Ref. Ares(2021)1386395 - 20/02/2021

FliPASED
WWW.FLIPASED.EU

D1.2 Requirements capture for a/c MDO design

Fanglin Yu (TUM), Muhammad Meddaikar (DLR-AE)
Thiemo Kier, Matthias Wuestenhagen, Simon Binder (DLR-SR)
Béla Takarics (SZTAKI), Charles Poussot-Vassal (ONERA)

GA number: 815058
Project acronym: FLIPASED
Project title: FLIGHT PHASE ADAPTIVE AERO-SERVOELASTIC AIRCRAFT DESIGN METHODS

Funding Scheme: H2020 **ID:** MG-3-1-2018
Latest version of Annex I: 1.1 released on 12/04/2019
Start date of project: 01/09/2019 **Duration:** 40 Months

Lead Beneficiary for this deliverable:	TUM
Last modified: 20/02/2021	**Status:** Delivered
Due date: 30/11/2020	

Project co-ordinator name, title and organisation: Bálint Vanek, SZTAKI
Tel: +36 1 279 6113
Fax: +36 1 466 7483
E-mail: vanek@sztaki.hu
Project website address: www.flipased.eu

Dissemination Level		
PU	Public	
CO	Confidential, only for members of the consortium (including the Commission Services)	X

Glossary

ASE	Aeroservoelastic
AFS	Active Flutter Suppression
CAD	Computer-aided Design
CPACS	Common Parametric Aircraft Configuration Schema
DLM	Doublet Lattice Method
FE	Finite Element
GLA	Gust Load Alleviation
LPV	Linear Parameter-varying
LPI	Linear Time-invariant
MDAx	MDAO Workflow Design Accelerator
MDO	Multidisciplinary Design Optimization
MIMO	Multi-Input Multi-Output
MLA	Manoeuvre Load Alleviation
PID	Proportional-Integral-Derivative
RCE	Remote Component Environment
ROM	Reduced Order Model
TCL	Tool Command Language
W3C	Wold Wide Web Consortium
XDSM	Extended Design Structure Matrix
XML	Extensible Markup Language
XSD	XML Schema Definition

Table of contents

List of Figures

1 Executive summary

In order to set up a collaborative design toolchain for an advanced, actively flight condition optimized wing design, requirements for the MDO toolchain need to be captured first. This deliverable documents outcomes of activities conducted for the requirement capture and serves as the top-level guideline for the subsequent MDO implementation.

This deliverable is organized as follows:

Chapter 2 introduces the objectives of the MDO toolchain and derived requirements. Two sorts of requirements are specified because of the different objectives for demonstrator wing design and commercial transport aircraft wing design. This chapter is contributed by Matthias Wuestenhagen(DLR-SR).

Chapter 3 provides an overview of the MDO toolchain. The MDO toolchain structure is visualized by MDAx, which is developed by DLR to support the ideation phase of MDO. The functions of individual blocks are specified. An introduction of the integration framework RCE is given here. This chapter is contributed by Simon Binder(DLR-SR), Fanglin Yu(TUM), Béla Takarics(SZTAKI), and Thiemo Kier(DLR-SR).

Chapter 4 defines interfaces of connected blocks in MDO toolchain. An introduction to CPACS, which is agreed by the consortium to serves as the standard interface medium, is also given here. This chapter is contributed by Thiemo Kier(DLR-SR), Fanglin Yu (TUM), Muhammad Meddaikar(DLR-AE), and Charles Poussot-Vassal(ONERA).

In chapter 5 a conclusion is given.

2 Objective functions of toolchain and requirements

It has to be distinguished between the MDO toolchain for the demonstration of technologies and the scale-up study.

For the demonstration of technologies including gust load alleviation (GLA), manoeuvre load alleviation (MLA), active flutter suppression and drag reduction through wing shape control the UAV demonstrator designed for FLEXOP with the flexible, flutter wing (-1) is considered as a starting point. Furthermore, the fuselage and V-tail of the demonstrator is fixed within the MDO toolchain. This reduces the design space significantly. The design freedom is limited to wing properties comprising

- the wing planform (sweep, taper ratio, span)
- the spar positions
- the skin stiffness
- the flap positions
- the pre-twist
- the position of the ribs.

The material choice of the wings can be optionally considered as well. The number of flaps per wing is fixed. However, three different designs with four, eight and sixteen flaps per wing are optimised in parallel. The focus of the final design is to demonstrate that loads and drag are reduced, while the flutter speed is increased, when including GLA, MLA, AFS and drag reduction through wing shape control into the MDO toolchain. Therefore, the objective function is selected to minimise the closed-loop drag D_{CL}, manoeuvre loads ML_{CL} and gust loads GL_{CL} with respect to the open-loop drag D_{OL}, manoeuvre loads ML_{OL} and gust loads GL_{OL}. The closed-loop flutter speed FS_{CL} is wanted to be increased relative to the open-loop flutter speed FS_{OL}. As a result the objective function is given by

$$J = a(D_{OL} - D_{CL}) + b(FS_{CL} - FS_{OL}) + c(ML_{OL} - ML_{CL}) + d(GL_{OL} - GL_{CL}), \tag{1}$$

where a, b, c and d represent positive weights to be determined. The overall task is to maximise the objective function J. As a constraint the open loop flutter speed FS_{OL} is bounded to a maximum of $50\,\mathrm{m/s}$ and the overall aircraft weight should not exceed $65\,\mathrm{kg}$. After a design solution is found, the new wings and technology will be built and demonstrated.

The scale-up task, which is supposed to provide an improved commercial aircraft design based on the technologies demonstrated with the unmanned demonstrator, focuses on a passenger aircraft. Different from the demonstrator, the wing design is driven by loads in contrast to technology demonstration purposes. There are two possible objective functions to drive the MDO process. Firstly, maximising the size - payload can be an objective. Secondly, minimising the block fuel can be an objective as well. Depending on the objective function the aircraft designs obtained by the MDO process differ. However, both ways provide an aircraft design with improved fuel consumption.

For the design of the demonstrator as well as for the passenger aircraft, it is required that the MDO toolchain works automatically. Therefore, the interfaces including the inputs and outputs of each block needs to be predefined. Another important part is that the given blocks provide the information needed for the solver to optimise the objective function. For example, when using a gradient-based solver each block of the MDO toolchain has to provide gradients with respect to the design parameters.

3 MDO toolchain

3.1 MDAx

Because of the inherent complexity of the multidisciplinary design optimization workflow, the first step of the development was the discussion and joint elaboration of the process and interfaces between the various disciplines. The DLR tool MDAx (MDAO Workflow Design Accelerator) was used to digitally support the ideation phase of the intended optimization workflow [7]. The tool processes XML files defined by integrators and disciplinary experts for each workflow component and enables the user to model, inspect and explore the components and their relationships. MDAx automatically establishes connections between all components with the supplied interface information. These process and data connections are then inspected to resolve parameter collisions and possible internal feedback connections that require convergence or optimizers. The tool allows the visualization of the MDO process and architecture in the form of an extended design structure matrix (XDSM), a diagram that is widely used by the community. The format simultaneously shows data dependency and process flow between the involved disciplines on a single diagram and thus allows more effective collaboration and discussion [5].

The current status of the intended workflow at the time of writing this deliverable is shown in Fig 1 in the form of an XDSM.

