method for crashworthiness evaluation in preliminary design must provide accurate predictions of both passenger accelerations and structural deformations to ensure comprehensive safety assessment.

2.2. FE Models in Preliminary Design

From a preliminary design perspective, the goal is to incorporate crash analyses into an iterative design optimisation programme. The IFL tool PrADO by Heinze et. al. [5, 6] exemplifies such a programme, incorporating multiple load cases from the flight envelope, ground loads, and cabin pressure, though crash cases are not included. PrADO utilises both handbook methods and a global finite element model (GFEM), which represents the structure using simplified beam and shell elements. Structural parameters are subsequently optimised to satisfy all implemented load cases whilst achieving lightweight solutions.

However, elastic-plastic behaviour and damage mechanisms cannot be accurately modelled using a GFEM. For crashworthiness analyses, the model must be converted to a detailed finite element model (DFEM), which captures structural buckling phenomena through comprehensive shell element representation. The distinction between GFEM and DFEM approaches is illustrated in Figure 2. The GFEM represents frames, struts, and crossbeams as beam elements, whilst the DFEM utilises shell elements for these components. Stringers, clips, and vertical struts are exclusively modelled in the DFEM to capture the detailed structural behaviour required for accurate crash analyses.

Numerous research efforts have addressed this challenge by developing tools for automatic DFEM generation based on preliminary design data. Particularly noteworthy is the work conducted at the DLR, which has established a comprehensive infrastructure including the common parametric aircraft configuration schema (CPACS) [7, 8] and specialised model generators such as AC-CRASH [9], PANDORA [10], and FUGA [11, 12, 13].

The computational process chain for crashworthiness evaluation during the preliminary design stage through FE simulations is illustrated in Figure 3. Design parameters are optimised through iterative evaluation of crash simulation results alongside other load cases. The crash model generation process begins with the creation of a geometric model, followed by meshing and the FE simulation. The simulation phase represents the primary computational bottleneck, as high mesh density is required to achieve accurate results, and complex material models are necessary to capture plasticity and damage behaviour. Even for relatively modest models, such as the typical fuselage section shown in Figure 2, simulation times can extend to several hours on high-performance computing systems. Additionally, geometric modelling can consume 10-20% of the total iteration time when extensive geometric-cutting operations are required.

Within the methodology of this study, a model generator is required to create finite element crash models for the designated substructures, subsequently enabling surrogate model training using simulation results. The process chain of Figure 3 is employed for this purpose. To ensure seamless integration between geometric modelling and finite element model generation, both processes are combined using the complete Abaqus environment (Abaqus/CAE). The DFEM model of Figure 2 is created with this fuselage generator. The GFEM on the other hand, is a model of the preliminary design tool PrADO.

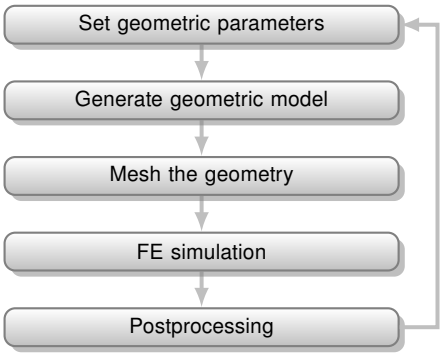


FIG 3. Process chain for evaluating crashworthiness with FE simulations

3. METHODOLOGY

To achieve substantial computational speed-ups, an ideal surrogate model should replace finite element simulations with a significantly faster method and ether accelerate the model generation or eliminate it entirely. The surrogate model inputs comprise structural design parameters, whilst outputs consist of passenger acceleration curves and structural deformation data. Based on these values, survivability ratings can be calculated during post-processing, similar to the traditional finite element simulation process.

3.1. Substructuring

A typical fuselage section, as illustrated in Figure 2, already encompasses numerous design variables, making it impractical to construct a single comprehensive surrogate model. In contrast, the presented approach divides the structure into smaller regions — termed substructures — each representable by an individual surrogate model. These substructures can subsequently be assembled to model the complete fuselage structure. Figure 4 illustrates this approach through the substructuring of a typical fuselage section. Each circle on the right hand side represents a connection between different surrogate models, which are depicted as straight lines. The red-coloured line represents the vertical strut connecting the frame and passenger crossbeam. The structural integrity of the assembled model requires that the deformation at the connection points between the surrogate models remains consistent. Beyond this restriction, the deformation of the substructure can be represented entirely by surrogate models, if contact interactions are not taken into account.

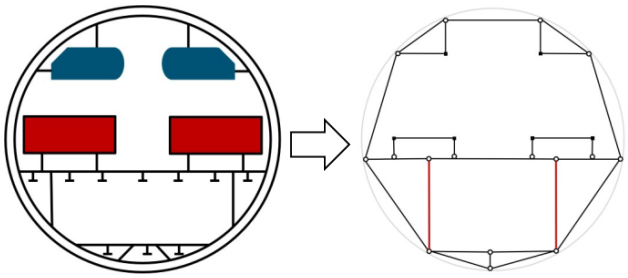


FIG 4. Substructuring approach

Substructuring alone does not accelerate the computational process but is essential for the presented surrogate model implementation. An additional benefit lies in the capability to create models simulating only specific substructure behaviour. Considering the vertical strut example, Figure 5 presents a schematic model encompassing partial frame and crossbeam sections alongside the vertical strut. The strut integration mirrors that of the complete model, enabling representation of all design variables of that substructure through this reduced model. Applying appropriate boundary conditions leads to similar crash behaviour, so that the results can be used for training surrogate models. This speeds up the training process, as the reduced model contains significantly fewer mesh elements than the complete model in Figure 2 when using the same element sizes, and the crash simulation can therefore be computed much faster.

