FliPASED
WWW.FLIPASED.EU

D2.2 Report on tool adaptation for collaborative design

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: 31/10/2020	

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
CAD	Computer-aided Design
CPACS	Common Parametric Aircraft Configuration Schema
DLM	Doublet Lattice Method
DMAP	Direct Matrix Abstraction Program
FE	Finite Element
GLA	Gust Load Alleviation
LPV	Linear Parameter-varying
LRA	Load Reference Axis
LPI	Linear Time-invariant
MDA	Multidisciplinary Analysis
MDO	Multidisciplinary Optimization
MIMO	Multi-Input Multi-Output
MLA	Maneuver Load Alleviation
PID	Proportional-Integral-Derivative
RCE	Remote Component Environment
TCL	Tool Command Language
W3C	Wold Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition

Table of contents

List of Figures

1 Executive summary

In order to achieve advanced, actively flight condition optimized wing design, a collaborative design toolchain needs to be set up through integrating separately existing tools in each discipline. To guarantee a successful integration, the tools need to be modified according to MDO requirements. This deliverable describes the approach followed for adaptation of tools developed in the previous project FLEXOP, to make those mature for integrated collaborative design. This deliverable will serve as the instruction for the later implementation and integration into a collaborative framework.

The work is divided generally based on disciplines(structure, aerodynamics, and control), meanwhile taking into account the existing outcome achieved from the project FLEXOP.

Chapter 2 introduces the common data format CPACS and the collaborative framework RCE. CPACS will serve as the interface between different tools. RCE will be used to integrate all the tools. This chapter was contributed by Thiemo Kier (DLR-SR).

Chapter 3 describes the tool adaptation in the structural design area. It covers the design freedom for the new wing, parametrized geometrical and structural model, structural condensation and optimization. This chapter was contributed by Muhammad Meddaikar (DLR-AE) and Fanglin Yu (TUM).

Chapter 4 describes the tool adaptation in the aeroelasticity area. It covers the coupling between the structural model and the panel model. Flutter calculation and load analysis are also included. This chapter was contributed by Muhammad Meddaikar(DLR-AE), Matthias Wuestenhagen(DLR-SR) and Fanglin Yu(TUM).

Chapter 5 describes the tool adaptation in the control design area. It was contributed by Charles Poussot-Vassal(ONERA) and Béla Takarics(SZTAKI).

Chapter 6 describes the evaluation of RCE implementation and the performance of blocks in the toolchain. This chapter was contributed by Béla Takarics(SZTAKI).

In chapter 7 a conclusion is given.

2 Integrated, collaborative design tool chain

2.1 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. 1, 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 [8] exist to interface, respectively visualize the current aircraft configuration, cf. fig. 2.

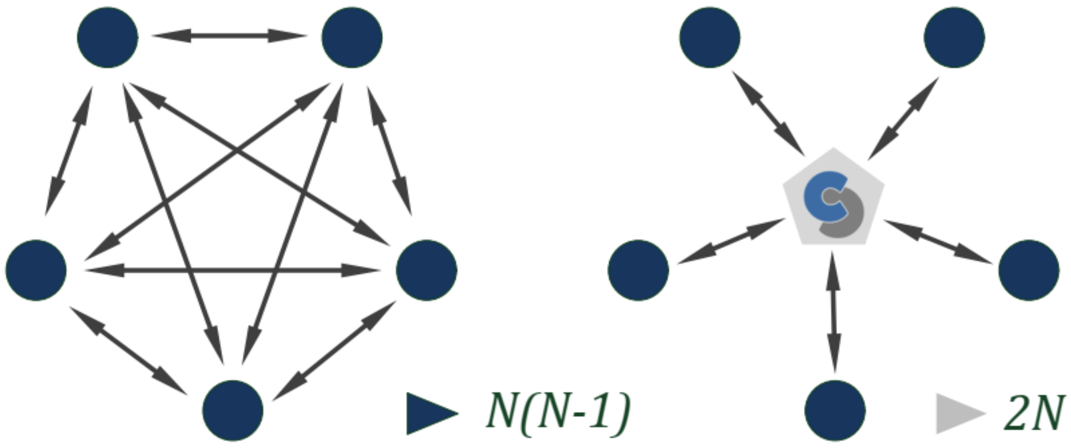

Figure 1: CPACS as common interface between disciplines

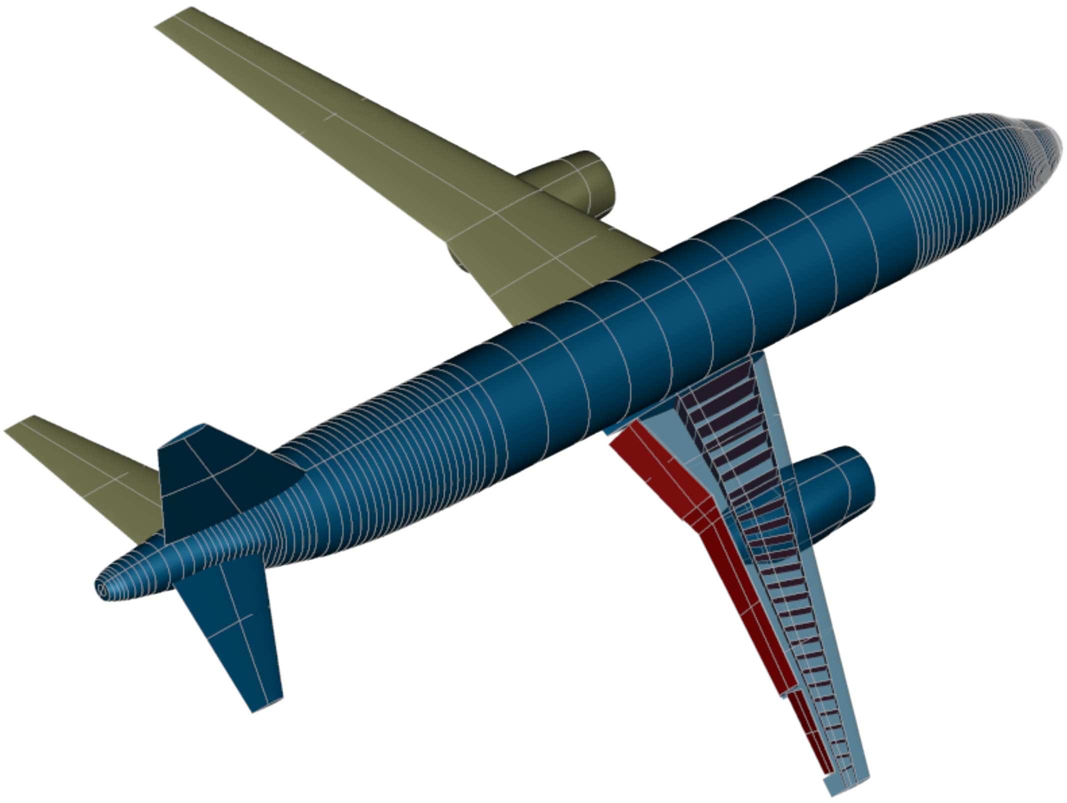

Figure 2: CPACS based aircraft configuration with internal structure as visualized by TiGL 3.0

2.2 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. 3, including those related to the protection of intellectual property.