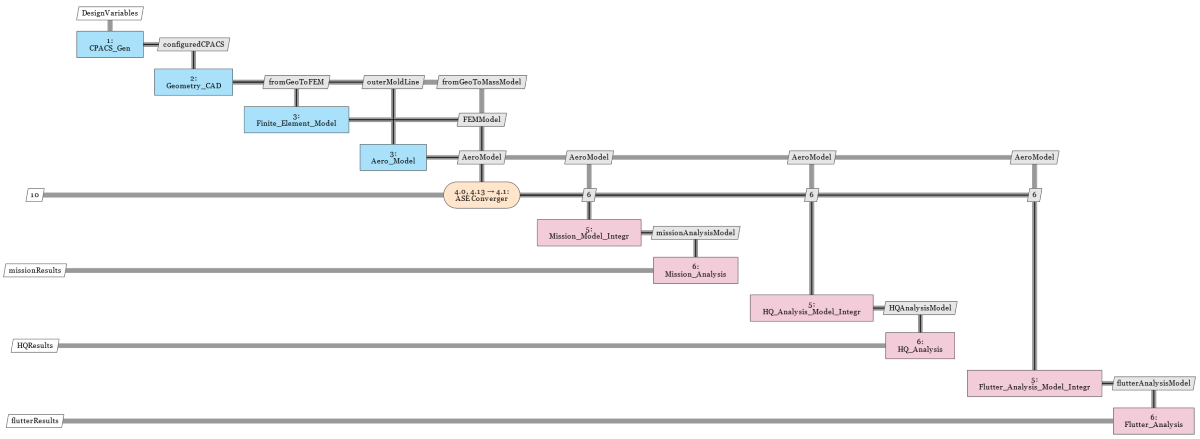

Figure 1: MDO workflow in XDSM format with collapsed ”ASE Converger”

The internal converger “ASE Converger” is shown in collapsed view for better readability in Fig 1. The expanded view is given in Fig 2.

3.2 Functions of blocks

3.2.1 CPACS generation block

CPACS generation block, which is the first block in the MDO toolchain, aims to generate the first version of CPACS file with a Matlab script. The inputs are all the geometry and structure related parameters, for instance, airfoil data, wing form, spar location, etc. The output of this block is a CPACS xml file.

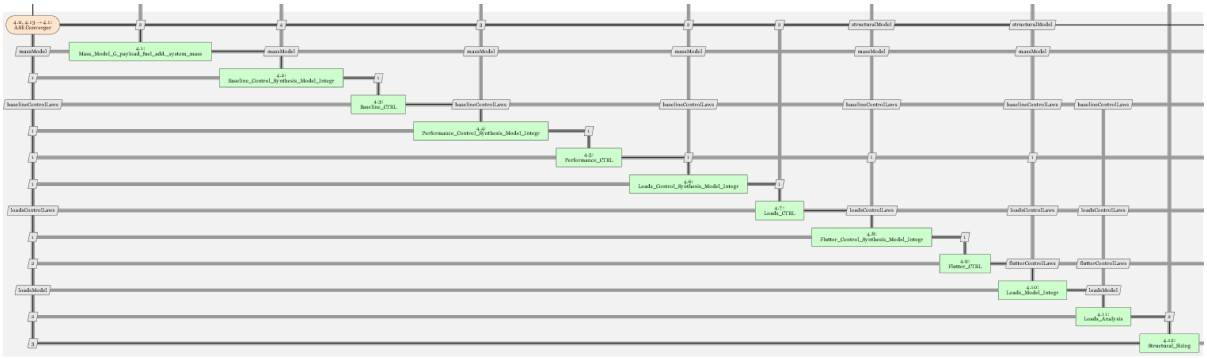

Figure 2: MDO workflow in XDSM format with extended ”ASE Converger”

3.2.2 Geometry block

Geometry block aims to update the Catia model based on the incoming CPACS file from the upstream CPACS generation block. A Macro script is used to track the parameter variation in CPACS file. The output of this block is a Catpart file.

3.2.3 FE-model block

The function of the FE-model block is meshing the geometry model and assigning structural properties. A Splining model, which couples the structural and aerodynamic model, is also generated in this block. The input of this block is Catpart file. A Nastran bdf file is exported as output.

3.2.4 Aero-model block

The aero-model block takes the geometry definition in CPACS file as input, generates the DLM aerodynamic model, and exports it to a Nastran bdf file.

3.2.5 Control oriented modeling

Modeling block inputs

The modeling block takes the structural dynamics (M_{hh}, K_{hh}, B_{hh}) and aerodynamics data (Q_{hh}) as input via CPACS.

Modeling block main algorithms

The control oriented models are based on the linear parameter-varying (LPV) framework, [8, 2]. The LPV framework can serve as a good approach to model aeroservoelastic (ASE) systems for control design. The benefits of utilizing the LPV framework are the following; it can capture the parameter varying dynamics of the aircraft and many of the linear time-invariant (LTI) control design techniques have been extended to LPV systems. An LPV system is described by the state space model [12, 8]

$$\dot{x}(t) = A(\rho(t))\ x(t) + B(\rho(t))\ u(t) \tag{2a}$$
$$y(t) = C(\rho(t))\ x(t) + D(\rho(t))\ u(t) \tag{2b}$$

with the continuous matrix functions $A\colon \mathcal{P} \to \mathbb{R}^{n_x \times n_x}$, $B\colon \mathcal{P} \to \mathbb{R}^{n_x \times n_u}$, $C\colon \mathcal{P} \to \mathbb{R}^{n_y \times n_x}$, $D\colon \mathcal{P} \to \mathbb{R}^{n_y \times n_u}$, the state $x\colon \mathbb{R} \to \mathbb{R}^{n_x}$, output $y\colon \mathbb{R} \to \mathbb{R}^{n_y}$ input $u\colon \mathbb{R} \to \mathbb{R}^{n_u}$, and a time-varying scheduling signal $\rho\colon \mathbb{R} \to \mathcal{P}$, where $\mathcal{P}$ is a compact subset of $\mathbb{R}^N$. The system is called quasi LPV model if the parameter vector ρ includes elements of the state vector x. The system matrix $S(\rho(t))$ is defined as

$$S(\rho(t)) = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix} \tag{3}$$

In a grid-based LPV representation ([12]), the system is described as a collection of LTI models $(A_k,\ B_k,\ C_k,\ D_k) = (A(\rho_k)\,, B(\rho_k)\,, C(\rho_k)\,, D(\rho_k))$ obtained from evaluating the LPV model at a finite number of parameter values $\{\rho_k\}_1^{n_{\text{grid}}} = \mathcal{P}_{\text{grid}} \subset \mathcal{P}$.

The main milestones of the modeling block are the following. The ASE model is formed by combining the structural dynamics model, the aerodynamics model and the flight mechanics model. In order to obtain an ASE model suitable for control design, model order reduction needs to be carried out. The model order reduction is based on the bottom-up modeling approach, [10, 6, 13].

The key idea of the bottom-up modeling is the following. The subsystems of the ASE model in general have simpler structure than the nonlinear ASE model. Therefore, the subsystems containing the structural dynamics and aerodynamics model can be reduced by simpler, more tractable reduction techniques. Combining these reduced order subsystems results in a low order nonlinear ASE model upon which a nominal, low order, control oriented models can be obtained. The main measure of the accuracy of the low order model is the ν-gap metric, [11]. The control design is using the linear parameter-varying (LPV) framework, [8, 2]. Therefore grid-based LPV models need to be obtained via Jacobian linearization.