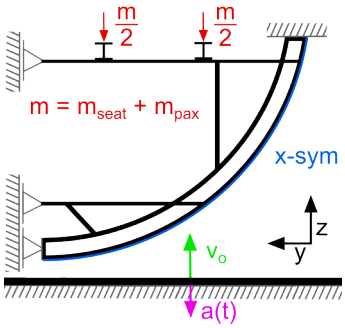


FIG 5. Integrated vertical strut model

3.2. Model Order Reduction

One approach to accelerate finite element simulations involves reducing the number of degrees of freedom through data-driven reduction techniques. Previously computed finite element simulations are utilised to identify the principal characteristics of the system. The dynamic characteristics are extracted with an unsupervised learning technique: Proper orthogonal decomposition. The N extracted deformation vectors $\boldsymbol{u} \in \mathbb{R}^n$ from all previous simulations are combined in a snapshot matrix $\boldsymbol{A} \in \mathbb{R}^{nxN}$, with each vector representing a snapshot of the dynamic system. To prevent the matrix from being dominated by vectors of large deformation, it is advantageous to normalise the snapshots beforehand. Through singular value decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Z} \approx \boldsymbol{V}\boldsymbol{\Sigma}_k\boldsymbol{Z}_k^T \tag{1}$$

the matrix is decomposed. The diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{nxn}$ contains the singular values $\sigma_1, ..., \sigma_n$, inherently sorted by magnitude. The presumption is that these values represent the significance of the information contained in the corresponding eigenmodes $\boldsymbol{U}$. Equivalently, they can be interpreted as the proportion of total system variance represented by each eigenmode. Consequently, only a reduced rank k is required to represent the majority systems variance. Thus, choosing $k << n$ substantially reduces the degrees of freedom needed to describe dynamic behaviour. The new degrees of freedom following POD are scalar factors $\boldsymbol{u}_r$ for each of the modes. The original displacement and velocity vectors

$$\boldsymbol{u} \approx \boldsymbol{V}\boldsymbol{u}_r \tag{2}$$

$$\dot{\boldsymbol{u}} \approx \boldsymbol{V}\dot{\boldsymbol{u}}_r \tag{3}$$

can be directly calculated by multiplying with the reduced basis $\boldsymbol{V} \in \mathbb{R}^{nxk}$.

A fundamental principle underlying model order reduction is the distinction between offline and online phases [14, 15]. During the offline phase, the dataset is constructed, the reduced basis is calculated, and other computationally intensive operations are performed. This enables acceleration of the online phase, where users can obtain model results by simply applying the model.

Methods employing reduced-order systems are categorised as either intrusive or non-intrusive by Bach et. al. [14] and Czech et. al. [15]. In mechanical systems, intrusive methods require access to the full-order model (FOM), specifically in FE application to the finite element solver, whereas non-intrusive methods are entirely data-driven. The presented methodology naturally employs a completely data-driven approach. So users do not need access to an FE solver.

The fundamental partial differential equation of the dynamic FE system

$$\boldsymbol{M}\,\ddot{\boldsymbol{u}} + \boldsymbol{K}\,\boldsymbol{u} = \boldsymbol{f}_{ext} \tag{4}$$

includes the mass matrix $\boldsymbol{M} \in \mathbb{R}^{nxn}$, stiffness matrix $\boldsymbol{K} \in \mathbb{R}^{nxn}$, both depending on the displacement $\boldsymbol{u}$ in a nonlinear system, and the external forces f_{ext}. For both categories it is projected onto the latent space, spanned by the reduced basis

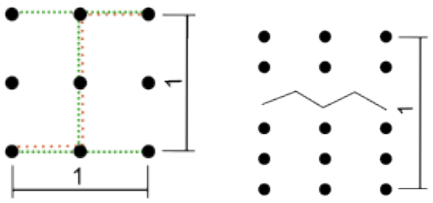
$$\boldsymbol{V}^T\boldsymbol{M}\boldsymbol{V}\,\ddot{\boldsymbol{u}}_r + \boldsymbol{V}^T\boldsymbol{f}_{int} = \boldsymbol{V}^T\boldsymbol{f}_{ext} \tag{5}$$

with the internal forces $f_{int} = \boldsymbol{K}\boldsymbol{u}$, as is also done by Czech et. al. [15] and Lülf et. al. [16]. In intrusive schemes, forces are calculated within the full-order model and subsequently

projected onto the latent space, where Equation 5 is solved. Bach et. al. [14] employ hyper-reduction, a technique where the internal force is only calculated at selected nodes whilst approximating it at the remaining nodes. It reduces the expensive FE solver calculations, leading to a significant computational time reduction. However, this also requires access to the core of the solver, as only the internal forces are to be calculated. For commercial solvers, established in industry, such access cannot generally be assumed.
Conversely, solver calls can be completely avoided by using non-intrusive methods, resulting in considerable time savings. The degrees of freedom of the latent space $\boldsymbol{u}_r$ are predicted with a regression model. Training such a regression model demands more computational effort than only the POD calculations, but this investment is worthwhile when the rapid model evaluation is required in the application. Czech et. al. [15] show both the intrusive method from Bach et. al. [14] and a completely data-driven approach. The computational time for non-intrusive model evaluation is two orders of magnitude lower than hyper-reduced methods, making them highly attractive for the presented application.
The reason why the method of Czech et. al. [15] is so fast, is that the degrees of freedom $\boldsymbol{u}_r = f(t, \mu)$ of the reduced system of Equation 2 are directly predicted by a machine learning model based on the time t and model parameters μ. However, using time as a model input is not feasible in preliminary design applications. This is because substructures may be applied in various configurations (e.g. different positions, surrounding structures), all directly influencing substructure temporal behaviour. The deformation is therefore very different depending on all of the model parameters. Including all model parameters as inputs for each substructure surrogate model undermines the fundamental concept of substructuring. A neural ODE approach is introduced later in this paper to avoid this.