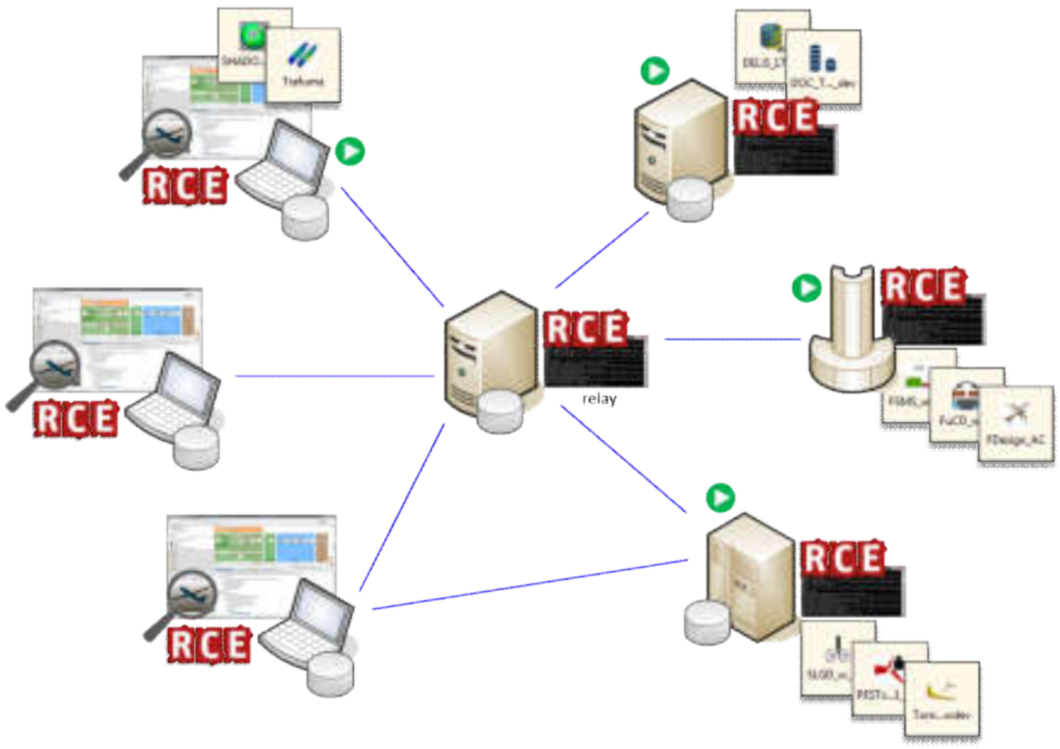

Figure 3: distributed RCE workflow

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. 4.

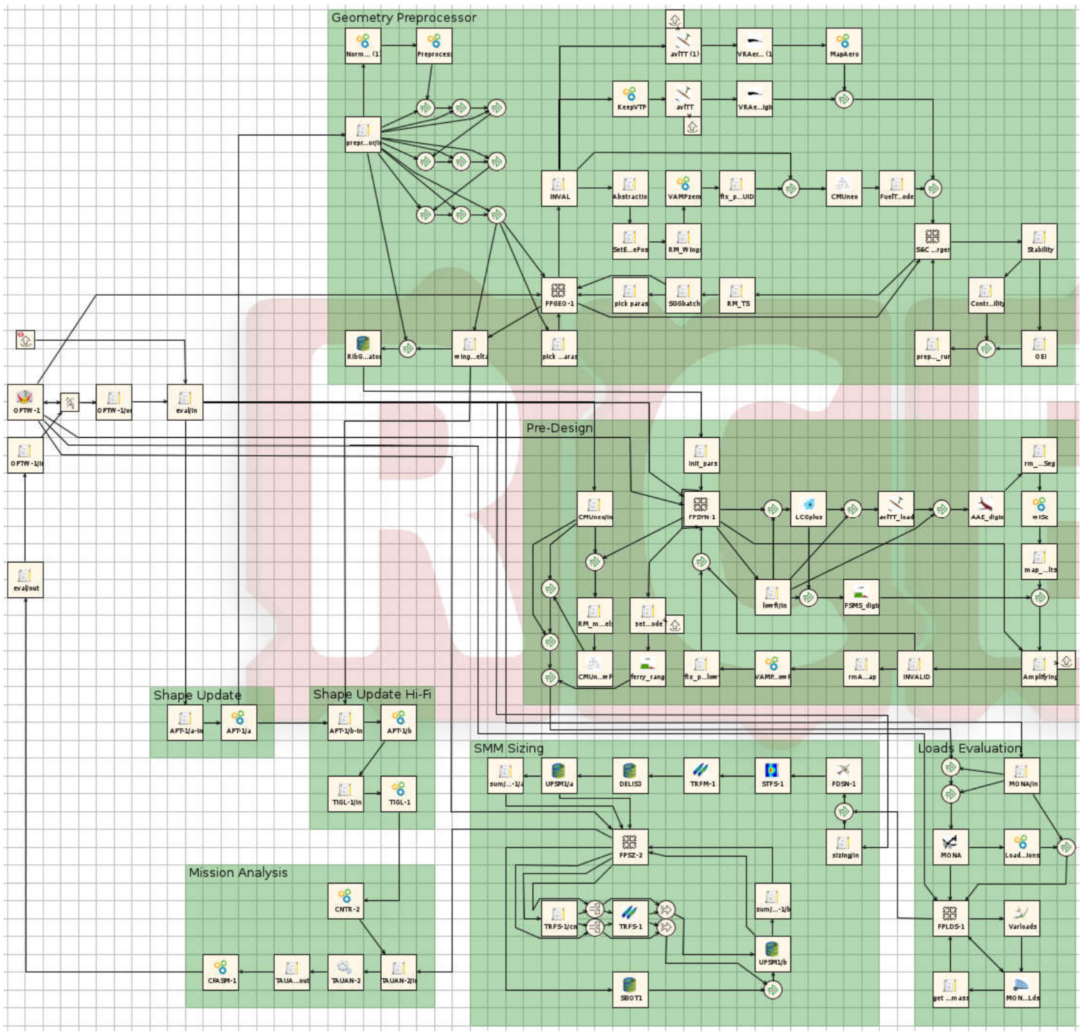

Figure 4: RCE implementation of the Digital-X MDO process

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.