It is worth noting that rigid body baseline controller design and flexible dynamics control might require different model setups, since trimming and linearization is sensitive for small deviations in control allocation matrix and the resulting LTI model might contain unwanted numerically ill-conditioned parts. For this reason the mathematical model might split into two branches even before linear models are obtained.

Reduction of the structural dynamics model

The structural dynamics of the aircraft are of the form

$$\mathcal{M}\ddot{\eta} + \mathcal{C}\dot{\eta} + \mathcal{K}\eta = F_{\text{modal}} \tag{4}$$

where F_{modal} is the force acting on the structure in modal coordinates, $\mathcal{M}$, $\mathcal{C}$ and $\mathcal{K}$ are the modal mass, damping and stiffness matrices respectively. The structural dynamics model is an LTI system, thus state truncation can be applied.

Reduction of the aerodynamics model

The aerodynamic lag terms take the state-space form

$$\begin{aligned} \dot{x}_{\text{aero}} &= \frac{2V_{\text{TAS}}}{\bar{c}} A_{\text{lag}} x_{\text{aero}} + B_{\text{lag}} \begin{bmatrix} \dot{x}_{\text{rigid}} \\ \dot{\eta} \\ \dot{\delta}_{\text{cs}} \end{bmatrix} \\ y_{\text{aero}} &= C_{\text{lag}} x_{\text{aero}} \end{aligned} \tag{5}$$

where V_{TAS} is the true airspeed, x_{rigid} is the rigid body state, η is the modal state of the structural dynamics, δ_{cs} is the control surface deflection and $\bar{c}$ is the reference chord. Using the aerodynamics

model given by A_{lag}, B_{lag} and C_{lag} in (5) an LTI balancing transformation matrix T_b is computed. The balanced states of the aerodynamic model with the smallest Hankel singular values are residualized, leading to a reduced order aerodynamics model.

The initial model order reduction produced the following results. The structural dynamics model can be reduced in the following way. In order to keep the ν-gap between the high fidelity and the low order model low the first six structural modes and modes 19, 20, 21 are retained for the reference aircraft model. Te removal of the latter results in a large increase in the ν-gap. This way, a 18 state structural dynamics model can be obtained from the 100^{th} order model. In case of the aerodynamics model, retaining two lag states results in a low order model with acceptable accuracy. The resulting nonlinear ASE bottom-up model has 32 states that consists of 12 rigid body states, 18 structural dynamics states, 2 aerodynamic lag states. Note, that the actuator dynamics are not included in the control oriented model. The ν-gap between the nominal, high-fidelity and the reduced order model for different airspeed values is given in Figure 3.

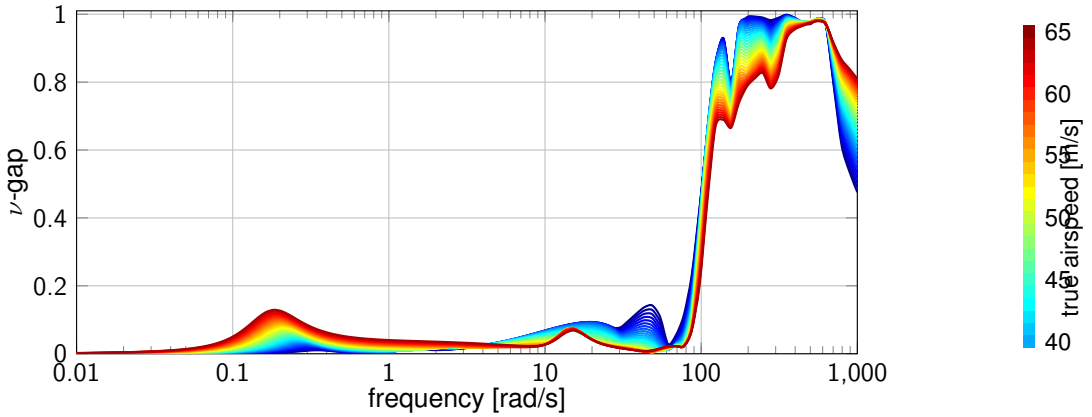

Figure 3: ν-gap values between the nominal low order and high-fidelity models.

Uncertain low order model

The next step is to develop uncertain LPV models of the aircraft. Uncertain models can be developed by extending the structural dynamics model with the uncertain parameters. These uncertainties appear in the mass matrix $\mathcal{K}$ and in the damping matrix $\mathcal{C}$ in (4) of the nonlinear ASE model and are denoted by $\delta_{\mathcal{K}}$ and $\delta_{\mathcal{C}}$, respectively. Based on this uncertain, nonlinear model a grid-based uncertain LPV model is constructed. The grid-based uncertain LPV model is obtained over a 3 dimensional grid. The grid consists of 81 equidistant points of the airspeed between 30 m/s and 70 m/s, 3 points of the natural frequency in the structural dynamics between $\pm 1\%$ of the nominal value, and 3 points of the damping in the structural dynamics between $\pm 10\%$ of the nominal value. This results in a total of $81 \times 3 \times 3 = 729$ grid points. The scheduling parameter ρ can then be defined as

$$\rho = \begin{bmatrix} \rho_{V_{TAS}} \\ \delta_{\mathcal{K}} \\ \delta_{\mathcal{C}} \end{bmatrix} \tag{6}$$

where $\rho_{V_{TAS}}$ is a measured parameter and $\delta_{\mathcal{K}}$ and $\delta_{\mathcal{C}}$ are unmeasured. These uncertainties have a significant effect on the flutter speeds and frequencies. The nominal and uncertain flutter modes of the control oriented LPV model are shown in Figure 4.

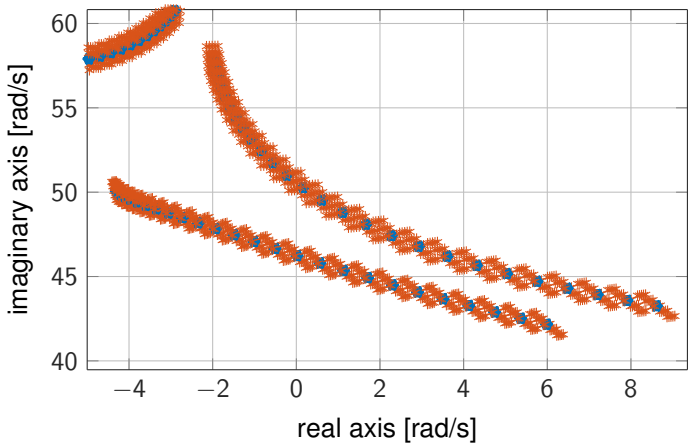

Figure 4: Uncertainty of the flutter modes: nominal model (blue), uncertain (red).

Modeling block robustness

As it can be seen, the bottom-up modeling approach involves a certain degree of heuristics. These heuristic steps include the selection of the structural dynamics states to retain and setting the the number of retained aerodynamic lag states. These parameters are hand tuned for the initial, reference aircraft model. The modeling tool needs to be adopted to the collaborative design in this respect. This means that the retained the initial structural modes to be retained are the ones of the reference aircraft. However, it is crucial that after every MDO iteration, the ν-gap metric is analyzed and that it does not exceed a threshold value. If this value is exceeded, it means that the bottom up-model is not accurate enough. Therefore, at the expense of increasing the order ot the resulting model, additional structural modes need to be retained. The number of retained modes is increased until the ν-gap values are satisfactory. A similar approach is used for the order of the lag state aerodynamics model. In this case the number of the retained lag states is increased until a satisfactory ν-gap level is obtained.