3.3. Generalisation

Before introducing the neural network architecture, a fundamental consequence of employing mesh-based model order reduction techniques must be addressed. The snapshot matrix in Equation 1 comprises displacement snapshots of dimension n. The number of degrees of freedom in the full-order model must stay the same and must match identical nodal positions. This makes the whole approach mesh-dependent.
This requirement conflicts with the objective of representing diverse geometries using a single surrogate model, particularly considering that iterative preliminary design processes inherently involve geometric modifications at each iteration. A generalised point cloud is introduced to circumvent this limitation. This cloud remains invariant across all geometries whilst extracting data from different mesh configurations. Since only field data, such as displacement, is required for the reduced basis construction and subsequent neural network training, a straightforward mapping strategy can transfer data from finite element mesh nodes to the generalised point cloud.
The point cloud geometry must represent a generalised version of the actual meshes. The strut point cloud concept is illustrated in Figure 6. Each cross-section comprises nine points, enabling accurate resolution of typical strut profiles including H-, Z-, C-, or T-shapes. Cross-sections are stacked longitudinally, with outer dimensions normalised to unit length. During displacement transfer and result evaluation, scaling is applied according to the actual strut's geometric parameters.



(a) Top view (b) Side view

FIG 6. Point cloud of generalised struts

Deformation interpolation between different meshes is a well-established problem in aeroelasticity, see Barnewitz and Stickan [17] and Beckert and Wendland [18]. Existing concepts can be adapted for the presented methodology. For each time step and direction (x, y, z) a radial basis function is constructed that describes the displacement at point x in space using the linear combination:

$$u(x) = \sum_{i=1}^{n} \alpha_i \Phi(||x - x_i||) \tag{6}$$

The basis function $\Phi(||x - x_i||$ depends on the distance between the point x and all base points x_i. The Euclidian norm is the distance norm in this work. The base points x_i correspond to the mesh nodes of the actual finite element mesh where displacement is known. The parameters α_i are calculated by inverting Equation 6 and solving a system of linear equations

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{b} \tag{7}$$

with $\boldsymbol{A} = \Phi(||x_i - x_j||)_{(i,j)=1,\dots,n}$, $\boldsymbol{\alpha} = (\alpha)_{i=1,\dots,n}$ and $\boldsymbol{b} = f(x_i)_{i=1,\dots,n}$. Several functions can be used for the radial basis function Φ. A common one for the transfer of displacements is the *thin plates spline* [17, 18]:

$$\Phi(||x_i - x_j||)_{TPS} = ||x_i - x_j||^2 ln(||x_i - x_j||) \tag{8}$$

A very simple, but working solution is to use directly the norm as a linear function without any addition:

$$\Phi(||x_i - x_j||)_{linear} = ||x_i - x_j|| \tag{9}$$

To solve the linear system, it is necessary to invert the matrix $\boldsymbol{A}$, which has a significant computational cost, particularly because it has to be done three times for each snapshot. The computational expense scales with third power of the number of points [17], making mesh point reduction beneficial.
Base point reduction methods, as described in Barnewitz and Stickan [17], can achieve this objective. A straightforward approach involves equidistant reduction, as illustrated in Figure 7. In this iterative approach, all points with a distance less than a critical distance d_{crit}, are rejected as base points. The critical distance d_{crit} is optimised iteratively until the desired number of base points are selected. This calculation needs to be performed only once for each mesh, but the computational gain can be utilised three times the amount of the mesh's snapshots.
The generalisation process transforms the finite element meshes into a common representation. During the online phase, the model can be used without creating a detailed finite element model. For struts, longitudinal dimensions can be extracted from the global finite element model, while cross-sectional parameters are obtained from stored pro-

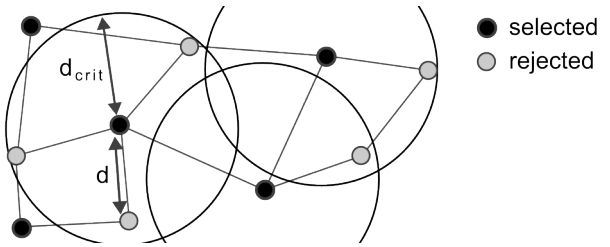


FIG 7. Base point selection with equidistant reduction technique

files. These parameters enable point cloud scaling to match the actual dimensions of the struts. This completely eliminates the FE model generation step in the online phase. This speeds up the process from Figure 3 at the second bottleneck.