3 Tool adaptation for structural design

3.1 Design-space for model-generation

One of the first blocks in the MDO toolchain is the generation of structral models. Parametrized structural models using CAD and FEM methods are generated using established workflows by TUM. These models will then serve as input for the generation of reduced and parametric dynamical models, tailored to the required down-stream processes.

In order to define the potential design space during the MDO optimization, a discussion of the possible parameters that can be varied as part of the modelling process is considered first. With regards to the wing model, the following paramaters are considered as part of the design freedom,

– wing planform - sweep, taper ratio, span
– spar positions of the primary wingbox
– pre-defined jig twist
– number of flaps - a single structural model is generated comprising of the highest number of flaps to be considered; by using suitable stiffness connections, several of these control surfaces can be effectively made to function as a discrete smaller set of control surfaces
– flap positions - via discrete combination of the modelled flaps
– skin stiffness - enabled through property cards that can be varied during down-stream processes

The fuselage and V-tail models are re-used from FLEXOP. It is however considered at this point that a large variation in the planform of the wing could require a potential change in the dimensions of the V-tail as well.

3.2 Parameterized CAD and FE-Model

In order to achieve MDO, a parametrized CAD model is the cornerstone of the whole toolchain. The geometrical modeling highly depends on individual cases and different needs. Developing a general geometric modeling tool to automatic generate CAD model from scratch requires lots of programming efforts. Considering huge effort demand and project time constraint, it is wise to generate the first version of the CAD model manually and achieve parameter variation through an automatic model update. This is also consistent with the description of this task, ”semi-automated model design” in the project proposal.

The first step to construct a parametrized CAD model is choosing carefully variable parameters by considering the design objectives of the new wing and balancing the implementation difficulties. Section 3.1 provides a detailed description of the whole design space. Only the geometry relevant parameters of the wing are listed here in Table 1.

Catia V5 is widely used in the aviation industry for geometrical modeling. Therefore Catia V5 is also chosen to construct a parametrized CAD model in the project. During the construction process, continuous parameters, for instance, sweep angle, wingspan, etc. are saved in a construction table as a form of a text file. At each iteration of optimization, a Makro script would be invoked to read inputs

Parameter	Type
wing kink position	continuous
wing root chord	continuous
wing tip chord	continuous
wing root twist	continuous
wing tip twist	continuous
wing span	continuous
wing sweep	continuous
flap spanwise start position	continuous
flap spanwise end position	continuous
number of flaps	discrete

Table 1: variable geometrical parameters in CAD model

and update the geometry model. For the discrete parameter, the number of flaps, three configurations are prepared during the construction. Wenn the parameter, flap number, is read, the corresponding configuration would be activated in Catia. The final parameterized CAD model is shown in Figure 5.

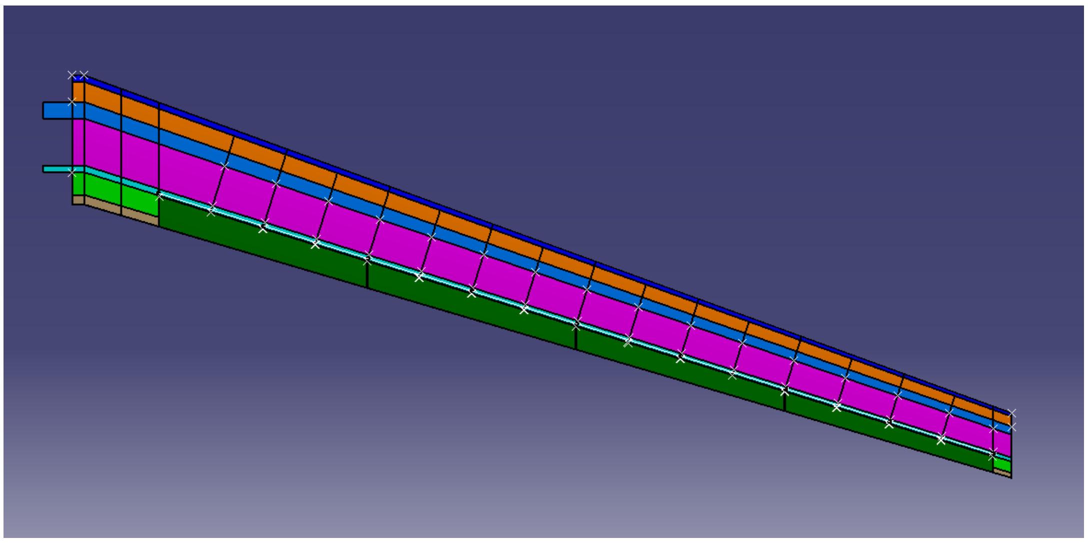

Figure 5: parameterized CAD model

The parametrized CAD model serves as the input for the FE model generation. The FE model is parametrized through the parametrization of the CAD model. The structure relevant parameters (listed in Table 2) are implemented in the CAD model during the construction process. Among the parameters,

Parameter	Type
front spar position root	continuous
front spar position tip	continuous
rear spar position root	continuous
rear spar position tip	continuous

Table 2: variable structural parameters in CAD model

one point needs to be emphasized that the position and number of ribs are not listed as a variable. Because the position and number of ribs highly depend on the configuration of flaps. Each flap needs two supporting ribs at both ends. To reduce the complexity of geometry modeling, the position and number of ribs are regarded as an indirect variable, which changes following flaps' configuration.

HyperMesh is chosen as the mesh tool for its abundant features and TCL Makro script language. Using TCL script, HyperMesh can import the Catia file and mesh all the surfaces automatically. To meet the demands for design regions from downstream structural optimization, elements in one design region are grouped and assigned with individual property, as Figure 6 shows. The maximal number of design regions for each area are listed below.

– a - fuselage hull is modelled as equivalent beam elements (CBEAM in MSC.NASTRAN)
– DLM model - a cruciform T-arrangement for the aerodynamic panels
– concentrated point masses for non-structural components
– interface - provisions for connection with the wings and empennage

The number of design regions can be easily reduced by assigning the same properties to different regions.