Modeling block outputs

The modeling block provides two models, one for the baseline control design (RigACModel) and one for the flutter control design (FlexACModel). The FlexACModel is the low order, uncertain LPV model of the aircraft obtained by the steps described above. The RigACModel is obtained from the nominal low order aircraft model by rezidualizing the structural and lag state dynamics. This model serves for the baseline control design, containing only the 12 rigid body states. These resulting models are saved in the ToolSpecific section of CPACS.

3.2.6 Baseline control design block

Baseline control design inputs

The flutter control design takes the actuator dynamics and the baseline control design model RigAC-Model as inputs via CPACS.

Baseline control design main algorithm

The baseline control system features a classical cascade flight control structure with scheduled control loops to augment the lateral and longitudinal axis of the aircraft. As the cross-coupling between longitudinal and lateral axis is negligible, longitudinal and lateral control design is separated. The control loops use scheduled elements of proportional-integral-derivative (PID) controller structures with additional roll-offs in the inner loops to ensure that no aeroelastic mode is excited by the baseline controller. Scheduling with indicated airspeed V_{ias} is used to ensure an adequate performance over the velocity range from 30 m/s to 70 m/s.

The baseline control design needs to be augmented with verification/analysis algorithms that ensure that the resulting controllers after each MDO iteration satisfy the control performance specifications.

Baseline control design outputs

The output of the block is the baseline controller, saved via CPACS.

3.2.7 Flutter control design block

Flutter control design inputs

The flutter control design takes the outer aileron (denoted by L4 and R4) actuator dynamics and the flutter control design model FlexACModel as inputs via CPACS.

Flutter control design main algorithm

There are two main sub blocks in the flutter control design blocks. First, the design model is split into longitudinal and lateral. These models are then used to synthesize a stabilizing controller for the symmetric and asymmetric flutter mode respectively. Second, the control design consists of the construction of two uncertain plants, performance definitions, and the synthesis of two low-order controllers. These controllers are blended together to obtain the flutter controller. The stability of the resulting flutter controller and a couple of implementation criteria are also tested.

In case of the flutter suppression control desing, the airspeed and the uncertainties in the structural dynamics model are treated at parametric uncertainties and dynamic uncertainty is added to account for the model reduction. In order to reduce the computational time of the control synthesis, structured H_∞ design is chosen that result in an LTI flutter suppression controller. Similarly to the baseline control design algorithm, the flutter suppression control design block needs to be augmented with basic analysis algorithms to verify if the resulting controller satisfies the control performance specifications. As a main measure, the multi-input multi-output (MIMO) disc margins are selected.

Flutter control design outputs

The output of the block is the flutter suppression controller, saved via CPACS.

3.2.8 Closed loop analysis block

Closed loop analysis block inputs

The analysis block requires the baseline, the flutter controllers, the actuator and flexible aircraft dynamics. All the input data is handled via CPACS.

Closed loop analysis block main algorithm

The open and closed-loop flutter speed are determined by multi-loop input-output margin computations. The open+loop flutter speed is the speed below which the margins obtained with only the baseline controller engaged are larger than a predefined threshold. The closed-loop flutter speed is defined similarly with both the baseline and the flutter controllers engaged.

Closed loop analysis block outputs

The closed loop analysis block provides the symmetric and asymmetric gain margins and the analysis runtime results as the main indicators for the designed controller's performance. The data are saved via CPACS.

Gust-load alleviation block
The function is not implemented yet, but will augment the rigid-body control laws and the flutter control laws in providing reduced wing root bending moment during the encounter of windgust disturbance.

The control design will utilize the modified aircraft mathematical model, where wing loads will be performance outputs of an optimization based control design framework for disturbance inputs.

Maneuver-load alleviation block
The function is not implemented yet, but will augment the rigid-body control laws and the flutter control laws in providing reduced wing root bending moment during the pilot commands, especially during maneuvering phase. The control functionality utilizes feed-forward and feedback terms to shift the loads on the wing inboards when high G maneuvers are executed at the expense of small drag penalty, while reducing wing root bending moment.

The control design will utilize the modified aircraft mathematical model, where wing loads will be performance outputs of an optimization based control design framework for disturbance inputs.

Wing shape control block
The function is not implemented yet, but will augment the flexible wing control laws in providing optimal wing shape for the different phases of flight. The control functionality utilizes feed-forward and feedback terms to obtain the optimum cruise wing shape at different part of the mission at various parts of the flight envelope.

The control design will utilize the improved aerodynamics model of the aircraft, since standard ASE dynamical models do not account for induced drag terms. For this reason the FLiPASED team is working on developing low-medium fidelity, 3D panel method based models, including induced drag to provide a framework to optimise wing shape (mainly first bending and torsion) based on fuel consumption at different mass cases and velocities.

3.2.9 Report generation block

A pdf report is generated containing key information about the synthesis and analysis: the open and closed-loop flutter speeds, the robustness margins, the gain of the controller, etc.

3.3 RCE

DLR's Remote Component Environment (RCE) [3] is an open-source software environment for defining and executing workflows containing distributed simulation tools by integrating them into a peer-to-peer

network. The following description has been taken from the related publication by the main developers, Boden et al. [3]. RCE is being developed primarily by DLR and has been used in various engineering projects, including several aerospace projects dealing with multidisciplinary optimization (MDO) and multidisciplinary analysis (MDA). RCE has several advantages that can help to achieve more reusable multidisciplinary processes. The workflow is composed of built-in and user-defined components. Disciplinary tools are integrated as standalone components, with defined inputs and outputs, and then distributed over the network. While executing the workflow, data dependencies between the components are automatically detected, and a component is executed as soon as all its input data is available. Thus, multiple components can run at the same time. The components of a multidisciplinary process can also be executed in a distributed manner, where the tools are located on different machines with possibly different operating systems. Once configured, the peer-to-peer network is automatically established between the RCE instances running on different machines, making components visible and executable even between instances that are only connected indirectly. The distributed execution capability alleviates tool deployment issues, fig. 5, including those related to the protection of intellectual property.