3.4. Neural Differential Equation

The generalisation step results in loss of mesh connectivity information, preventing transfer of stiffness and mass matrices from Equation 4 without employing complex interpolation schemes. Conversely, field data, in this case the displacement, can be projected straightforwardly using radial basis functions. Consequently, constructing a model based exclusively on displacement data would be highly advantageous. An additional requirement involves implementing an integration scheme, as explained in subsection 3.2, such that time is not a parameter for the underlying model. This approach enables utilisation of the substructuring methodology, where substructures can be solved independently whilst sharing only specific degrees of freedom.
Within the project, various approaches were investigated, including reconstruction of the projected mass and stiffness matrices from Equation 5. Whilst this approach proved successful for simple truss models, it failed for complex crash simulation models.
This paper introduces a more direct methodology, based on NeuralODEs, first described by Chen et. al. [19]. Unlike classical residual neural networks that model discrete variable changes, NeuralODEs model the derivative:

$$\frac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}(t), t, \theta) \tag{10}$$

This formulation constitutes an initial value problem solvable using a black-box differential equation solver. The initial value of the hidden state $\boldsymbol{h}(0)$ must be specified, whilst the neural network parameters θ are optimised in the training process. As previously stated, time should not serve as a model parameter in the presented methodology. Therefore, function f in Equation 10 is constructed in such a way that temporal dependencies are excluded. The continuous formulation offers the additional advantage of handling arbitrary data series, whereas recurrent networks require identical temporal data points across all incorporated data [19].
Equation 10 is now adapted to address the second-order differential finite element problem described in Equation 4. The state $\boldsymbol{h}$ must comprise displacement $\boldsymbol{u}$ and its first derivate - velocity $\dot{\boldsymbol{u}}$. This fundamental structure is well-established and has been previously employed in acceleration prediction neural networks by Omar et. al. [20]. FE crash simulations also incorporate multiple units that describe material nonlinearities, including plasticity and damage. The surrogate model must learn this behaviour. Consequently, the hidden state h is expanded. These additional values will also be integrated and are part of the neural network architecture, despite no training data is available for them.
On the right side of Equation 10, the geometric parameters must also be included. These parameters are known from the generalisation step and remain temporally constant, eliminating the need for integration. Finally, Equation 10 is modified to:

$$[\dot{\boldsymbol{u}}_r(t), \ddot{\boldsymbol{u}}_r(t), \dot{\boldsymbol{h}}(t)] = f(\boldsymbol{u}_r(t), \dot{\boldsymbol{u}}_r(t), \boldsymbol{h}(t), p, \theta) \tag{11}$$

The derivate side (left) of the second order system comprises velocity $\dot{\boldsymbol{u}}_r$ and acceleration $\ddot{\boldsymbol{u}}_r$ in the latent space plus the derivate of the hidden state $\dot{\boldsymbol{h}}$. These values are calculated using a neural network, which takes displacement $\boldsymbol{u}_r$, velocity $\dot{\boldsymbol{u}}_r$, hidden state $\boldsymbol{h}$, as well as the model parameter p, as inputs. These calculations form the core of the entire process, as shown in Figure 8. The additional input parameters of the graphic are explained in the boundary conditions section.
When damping is neglected, as in Equation 4, velocity can be forwarded directly. In this case, the neural network only needs to calculate the derivatives of acceleration and the hidden state. This is the case for all crash models in this study.

3.5. Boundary Conditions

Examining Figure 4, the only boundary condition is contact with the ground. However, each connection point must remain attached to the connecting substructures. From the substructure's perspective, the connection node functions as a boundary condition during training. The connection node's displacement is prescribed temporally, and the substructure responds to this displacement as it represents the sole external influence. In the actual mesh, each connection node comprises multiple mesh nodes, as highlighted in Figure 9 - hereinafter referred to as connection zone. In FE models, connections between parts are modelled as tie constraints, which ensure that the involved degrees of freedom are displaced compatibly.
In the presented methodology, these nodes are not described exclusively by a single substructure, but are also handled independently. An independent POD is performed on all nodes involved in a connection zone belonging to different substructures.
During the training of a surrogate model for one substructure, the latent degrees of freedom of the connection nodes can act as a boundary condition. Another training process is then required to describe the displacement of the connection zone depending on the displacement of the connecting substructures. However, this aspect is beyond the scope of this abstract. The degrees of freedom of the connection zone are therefore described by two models. They are part of the substructure and the connection description. In the application, where the latent degrees of freedom are transformed back to the full order space, the connection degrees of freedom are not visible at all, since all nodes of the model are described via the substructures. Therefore, the boundary condition modes form a separate part of the equation.
For training one subsection using faster/smaller FE simulations of a certain region of the fuselage, Equation 11 is expanded by the boundary condition modes $\boldsymbol{u}_{bc}(t)$, which are described independently of the differential equation:

$$[\dot{\boldsymbol{u}}_r, \ddot{\boldsymbol{u}}_r, \dot{\boldsymbol{h}}] = f(\boldsymbol{u}_{bc}, \boldsymbol{u}_r, \dot{\boldsymbol{u}}_r, \boldsymbol{h}, p, \theta) \tag{12}$$