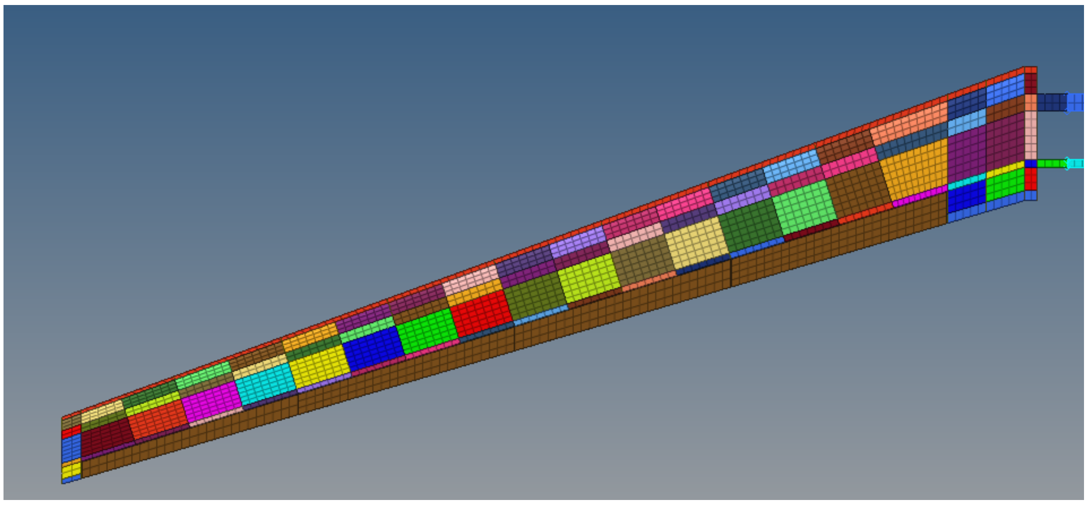
Figure 6: FE model of the wing with design regions

Non-structural mass, servo cable, wing wheel, etc. are also automatically added in the FE model. To enable the downstream Guyan reduction of the FE-model, grids along the LRA of the wing are also created, as Figure 7 shows. The FE-model of the wing would be integrated with the fuselage and empennage. To smoothen this integration and avoid conflicts between these models, the whole modeling process follows the same numbering scheme for grids, elements, and properties. The connection grids between components are also specified.

3.3 CAD-FEM interface

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.

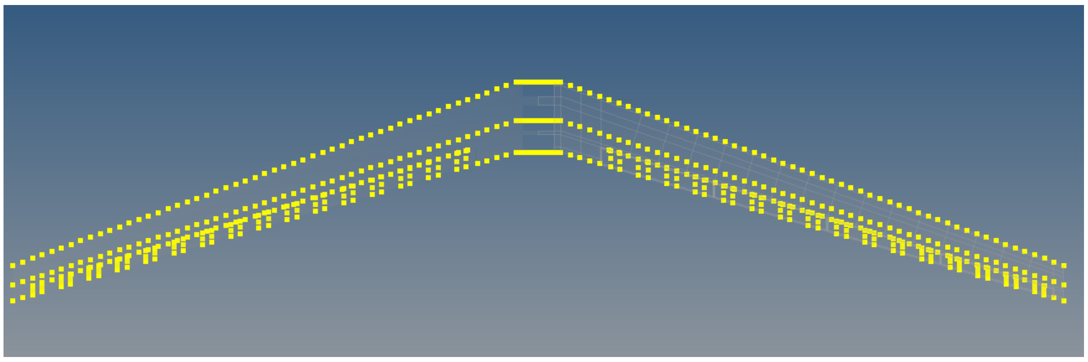

Figure 7: condensation points of the wing

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.

3.4 Full aircraft aeroelastic model integration

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. These were generated using an in-house model generator ModGen [5].

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.

For the sake of completion, a brief summary of the fuselage and empennage models are presented below.

Fuselage

– FE model - fuselage hull is modelled as equivalent beam elements (CBEAM in MSC.NASTRAN) (Figure 8)

– DLM model - a cruciform T-arrangement for the aerodynamic panels

– concentrated point masses for non-structural components

– interface - provisions for connection with the wings and empennage

Empennage

– FE model - a shell-element based model comprising of upper and lower skins, structural ribs, spars and spar-cap (Figure 9)

– DLM model - based on the planform of the empennage

– concentrated point masses for non-structural components

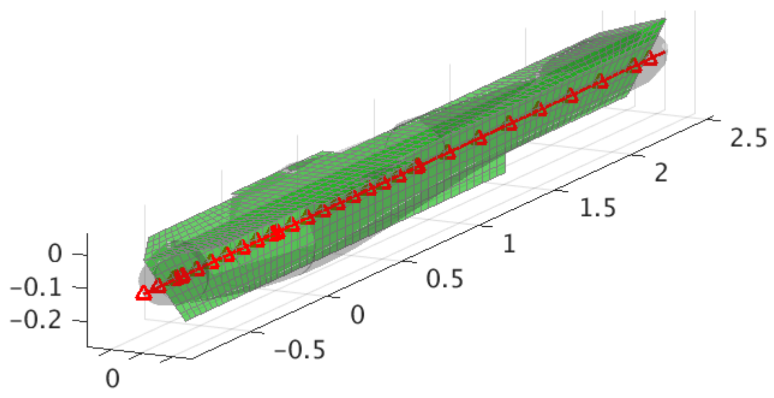

Figure 8: Fuselage DLM, hull, beam model

– provisions for splining using the load reference axis (LRA) method, necessary for aeroelastic analysis (Figure 10)

The structural and aerodynamic models of the fuselage and empennage are presumed to remain the same during the course of the MDO studies. The wing models which will evolve during the course of an MDO run are referenced using suitable 'include' cards in MSC.NASTRAN in their respective solution decks.

3.5 Structural FE model condensation

Solution decks for MSC.NASTRAN are prepared beforehand for modal analysis using the full FE model and to export mass and stiffness matrices via the Guyan reduction. The condensation points for the this reduction include the following: fuselage nodes, nodes along the LRA of the empennage, nodes along the LRA of the wings and flaps.

A Python-script is prepared as a wrapper for the FE model integration block. Test runs for modal analysis, Guyan condensation, aeroelastic trim analysis and flutter analysis are performed once new wing models are generated and the FE model integration block is invoked. The output mass, stiffness matrices and additional bulk data required for aeroelastic analyses are then transferred to a suitable directory, for access to the MDO blocks downstream.

3.6 Structural optimization

The structural optimization block in the MDO toolchain is carried out using an existing in-house composite tailoring framework at DLR-AE [9].

The framework is an externally-driven optimization process, where the workflow is programmed in Python while solutions and solution derivatives are carried-out by external software such as MSC.NASTRAN. The libraries used to implement such execution flows have been named tentatively as ModOpt and are designed to be used optionally and together with OpenMDAO in a complementary fashion as shown in Figure 12.