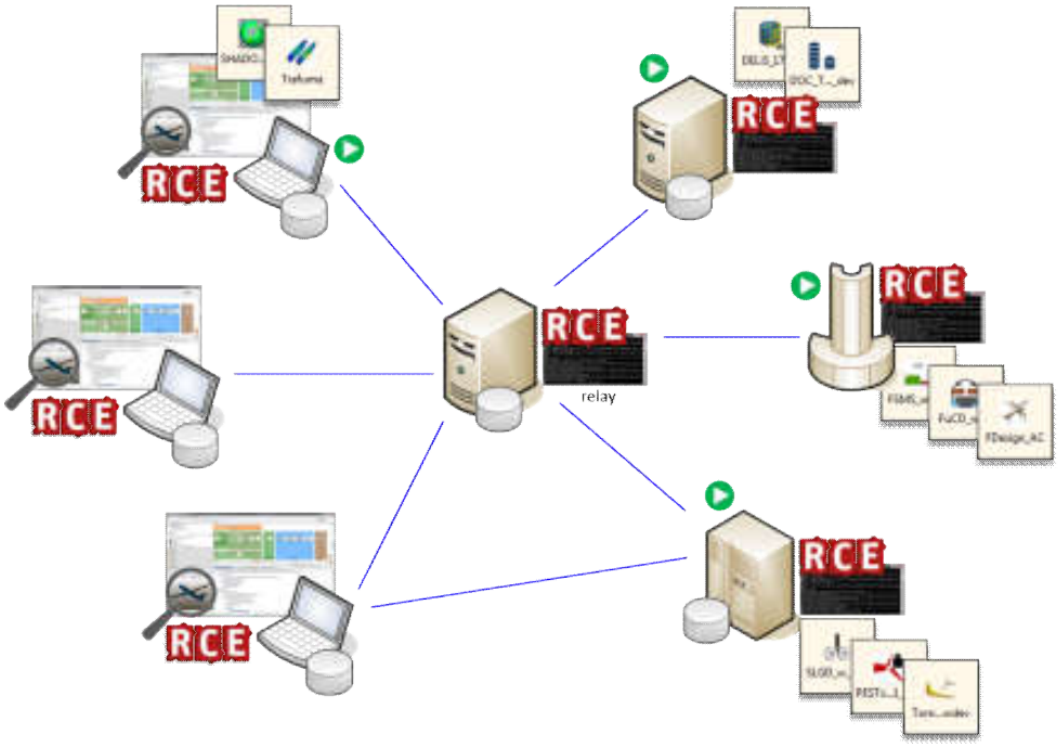

Figure 5: distributed RCE workflow

RCE supplies a graphical editor for creation of workflows, using the built-in components to control the data flow. Some built-in components can be used to perform optimization tasks within the workflow, including nested loops, using built-in or user integrated optimization algorithms. Integrating an external tool into RCE amounts to defining an interface between RCE and the tool, i.e. defining its inputs and outputs to make them accessible by RCE, adding pre- and post-processing steps for the input and output data, and defining the commands for the invocation of the tool. In order to aid the user in defining this integration, RCE features a graphical wizard that guides the user through this process. Once created, the integration is defined by a plain text file which can also be edited using standard text editors. An integrated tool is available locally as a component to be used in workflows and fits seamlessly into

the user interface. In addition to user-integrated tools, RCE provides a number of predefined tools which can be used in conjunction with integrated tools to construct complex workflows. These predefined tools supply a multitude of basic functionalities used in numerous workflows such as handling the flow of data through the workflow, reading and extracting data from files, manipulating XML, executing user-defined Python scripts, and evaluating incoming data. Furthermore, there are predefined components that simplify the construction of workflows for multidisciplinary design optimization, such as basic mathematical and statistical methods. There is also a component that determines absolute or relative convergence of its input values, and a component that provides access to the optimization algorithms implemented by the Dakota software library. Moreover, RCE features a component that allows for the exploration of a parametric design space. This component provides several algorithms for this exploration, among them a design based on Latin Hypercube sampling or on a Monte Carlo approach. The user, however, also has the flexibility to specify a custom design. After integrating the tools required for the execution of the workflow, the user may compose them into a workflow. To this end, RCE offers a graphical editor allowing the user to construct a workflow by first dragging and dropping the required components into the editor and subsequently connecting their respective inputs and outputs. After constructing such a workflow, the user can execute it. RCE has been used in many different MDO/MDA related projects. A rather involved process flow was implemented in the Digital-X project [4] which is depicted in fig. 6.

The MDO process in Digital-X was a multi-level one, comprising tools and sub-processes ranging from lowfidelity, over mid-fidelity, to high-fidelity and computation-intensive.