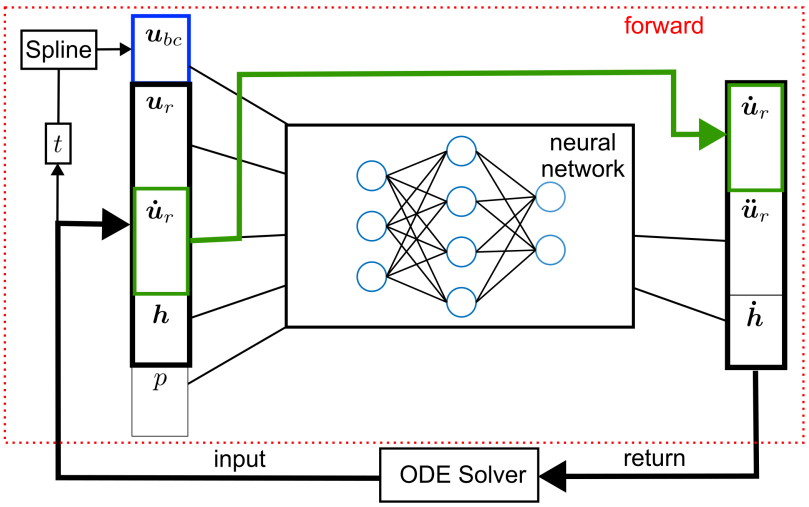
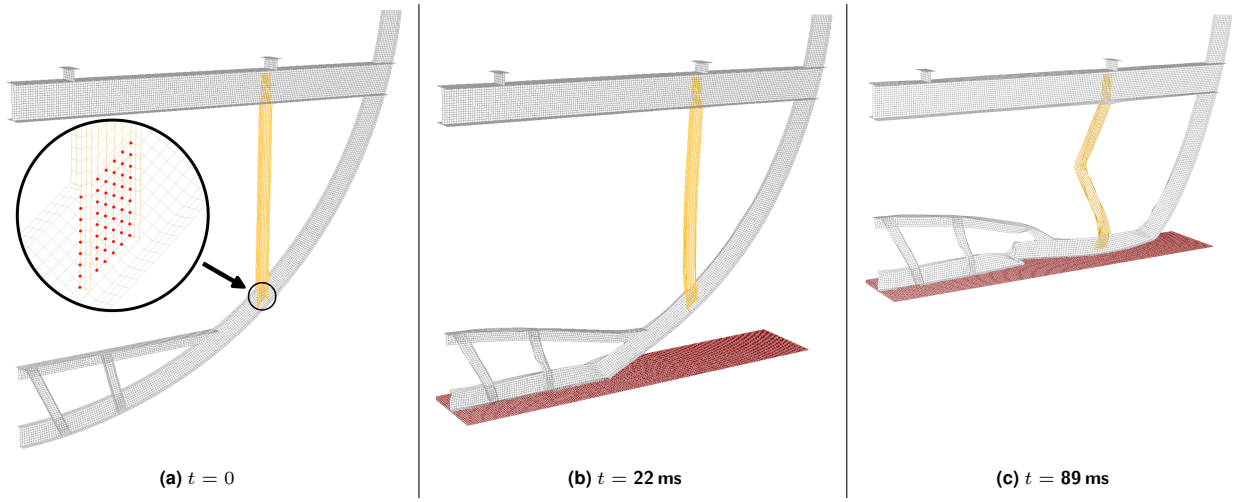


FIG 8. Architecture of the neural differential equation integration including the state $u_r, \dot{u}_r, h$, model parameters p and boundary condition modes u_{bc}



(a) $t = 0$ **(b)** $t = 22$ ms **(c)** $t = 89$ ms

FIG 9. Snapshots of the vertical strut FE crash simulation for surrogate model training. Tie-constraint nodes of the vertical struts connection zone are highlighted in (a)

The fundamental idea behind a neural differential equation is to create a continuous system. The boundary conditions also need to fulfil this requirement in order to be applicable. Currently, only snapshots of the boundary modes are available, which are calculated using a POD of the boundary degrees of freedom. There are many possible techniques for interpolating between the snapshots, including machine learning techniques such as Gaussian processes. However, in the current state, splines are used. One difficulty is that the interpolation must run alongside the neural network model on the computer's GPU for effective training. Standard Python modules such as SciPy are therefore not feasible. For splines an effective library *torchcubicspline* by Kidger [21] is freely available.

3.6. Neural Network Architecture

The setup is implemented in a Python program using the *torchdiffeq* library by Chen [22], which fully supports the describe approach and makes use of machine learning library PyTorch, enabling parallel evaluation of multiple trajectories. A visualisation of the architecture can be seen in Figure 8.

The state is represented as a one-dimensional vector, with displacement, velocity, and hidden state components arranged sequentially. In the model's forward pass, Equation 11 must be solved. As previously stated, velocity is directly forwarded, consequently, the neural network only needs to calculate the remaining components of Equation 12.

Calculations are performed using a feedforward neural network, currently equipped with fully connected layers. The neural network output and forwarded velocity can be integrated using any solver available in *torchdiffeq*. During the evaluation phase, any compatible solver may be employed.

3.6.1. Training of the Neural Network

During training, Figure 8 illustrates the forward pass. A predicted trajectory $\boldsymbol{u}_{r,pred}$ constitutes the forward run output, whilst the true trajectories are extracted from finite element simulations. It has been determined that it is beneficial to calculate the loss based on the decoded data $\boldsymbol{u}_{pred} = V\boldsymbol{u}_{r,pred.}$. The mean squared loss

$$L = \frac{1}{n}(\boldsymbol{u}_{pred} - \boldsymbol{u}_{true})^T(\boldsymbol{u}_{pred} - \boldsymbol{u}_{true}) \tag{13}$$