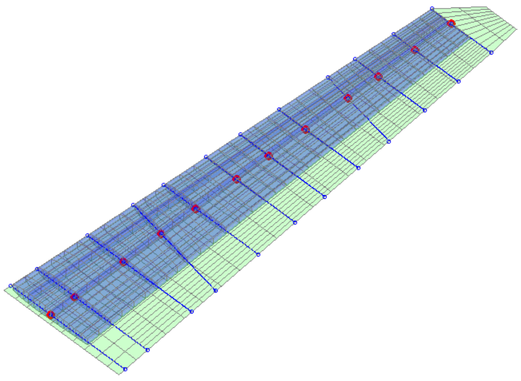
Figure 9: Overlay of FE, DLM model and LRA-based splining of empennage

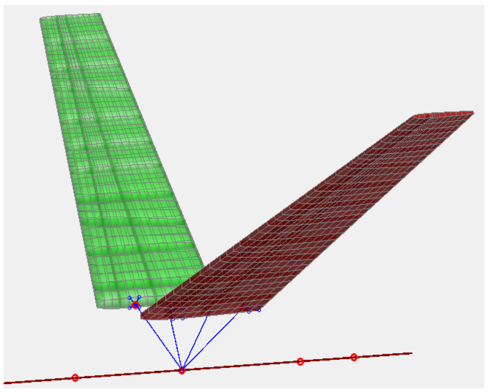
Figure 10: Empennage FE model generated using ModGen

Interface with the structural model

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 output of this block is a new set of property cards, that represent the stiffness-optimized structure. Additionally, useful information such as structural weight and failure indices corresponding to the different structural failure modes considered, are written out in log files to enable viewing of the present status of the structural design at any point during the MDO run.

Adaptation for MDO workflow - demonstrator design

Discussions during the requirements capture in WP1 led to the conclusion that a traditional structural sizing, in the sense of minimizing structural weight of the wings, would not be a practical objective function toward the wing's design. Instead, the global objective for the design of the wings would be to enhance the demonstration or visibility of the various controls technologies.

Towards this end, the structural optimization block would be in effect, a structural-check block. A pre-defined and conservative thickness for the various wing components would be given at the start of the MDO loop. The different structural failure modes such as material failure and buckling would be checked during each iteration to be certain that a sufficient safety margin exists.

In order to tap the tailoring potential of composites in improving the outer objective function, the wing stiffness will be parametrized by two principal stiffness directions. A pre-defined conventional laminate will be specified for the upper and lower skin. Two design variables on the outer MDO loop will alter the principal direction of this laminate on the upper and lower skins. This principal angle alters the property cards related to the stiffness of the skins. This change is effected down-stream via the mass and stiffness matrices obtained from an updated Guyan condensation run.

The advantage of elevating the stiffness-related design variables to the outer-most loop is that inner convergence loops are avoided. This helps reduce the complexity of the implementation significantly and improves run times to convergence.

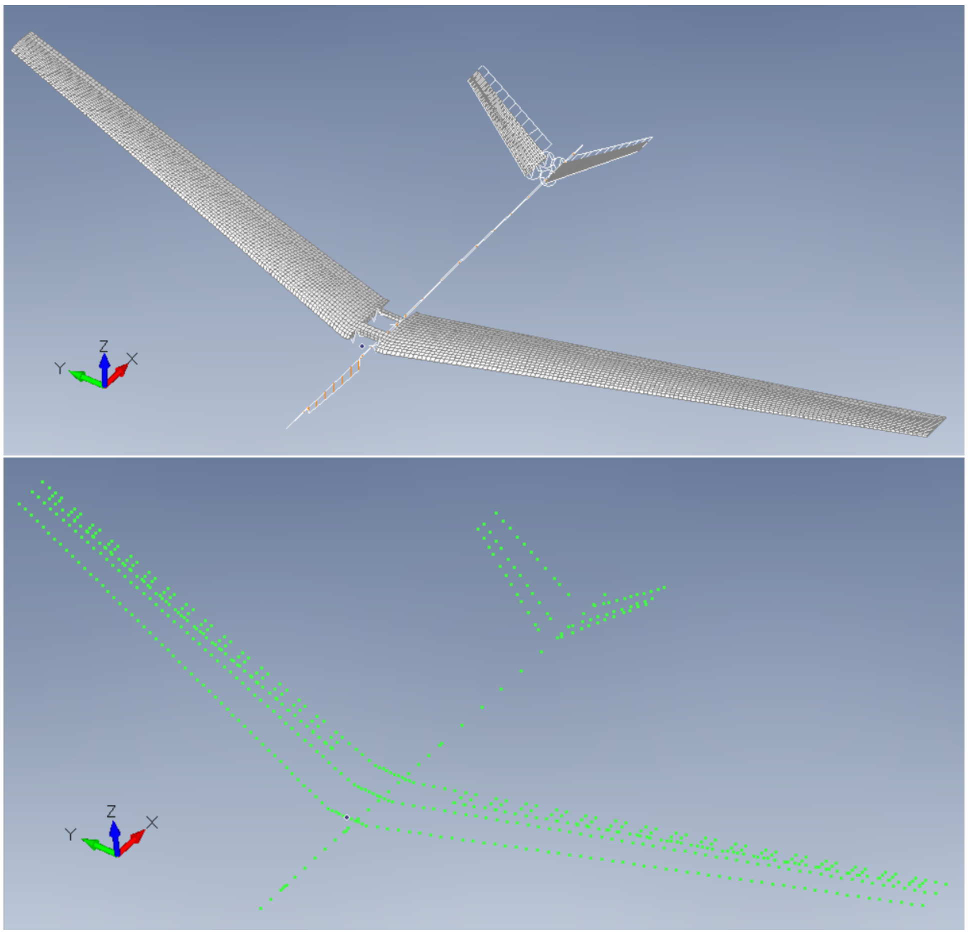

Figure 11: Full aircraft FE model (above), reduced model after Guyan reduction (below)