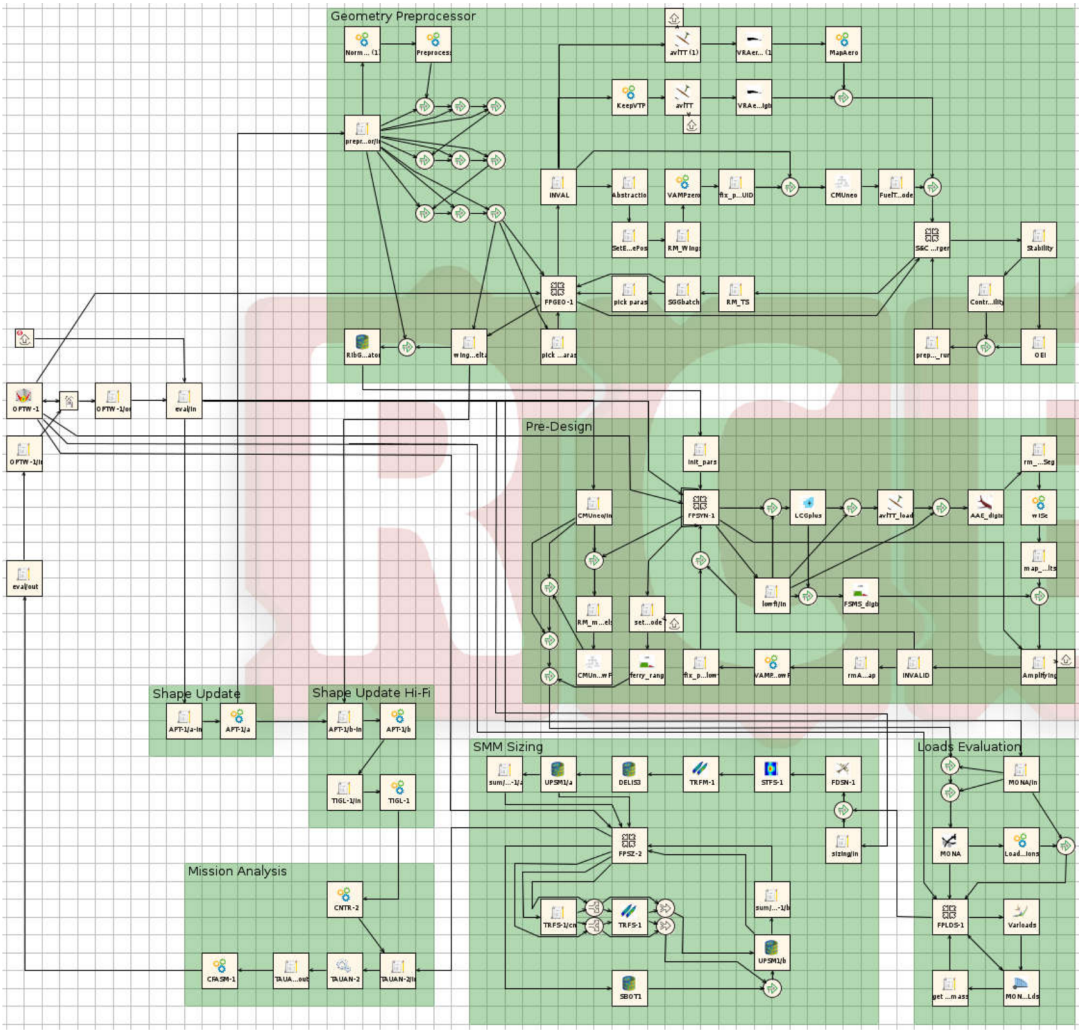

Figure 6: RCE implementation of the Digital-X MDO process

4 Definition of interfaces

4.1 CPACS

Aircraft design projects are characterized by interdisciplinary collaboration between a large amount of heterogeneous disciplines. Each discipline contributes with specialized knowledge and software tools which are integrated in automated simulation workflows. The utilization of a common data model significantly reduces the number of possible interconnections between the modules and ensures a consistent source of information. Such a data model was established in the Common Parametric Aircraft Configuration Schema (CPACS) by the DLR. The following description is largely based on the associated publication of Alder et al. [1]. Aircraft design workflows must account for the interaction of various disciplines such as structural mechanics, aerodynamics or flight mechanics. One possibility to consider this interaction is to integrate the required sub-disciplines in a monolithic software architecture used to synthesize an aircraft on conceptual and design level by applying sequential iteration methods. From a developer's point of view, the internal data exchange between analysis modules within the monolithic system is advantageous concerning the easiness of resolving data and model inconsistencies. Due to the increased complexity of the design considerations already in early aircraft design stages nowadays however, the process cannot be handled by a single person anymore. Therefore, today's research on collaborative MDO is often based on tool integration framework which allow to integrate analysis modules in decentralized workflows enabling engineering departments at different sites to be involved in the design process. The open-source software RCE [3] (Remote Component Environment) is an example of such a process integration framework. It enables the connection of analysis modules via a server-client based network infrastructure. Upon workflow execution, the execution of individual modules hosted on their respective server instances is triggered when required and the required data is automatically exchanged. In this construct, only input and output data is exchanged while the tool itself remains under control of the tool owner. A challenge arising within this approach is that the stakeholders might use different data models and vocabulary resulting in N(N-1) possible directions of data exchange between N tools. Within such a simulation process, the consistency among the multiple disciplinary models and different levels of details of the simulations needs to be guaranteed. One solution to this challenge is obtained by introducing a central data exchange format based on common semantics for the whole system to be designed (e.g., full parametrization of an aircraft) which is easy to read and interpret by human. As depicted in fig. 7, by using this single source of truth the amount of connections reduces to 2N. This is the main motivation for the development of common aviation data models.

The data model CPACS has been introduced and developed at the German Aerospace Center (DLR) since 2005. CPACS is implemented in XML. XML is an open standard, which is officially coordinated and documented by the Wold Wide Web Consortium (W3C) and nowadays globally accepted in the field of information technology. XML has a very generic character and can therefore serve as a computer-processable meta-language enabling the development of an aviation ontology as a markup language itself. Another strength of XML is the separation of the data structure from the actual content. This allows for the definition of complex structural and semantic rules in a separate XML Schema Definition (XSD) file, while users can independently describe data using an exchange format that is easy to read by just using a text editor. Making use of the hierarchical representation of data in XML the structure of CPACS mainly follows a top-down approach which decomposes a generic concept (e.g., an aircraft) into a more detailed description of its components. This originates from the conceptual and preliminary design of aircraft, where the level of detail is initially low and continues to increase as the design process progresses. The hierarchical structure furthermore promotes the simplicity of the exchange format which is required in collaborative design environments so that the various stakeholders can easily append their results. Furthermore, supporting libraries like TIXI and TiGL [9] exist to interface, respectively visualize the current aircraft configuration, cf. fig. 8.

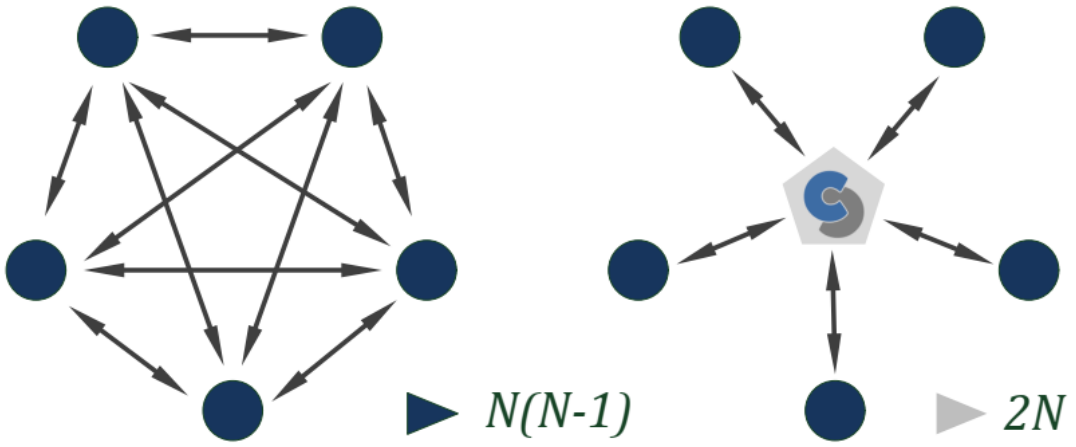

Figure 7: CPACS as common interface between disciplines

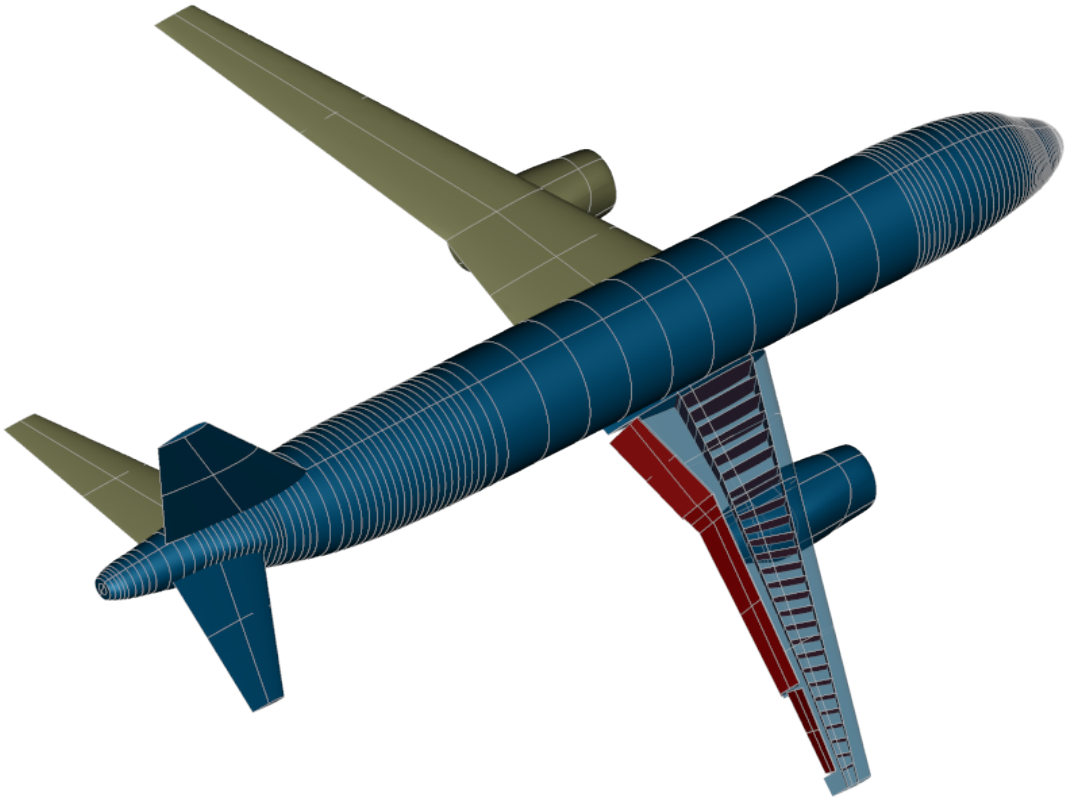

Figure 8: CPACS based aircraft configuration with internal structure as visualized by TiGL 3.0

4.2 Interface: CAD model to FE-model of wing

CPACS serves as the interface between the CAD model and FE-model. The path of the Catia file, which is the output of the geometry modeling block, would be saved in a CPACS file. The structural modeling block would read the CPACS file and find the Catia file based on the path. Thanks to the central data

model CPACS, there is no needs to develop a specific interface between Catia and HyperMesh.