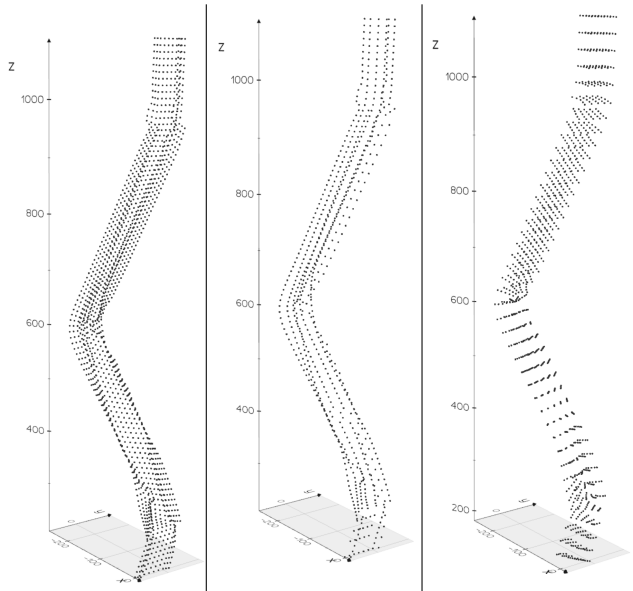
is used. This represents the mean loss across all simulation snapshots and all degrees of freedom in the generalised model.
Training a neural network involves calculating the derivative of the loss with respect to each parameter in the network θ. This derivative is then used to optimise the parameters. In this *backpropagation* step, the loss derivate must be propagated through all all solver operations, since the solution of time step t_n depends on the deformation of time step t_{n-1} in the recurrent formulation. Chen et. al. [19] highlight that this straightforward approach "incurs a high memory cost and introduces additional numerical error" [19, p. 2]. This remains feasible for simpler solvers such as Euler or Dopri5 [22]. For more complex solver Chen et. al. [19] suggest an adjoint sensitivity method from Semyonovich Pontryagin [23] to calculate the loss derivates. In this approach, a second differential equation is solved backwards in time. This method "scales linearly with the problem size, has low memory cost, and explicitly controls numerical error" [19, p. 2]. It is worth mentioning that a solution to this issue exists and is compatible with the implemented process.
It has been found that scheduling both the learning rate and the solver during training is beneficial. Starting with a fast, explicit solver, such as the Euler solver, accelerates training, as this is sufficient to guide the model in the correct direction. Fine-tuning can then be performed using a higher-quality solver. This flexibility is a key advantage of using neural ordinary differential equations.

4. DEMONSTRATION CASE

A major advantage of the substructuring approach is that substructures can be defined by significantly fewer design variables than the complete system. This principle extends to the surrogate training process. Finite element simulations need not to encompass a fully assembled fuselage but rather the substructure and its surrounding components. However, selecting an appropriate simplification for the integration remains a challenging task.
In this work, the vertical strut, connecting the frame and passenger crossbeam, serves as the demonstration substructure. During a crash event, the cargo floor initially deforms, followed by frame deformation directly beneath the vertical strut connection, where the strut provides structural stiffening as visualised by Waimer et. al. [24]. When the connection area impacts the ground, the vertical strut buckles and deforms. The simplified structural model must accurately represent this behaviour to ensure the surrogate learns the correct response characteristics.
Figure 5 illustrates the simplified structure configuration. The upper boundary nodes of the frame are constrained in all degrees of freedom. Since a half-model approach is employed, nodes at the xz-plane are subject to y-symmetry conditions. The skin is omitted from this model as it lacks direct connection to the vertical strut. However, since the skin prevents frame movement in the depth direction, a x-symmetry constraint is applied to the frame's outer flank nodes to simulate this effect. An elastic-plastic aluminium material model is utilised for all structural components.
Unlike actual accident scenarios, the movement is introduced by an upward movement of the ground. A total of 30 different load cases have been simulated, with initial velocities of $30\,\mathrm{ft\,s^{-1}}$ and $25\,\mathrm{ft\,s^{-1}}$. The ground subsequently decelerates with constant acceleration, ranging from $0.5\,\mathrm{g}$ to $12.5\,\mathrm{g}$. Snapshots of the displacement and velocity of all the vertical strut nodes are extracted every $0.2\,\mathrm{ms}$ throughout each simulation, yielding a total of 29,692 data points. The snapshots shown in Figure 9 prove that the described typical sequence of events is accurately represented by this FE model.

4.1. Generalisation

Figure 10 demonstrates the generalisation capabilities of the described methodology by visualising the final displacement from the simulation shown in Figure 9. Subfigure (a) displays the 1,547 displaced finite element mesh nodes. In Figure 10 (b), the number of mesh nodes is reduced to 750 nodes using the equidistant reduction method. Figure 10 (c) shows the displacement from (b) projected onto the scaled generalised point cloud.



(a) FE mesh nodes (b) reduced FE mesh nodes (c) scaled point cloud

FIG 10. Generalisation of the vertical struts deformation at $t = 89\,\mathrm{ms}$

As can be seen, the scaled point cloud approach is able to map the characteristics well. One advantage of the described methodology is that the number of points in the generalised model is not decisive for the speed of the model, as these points are used as the basis for further dimension reduction, which leads to the latent space in which the problem is solved. This means that there is no need to weigh up accuracy and speed, when deciding about the number of points in the cloud.

4.2. Dimensional Reduction

The dataset from the 30 simulations is divided into training, validation, and test sets. The first two datasets are included in the POD analysis, whilst the test set remains completely unknown to the system. The normalised snapshot matrix for

the vertical strut comprises 46,854 snapshots of displacement and velocity, each containing 4,850 degrees of freedom.
Figure 11 illustrates the cumulative variance of the POD modes. The first mode alone captures 42 % of the system's variance, whilst ten modes represent more than 95 % of the variance, which is considered sufficient for this application.
The same analysis is performed for the tie-constraint region of the strut, which connects the strut to both the frame (see Figure 9 (a)) and the crossbeam. As shown in Figure 11, 95 % of the variance in these boundary displacements can be represented using only six modes.