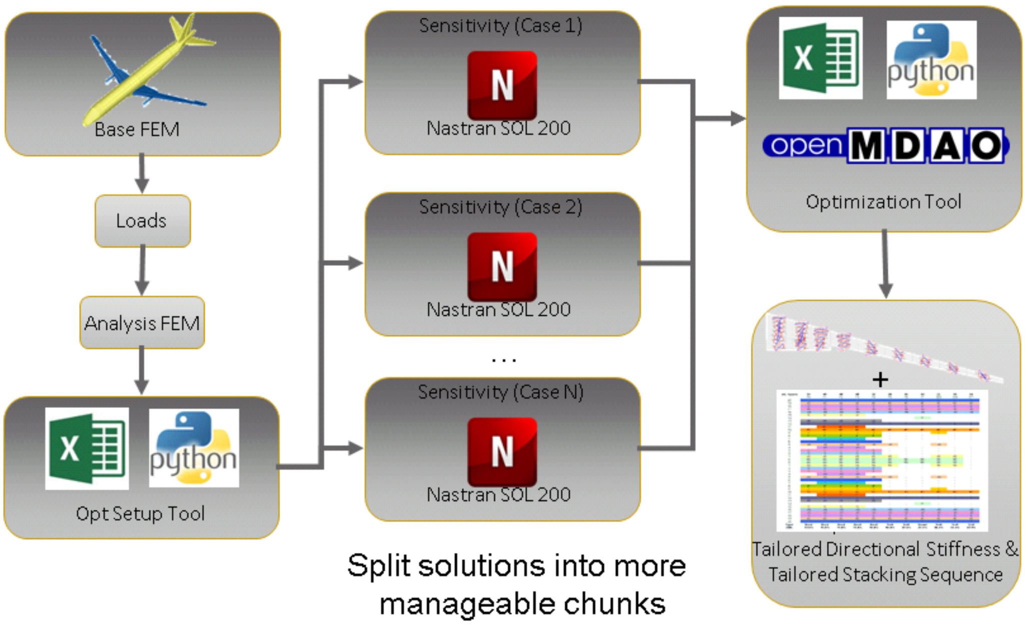

Figure 12: Workflow of structural optimization framework ModOpt

Adaptation for MDO workflow - scale-up wing design
For the scale-up wing design, a more traditional structural sizing coupled with a comprehensive loads analysis block will be performed. In this case, the full design-space in the composite optimization will be utilized in order to minimize wing structural weight.

The input to the structural optimization block include the FE model of the wing, a screened set of loads to be used for the optimization and the optimization options. The output of this block is the optimal stiffness design of the wing under the given loads. This stiffness and mass is propogated downstream via the mass and stiffness matrices.

4 Tool adaptation for aeroelasticity

4.1 CAD-Aero Interface

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.

4.2 Coupling

To carry out aeroelastic analysis, a coupling between the structural and aerodynamic models needs to be established. RBE2 elements(yellow in Figure 13) connect grids on the LRA with the leading edge and trailing edge grids to cover the whole wing surface. RBE3(yellow in Figure 13) elements connect grids on the LRA with the intersecting grids between the wing box's upper skin and ribs to distribute the aerodynamic forces onto the ribs. The same principle also applies to flaps. The whole process is carried out automatically in HyperMesh during structural modeling.

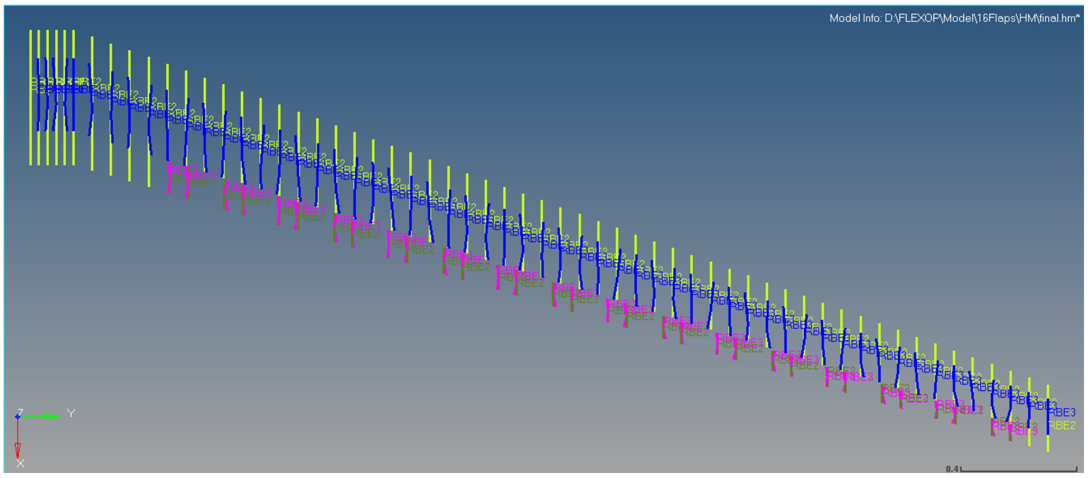

Figure 13: coupling model of the wing

4.3 Flutter calculation

By means of a flutter calculation the speed and the frequency for which the damping of an aeroelastic mode becomes zero are determined. Through linearisation of the nonlinear aeroelastic model at incrementally increasing speeds, flutter can be analysed. The open-loop poles of the linearised state-space matrices can be displayed in a root-locus plot, like shown in Figure 14. It can be seen that two poles migrate to the right half plane crossing the vertical axis with increasing speed. As soon as the poles cross the vertical axis, the corresponding modes become unstable. In Figure 14 this is the case for the poles of symmetric and asymmetric flutter. To identify the exact flutter characteristics, the damping and frequency are plotted with respect to the flight speed. It can be seen that the damping of the symmetric flutter mode crosses the zero line at approximately $48\,\mathrm{m/s}$, while the asymmetric flutter mode becomes