The position and number of ribs depend on the configuration of flaps. So there are no definitions of ribs in the CPACS file but only the flaps. For the sake of completion, ribs would be treated as the geometry modeling block's output and written into the CPACS file.

4.3 Interface: CAD model to Aero-model of wing

For the aerodynamic modeling, the consortium decided to use DLM. In order to generate the panel model, a Python script is written to extract the geometry information from the CPACS file and export the panel model in Nastran bdf format. There is actually no direct interface between the CAD model and the aerodynamic model because no information is extracted from the Catia model.

4.4 Interface: wing FE/aero-model to full aircraft NASTRAN aeroelastic model

The structural FE model of the wing is obtained from the CAD-FEM toolset at TUM. This wing model is integrated to the fuselage and empennage based on aeroelastic models generated during FLEXOP at DLR-AE.

In order to smoothen this integration, an interface between the models is set up. This is in the form of a document describing a numbering scheme for the different cards present in the models, for each component. Additionally, connection points between the components, for instance, between the fuselage and wings is also specified, such that iterations of the wing models can always be integrated to the aircraft model without any changes or adaptations necessary.

Similarly, the aerodynamic DLM model of the wing is also integrated with those of the fuselage and empennage. During the course of the MDO task, the FE and DLM models of the fuselage and empennage are proposed to remain unchanged, using the same design and models as in the FLEXOP project.

The interface to the downstream blocks is through mass, stiffness matrices and additional bulk data required for aeroelastic analyses. These are transferred to a suitable directory, for access to the MDO blocks downstream.

4.5 Interface: FE-models to structural optimization block

The interface between the FE modelling block and the structural optimization block is through property cards in the structural model. As inputs to the structural optimization block, the FE model of the wing including its property cards, the loads to be used for the sizing and a set of optimization options are provided. These options include for instance, thickness bounds, laminate design constraints, safety factor on structural constraints, etc. The structural properties of the fuselage and empennage are not optimized.

The output of the structural optimization block is a new set of property cards, that represent the stiffness-optimized wing. These property cards directly influence the stiffness of the wing. This is propagated to the downstream MDO blocks via the condensed NASTRAN aeroelastic model of the full aircraft.

4.6 Interface: controllers

4.6.1 Expected closed-loop structure

Generally, aircraft manufacturer control design workflow follows what we can call a frequency grid approach. This approach consists in designing different controllers, through a frequency guideline. Each of them address a phenomena an aircraft is faced during its operation. Within the overall MDO process philosophy and in this WP, we aim at following this approach. With reference to Figure 9, one may notice that different phenomena (flight, loads...) usually occurs at different frequencies. These frequencies are dependent on the geometry and structure of the aircraft. In the considered case, one may expect even more blending between each phenomenon. Still the big picture remains valid. This sequential control structure will be kept in mind in the WP2 flow to stick to industrial and practical expectations.

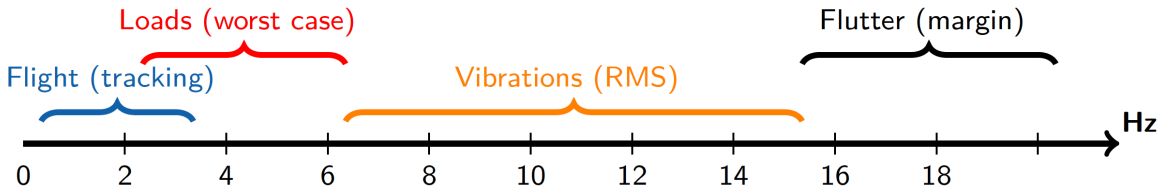

Figure 9: Frequency grid of the physical phenomena occurring over an aircraft. Ranges and values are different from an aircraft to an other.

As a matter of consequence, the closed-loops one is intended to develop is presented as in Figure 10, where each function in cascaded with the other. More specifically, the flight controller aims at focusing on the handling qualities and manoeuvrability while the load control focuses either on maneuver of gust phenomena. One underlying objective of this WP2 is to design such control law, but not only. As the complete process addressed in the FLiPASED project is an MDO one, aircraft parameters $\mathbf{p}$ will also be tuned and optimised, together with the control. These control law are usually designed following the increasing frequency physics: first flight control, then load, etc.

As presented in Figures 9 and 10, the flight control system layout will gather a set of multiple functions. Each function should be independently designed without affecting the others. Moreover, as the functions are connected but somehow with different objectives, we will consider designing them with the following sequence:

1. Flight control, a flight oriented control
2. MLA, a maneuvers load alleviation control
3. GLA, a gust load alleviation control
4. Flutter, a flutter shield control

As all these phenomena are specific and operate at different frequencies, the models $\mathbf{H}(\mathbf{p})$ (where $\mathbf{H}$ is a complex linear map and $\mathbf{p}$ the parameter vector) involved in the design optimisation step may vary from a function to an other. By this one intends that even if one single global model is provided by the upper WP, different sub models may be constructed within this WP, accordingly to the considered phenomena and control design objective.

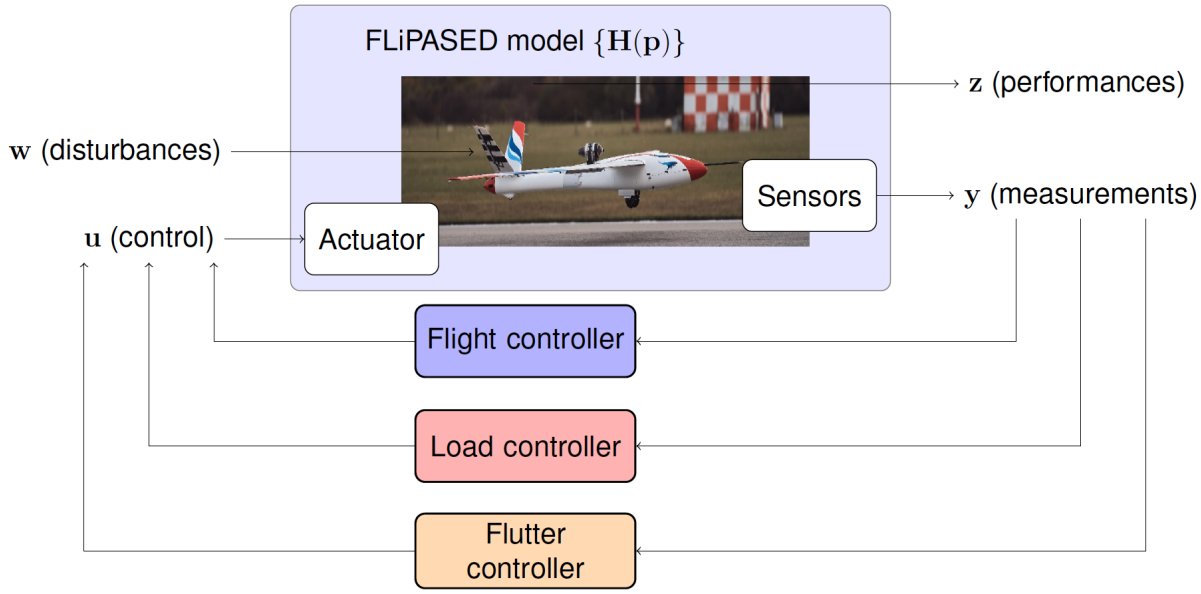

Figure 10: Multiple control loops considered in the WP2.