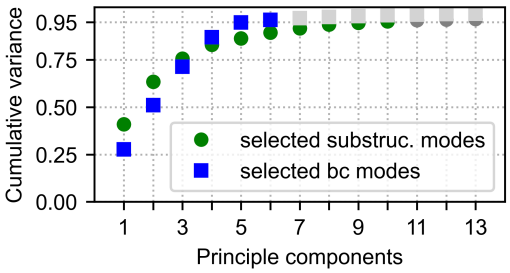


FIG 11. Cumulative variance of the principle components of the POD, representing the importance of the underlying mode

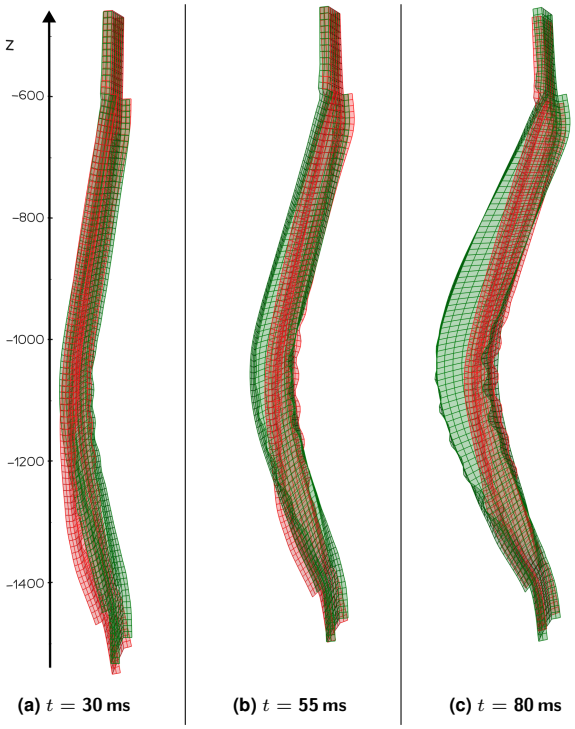


FIG 12. Test Case: Prediction of the trained neuralODE (green) vs. the true displacement extracted from FE simulation (red) at three different times

4.3. Solving the NeuralODE

Since the trajectories originate from different simulations but are derived from the same FE model, there is no need to include geometric parameters p in the network, which simplifies the setup by one further step. However, the different load cases lead to different plastic behaviour of the structure, so the hidden parameters are included to give the network the ability to learn plasticity growth. In addition to a total of 20 states from displacement and velocity, eight more were added. The neural network thus receives 18 degrees of freedom plus the six boundary condition variables as input.
The neural network consists of a total of ten hidden layers and expands the input tensor by a factor of 5, so that the hidden layers are of size 90. The Swish function (SiLu) was used as the activation function.
The network is trained with 18 of the 30 FE simulations. A learning rate scheduler is applied using the validation data set. In each epoch, all 18 training trajectories are predicted. Figure 12 shows the prediction of the trained model for a load case that was not considered in training. The sequence shows that the general behaviour is represented by the surrogate model. However, particularly in video animations, it can be seen that the dynamics are not predicted sufficiently and that the surrogate model always lags behind. Since the model only receives the displacement as training data at this stage, the dynamics must be implicitly learned. However, since the error is based solely on the displacement, this is not achieved sufficiently.
Taking velocity into account in training could prevent this problem, as it would then be directly included in the error. In this case, clever weighting must be applied to the error term so that it is not dominated by either of the two factors.
In terms of time savings, it should be noted that a forward pass calculation is almost instantaneous, making it ideal for use in iterative optimisation programmes. While the presented case only considers one substructure, it is anticipated that modelling the entire fuselage structure will produce similarly positive results. Compared to FE simulations, this is therefore expected to result in significant time savings.

5. CONCLUSIONS

This paper has presented a novel methodology for enabling efficient crashworthiness analyses in aircraft preliminary design through the combination of substructuring, mesh generalisation, proper orthogonal decomposition-based model order reduction, and neural differential equations into one process. The approach addresses the fundamental challenge of incorporating computationally expensive crash simulations into iterative design processes by developing non-intrusive surrogate models that operate in latent space. The substructuring methodology decomposes complex fuselage structures into manageable components, each representable by individual surrogate models. This decomposition enables individual training of surrogates. The mesh generalisation technique using radial basis functions effectively decouples the surrogate models from specific mesh configurations, allowing representation of diverse geometries within a single framework.
The demonstration case of the vertical strut surrogate illustrates the methodology's capability to capture essential crash behaviour characteristics. The POD-based dimensional reduction achieves substantial compression, with ten modes representing 95 % of the system variance. The NeuralODE implementation successfully learns the underlying physics without requiring explicit time parameterisation, making it suitable for the substructuring methodology. However, several limitations have been identified that warrant future investigation. The current approach exhibits dynamic lag, indicating insufficient learning of temporal characteristics when training is based solely on displacement data.

Incorporating velocity information into the training process offers a promising solution to address this deficiency. Additionally, the methodology requires further validation across diverse crash scenarios and structural configurations to establish its broader applicability.
The presented framework provides a foundation for integrating crashworthiness considerations into preliminary aircraft design processes. While the demonstrated vertical strut case shows promise in terms of capturing essential crash behaviour, significant further development is required before practical implementation. Within the ongoing HYFLIP project of TU Braunschweig, future work focuses on addressing the identified dynamic limitations through improved training strategies, developing models to connect the substructure surrogates, and expanding the methodology to incorporate additional substructure types to build towards complete fuselage section representation.

Acknowledgement

This paper has been created as part of the HYFLIP (grant 20E2218B) project of the German Federal Ministry for Economic Affairs and Energy under aegis of the ”Luftfahrtforschungsprogramm LuFo VI". We would like to take this opportunity to thank the ministry for its financial support.

Contact address:

henning.dahmen@tu-braunschweig.de

LITERATURE

[1] S. Heimbs, F. Strobl, and P. Middendorf. Integration of a composite crash absorber in aircraft fuselage vertical struts. *International Journal of Vehicle Structures & Systems* 3.2 (2011), pp. 87–95. DOI: 10.4273/IJVSS.3.2.03.