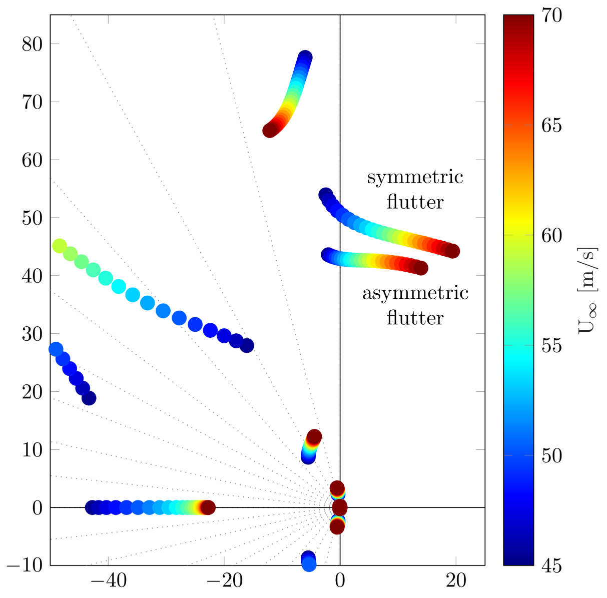

Figure 14: Root-locus plot at different speeds

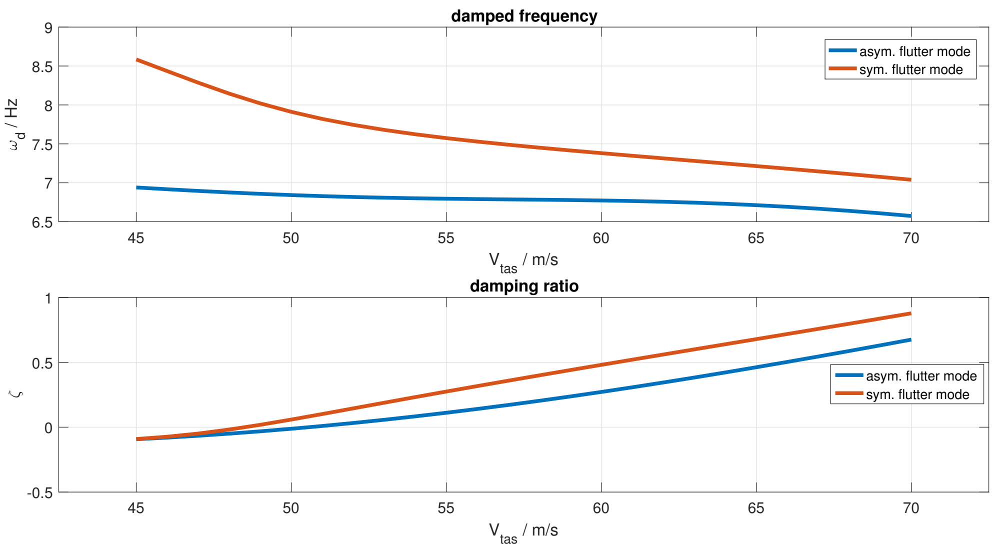

Figure 15: Damping and frequency of symmetric and asymmetric flutter

unstable at around $51\,\mathrm{m/s}$. Now it is easy to determine the flutter frequency for both modes, which is for symmetric flutter at 8.2 Hz and for asymmetric flutter at 6.8 Hz.

In this manner, flutter calculations are performed by analysis of a grid of linearised state-space models.

4.4 Loads Analysis

The loads analysis block at DLR-AE is performed applying an in-house software Loads Kernel [12]. The Loads Kernel Software allows for the calculation of quasi-steady and dynamic maneuver loads, unsteady gust loads in the time and frequency domain as well as dynamic landing loads based on a generic landing gear module. The following are excerpts from the user guide to Loads Kernel [12].

The Loads Kernel is split into three processing steps:

– pre

– main

– post

Relevant to the tool adaptation are the pre- and post-processing steps.

During the pre-processing, all required model data is read, processed and assembled to one model. Input to the pre-processing are the mass and stiffness matrices, the FE geometry and the aerodynamic panel mesh. The mass and stiffness matrices are exported from MSC.NASTRAN in the op4 file format, which is achieved with a DMAP alter. MSC.NASTRAN is only used in the role of a pre-processor.

In addition, a load case definition is needed as input. The load case definition contains parameters for every load case that is to be calculated. This typically comprises of parameters such as the type of maneuver, the mass configuration, flight speed, altitude, load factor, rates, accelerations, subcase identification number, subcase identification string, etc.

The post-processing is dedicated to the evaluation, visualization and export of the results. The dimensioning load cases are identified using established routines. For maximum compatibility, currently four different formats are supported for the export of the corresponding nodal loads:

- MSC.NASTRAN format using FORCE and MOMENT cards
- Internal, hierarchical format using the pickle module in Python
- DLR CPACS format using the Tixi XML interface library
- Matlab format using the scipy.io module in Python

5 Tool adaptation for control design

5.1 Preliminaries

Within the FLiPASED project, the Work Package 2 (WP2) is dedicated to the feedback control functions construction. The main objective of the WP is to develop a bundle of functions allowing to design the control functions in an automated manner, in order to be included in the global Multi Disciplinary Optimisation (MDO) process. This MDO being the central objective of FLiPASED. This WP involves three research groups, the DLR, ONERA and SZTAKI.

5.2 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 also. With reference to Figure 16, 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, and in the considered case, one may expect even more blending. Still the big picture remains. This sequential control structure will be kept in mind in the WP2 flow to stick to industrial and practical expectations.

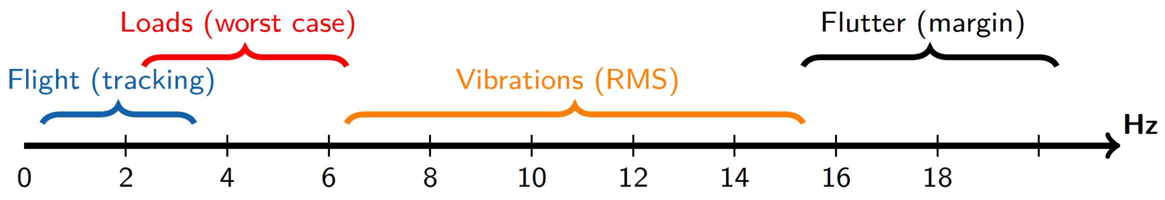

Figure 16: Frequency grid of the physical phenomena occurring over an aircraft. Ranges and values are different from an aircraft to an other.

As a matter of consequence, the closed-loops one is intended to develop is presented as in Figure 17, 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 manoeuver 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.

As presented in Figures 16 and 17, 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 loop
2. MLA, a maneuver load alleviation control loop
3. GLA, a gust load alleviation control
4. Flutter, a flutter shield control