4.6.2 Input-output data description

Without detailing the typical inputs and outputs, the following Figure 11, gathers the main input and output data that has to be exchanged from the upper WP to the lower ones. The global interconnection is referenced in the main project document.

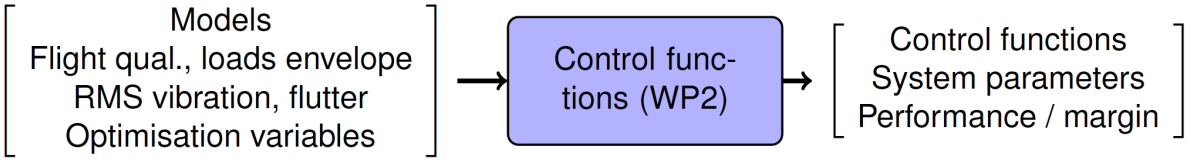

Figure 11: Data exchanges within WP2.

Inputs When dealing with control, the main input to be considered are the model, denoted **H** or $\mathbf{H}(\mathbf{p})$, when considering a parametric one, or $\{\mathbf{H}_i\}_{i=1}^{n_s}$ if a set of models is given. More specifically, the following inputs are expected.

- Linear (or nonlinear) dynamical models / simulators (finite or infinite dimension)
- Gust load envelope (nominal)
- Maneuver load envelope (nominal)
- Flutter-free envelope (nominal)
- Sensors/actuators dynamical characteristics and degree of freedom (such as location, speed...)

Outputs As rooted on the above inputs, the WP2 aims at delivering both *functions and results*. The former being *functions*, represent features developed that can be called within the MDO process, to get optimised aircraft (including control laws and optimised aircraft design parameters). The latter being *results*, represent tag (such as robustness, stability, ...) that can be used to decide the quality of the actual design and derive some optimisation directions. The following gives the main outputs, plus some potentially fruitful ones.

- Control functions (Flight / MLA-GLA / Flutter)
- Performance metrics
- System parameters

\+ Reduced models (suitable for control and analysis)

4.6.3 Functional task sharing

The organisation between the three research groups involved in this WP2 follows the Table 1. This table presents the functions to develop and the holder of each of them. As a preliminary report, this table should be amended and filled during the project life. Still, as this point, checking all these tasks will result in a complete tool for fast aircraft control law and parameter optimisation.

WP	Title	Function	Holder
2.1 Modelling	Construction linear ROM	✓ Order reduction automatic guess	ONERA
		✓ Model order reduction from finite realisation	SZTAKI
		✓ Model order reduction from infinite realisation or transfer function	ONERA
	Construction parametric ROM	✓ From bundle of LTI models	SZTAKI
	Construction LPV ROM	✓ From bundle of LTI models	SZTAKI
	Model integration	✓ Integrate the models within the MDO tool	SZTAKI
2.2 Control	Flight qualities	✓ Design using INDI	DLR
		✓ Design using scheduled PID	SZTAKI
	GLA	✓ Design using LTI $\mathcal{H}_\infty$ control	DLR
		✓ Design using LTI Modal control	DLR
		✓ Design using LPV control	SZTAKI
	MLA	✓ Design using (linear) MPC control	SZTAKI
		✓ Design using structured $\mathcal{H}_\infty$ control	ONERA
	Flutter control	✓ Design using LTI $\mathcal{H}_\infty$ control	DLR
		✓ Design using LTI Modal control	SZTAKI
		✓ Design using LPV control	DLR
	Others	✓ Actuators / sensors placement	DLR
		✓ Wing shape control for optimal drag configuration	SZTAKI
2.3 Analysis	Performances	✓ Assessment function of the performances	ONERA
		✓ HiL	SZTAKI
		✓ Worst case analysis	SZTAKI
		✓ Actuator limits	DLR
		✓ $\mathcal{H}_\infty$ norm computation (large-scale and delayed models)	ONERA

Table 1: Task sharing.

5 Conclusion

This deliverable clearly distinguishes the toolchains used for demonstrator wing design and commercial aircraft wing design and specifies the corresponding objectives and requirements. The preliminary structure and functionality of the MDO toolchain are proposed. The integration framework RCE and common data model CPACS are determined as the standard tools for the toolchain.

The aforementioned outcomes clarify the common stand for the consortium and lay the groundstone for the subsequent toolchain implementation.

6 Bibliography

[1] Marko Alder, Erwin Moerland, Jonas Jepsen, and Björn Nagel. Recent advances in establishing a common language for aircraft design with cpacs. In *Aerospace Europe Conference 2020, 25-28 Feb 2020, Bordeaux, Frace.*, 2020.

[2] G. Becker. *Quadratic Stability and Performance of Linear Parameter Dependent Systems*. PhD thesis, University of California, Berkeley, 1993.

[3] Brigitte Boden, Jan Flink, Robert Mischke, Kathrin Schaffert, Alexander Weinert, Annika Wohlan, Caslav Ilic, Tobias Wunderlich, Carsten M. Liersch, Stefan Görtz, Erwin Moerland, and Pier Davide Ciampa. Distributed Multidisciplinary Optimization and Collaborative Process Development Using RCE. In *AIAA Aviation 2019 Forum, 17–21 June 2019, Dallas, TX, USA*. American Institute of Aeronautics and Astronautics, 2019.

[4] S. Görtz, C. Ilic, M. Abu-Zurayk, R. Liepelt, J. Jepsen, T. Führer, R. Becker, J. Scherer, T. Kier, and M. Siggel. Collaborative multi-level mdo process development and application to long-range transport aircraft. In *30th International Congress of the Aeronautical Sciences, Daejeon, South Korea, September 25-30. 2016*. ICAS, 2016.

[5] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46(2):273–284, Aug 2012.

[6] Yasser M. Meddaikar, Johannes Dillinger, Thomas Klimmek, Wolf Krueger, Matthias Wuestenhagen, Thiemo M. Kier, Andreas Hermanutz, Mirko Hornung, Vladyslav Rozov, Christian Breitsamter, James Alderman, Bela Takarics, and Balint Vanek. Aircraft aeroservoelastic modelling of the FLEXOP unmanned flying demonstrator. In *AIAA Scitech 2019 Forum*. AIAA, jan 2019.

[7] Andreas Page Risueño, Jasper Bussemaker, Pier Davide Ciampa, and Bjoern Nagel. *MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems*.

[8] Jeff S. Shamma. *Analysis and Design of Gain Scheduled Control Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 1988.

[9] Martin Siggel, Jan Kleinert, Tobias Stollenwerk, and Reinhold Maierl. TiGL: An open source computational geometry library for parametric aircraft design. *Mathematics in Computer Science*, 13(3):367–389, jul 2019.

[10] Bela Takarics, Balint Vanek, Aditya Kotikalpudi, and Peter Seiler. Flight control oriented bottom-up nonlinear modeling of aeroelastic vehicles. In *2018 IEEE Aerospace Conference*. IEEE, mar 2018.

[11] G. Vinnicombe. *Measuring Robustness of Feedback Systems*. PhD thesis, Univ. Cambridge, Cambridge, 1993.

[12] Fen Wu. *Control of Linear Parameter Varying Systems*. PhD thesis, Univ. California, Berkeley, 1995.

[13] Matthias Wuestenhagen, Thiemo Kier, Yasser M. Meddaikar, Manuel Pusch, Daniel Ossmann, and Andreas Hermanutz. Aeroservoelastic modeling and analysis of a highly flexible flutter demonstrator. In *2018 Atmospheric Flight Mechanics Conference*. AIAA, jun 2018.