[2] EASA. Certification Specifications for Large Aeroplanes - Amendment 28. Dec. 19, 2023.

[3] M. Guida, G. Lamanna, F. Marulo, and F. Caputo. Review on the design of an aircraft crashworthy passenger seat. *Progress in Aerospace Sciences*. Impact Induced Dynamic Response and Failure Behavior of Aircraft Structures 129 (Feb. 2022), p. 100785. DOI: 10.1016/j.paerosci.2021.100785.

[4] A. M. Eiband. Human tolerance to rapidly applied accelerations: A summary of the literature. NASA Memorandum (MEMO) E-345. NASA Lewis Research Center Cleveland, OH United States, June 1959.

[5] W. Heinze. *Ein Beitrag zur quantitativen Analyse der technischen und wirtschaftlichen Auslegungsgrenzen verschiedener Flugzeugkonzepte für den Transport großer Nutzlasten*. PhD thesis. TU Braunschweig, 1994.

[6] W. Heinze, C. M. Österheld, and P. Horst. Multidisziplinäres Flugzeugentwurfsverfahren PrADO-Programmentwurf und Anwendung im Rahmen von Flugzeug-Konzeptstudien. *Jahrbuch der DGLR-Jahrestagung* (2001).

[7] M. Alder, J. Jepsen, and B. Nagel. Recent advances in establishing a common language for aircraft design with CPACS. Aerospace Europe Conference 2020. Bordeaux, France, Feb. 2020.

[8] A. Bachmann, M. Kunde, and M. Litz. Tool integration and data formats for distributed airplane predesign. ModelCenter European Users Workshop. Braunschweig, Germany, Oct. 2008.

[9] D. B. Schwinn. Parametrised fuselage modelling to evaluate aircraft crash behaviour in early design stages. *International Journal of Crashworthiness* 20.5 (2015), pp. 413–430. DOI: 10.1080/13588265.2015.1022435.

[10] M. Petsch, D. Kohlgrüber, and J. Heubischl. PANDORA - A Python based framework for modelling and structural sizing of transport aircraft. *MATEC Web of Conferences*. Vol. 233. Glasgow, United Kingdom: EDP Sciences, Nov. 2018, p. 00013. DOI: 10.1051/matecconf/201823300013.

[11] J.-N. Walther, C. Hesse, J. Biedermann, and B. Nagel. Extensible aircraft fuselage model generation for a multidisciplinary, multi-fidelity context. 33rd Congress of the International Council of theAeronautical Sciences (ICAS). Stockholm, Sweden, Sept. 2022.

[12] J.-N. Walther. *Knowledge-based engineering to provide aircraft fuselage design details for multidisciplinary and multifidelity analysis model generation*. PhD thesis. TU Berlin, 2023.

[13] C. Hesse, P. Allebrodt, M. Teschner, and J. Biedermann. Knowledge-based model generation for aircraft cabin noise prediction from pre-design data. *CEAS Aeronautical Journal* 15.4 (2024), pp. 1127–1136. DOI: 10.1007/s13272-024-00769-z.

[14] C. Bach, L. Song, T. Erhart, and F. Duddeck. Dimensionality reduction of crash and impact simulations using LS-DYNA. 12th European LS-DYNA Conference. Koblenz, Germany, 2019.

[15] C. Czech, M. Lesjak, C. Bach, and F. Duddeck. Data-driven models for crashworthiness optimisation: intrusive and non-intrusive model order reduction techniques. *Structural and Multidisciplinary Optimization* 65.7 (June 2022). DOI: 10.1007/s00158-022-03282-1.

[16] F. A. Lülf, D.-M. Tran, H. G. Matthies, and R. Ohayon. An integrated method for the transient solution of reduced order models of geometrically nonlinear structures. *Computational Mechanics* 55.2 (Feb. 2015), pp. 327–344. DOI: 10.1007/s00466-014-1103-4.

[17] H. Barnewitz and B. Stickan. Improved mesh deformation. *Management and Minimisation of Uncertainties and Errors in Numerical Aerodynamics: Results of the German Collaborative Project MUNA*. Berlin, Heidelberg: Springer, 2013, pp. 219–243. DOI: 10.1007/978-3-642-36185-2_9.

[18] A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerospace Science and Technology* 5.2 (2001), pp. 125–134. DOI: 10.1016/S1270-9638(00)01087-7.

[19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*. Vol. 31. Montreal, Kanada: Curran Associates, Inc., Dec. 2018.

[20] T. Omar, A. Eskandarian, and N. Bedewi. Vehicle crash modelling using recurrent neural networks. *Mathematical and Computer Modelling* 28.9 (Nov. 1998), pp. 31–42. DOI: 10.1016/S0895-7177(98)00143-5.

[21] P. Kidger. Torchcubicspline. URL: https : / / github . com / patrick - kidger / torchcubicspline (visited on 09/12/2025).

[22] R. T. Q. Chen. Torchdiffeq. June 2021. URL: https : / / github . com / rtqichen / torchdiffeq (visited on 09/11/2025).

[23] L. Semyonovich Pontryagin. *The Mathematical Theory of Optimal Processes*. Interscience Publishers, 1962.

[24] M. Waimer, D. Kohlgrüber, R. Keck, and H. Voggenreiter. Contribution to an improved crash design for a composite transport aircraft fuselage — development of a kinematics model and an experimental component test setup. *CEAS Aeronautical Journal* 4.3 (Sept. 2013), pp. 265–275. DOI: 10.1007/s13272-013-0070-3.