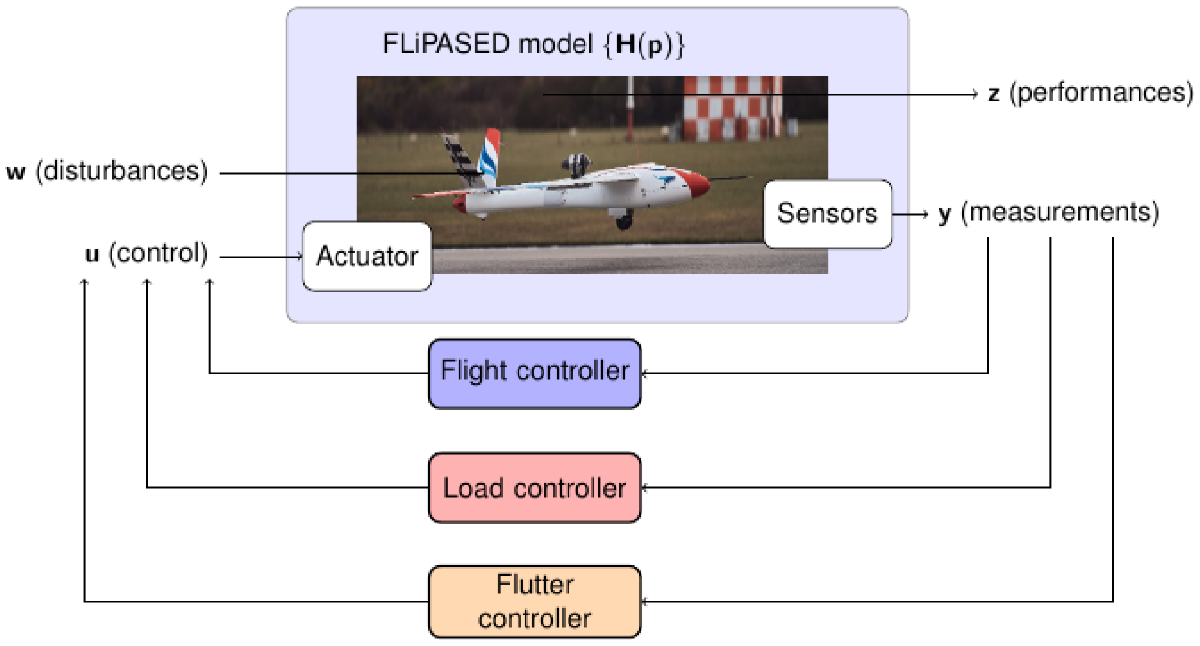

Figure 17: Multiple control loops considered in the WP2.

As all these phenomena are specific and operate at different frequencies, the models $\mathbf{H}(\mathbf{p})$ 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 obective.

5.3 Controller design

Consequently, to fit th MDO complete process, the step-by-step approach has to be somehow automatised. This will be done following the sequential approach summarised in the Table 3.

Therefore, the major tool adaptation concerns the construction of functions dedicated to each step of the process.

5.4 Control oriented model development

The structural dynamics model, the aerodynamics model and the flight mechanics model are combined to form the aeroservoelastic (ASE) model. Such subsystem interconnection is depicted in Figure 18. These ASE models in general are of too high order for control design, therefore, model order reduction is required. One approach applied for the MDO process is 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

WP	Title	Function
2.1 Modelling	Construction linear ROM	✓ Order reduction automatic guess ✓ Model order reduction from finite realisation ✓ Model order reduction from infinite realisation or transfer function
	Construction parametric ROM	✓ From bundle of LTI models
	Construction LPV ROM	✓ From bundle of LTI models
2.2 Control	Flight qualities	✓ Design using INDI ✓ Design using scheduled PID
	GLA	✓ Design using LTI control ✓ Design using LTI Modal control ✓ Design using LPV control
	MLA	✓ Design using (linear) MPC control
	Flutter control	✓ Design using LTI control ✓ Design using LTI Modal control ✓ Design using LPV control
	Others	✓ Actuators / sensors placement ✓ Wing shape control for optimal drag configuration

Table 3: Task sharing.

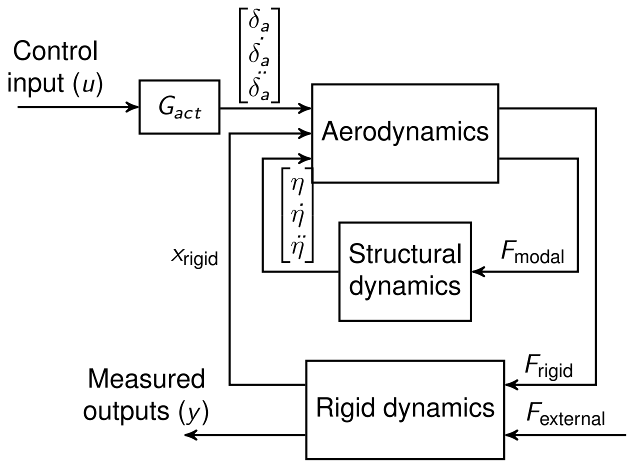

Figure 18: ASE subsystem interconnection.

the accuracy of the low order model is the ν-gap metric, [11]. The control design is using the linear parameter-varying (LPV) framework, [7, 2]. Therefore grid-based LPV models need to be obtained via Jacobian linearization.

5.4.1 Reduction of the structural dynamics model

The structural dynamics of the aircraft are of the form

$$\mathcal{M}\ddot\eta + \mathcal{C}\dot\eta + \mathcal{K}\eta = F_{\text{modal}} \tag{1}$$

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. 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 100th order model.

5.4.2 Reduction of the aerodynamics model

The aerodynamic lag terms take the state-space form

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

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 (2) 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. Retaining two lag states results in a low order model with acceptable accuracy.

The resulting nonlinear ASE bottom-up model has 56 states that consists of 12 rigid body states, 18 structural dynamics states, 2 aerodynamic lag states and 24 actuator dynamics states. The ν-gap between the nominal, high-fidelity and the reduced order model for different airspeed values is given in Figure 19.

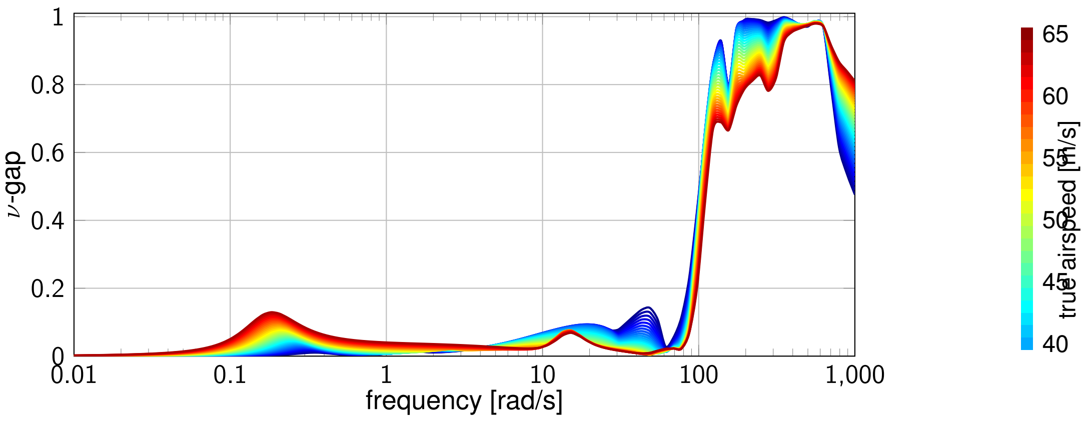

Figure 19: ν-gap values between the nominal low order and high-fidelity models.

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.

In addition to the model developed using the nominal model parameters, uncertain ASE models are also developed. Uncertainty is added to the stiffness and damping matrices of the first six modes of the structural dynamics model.

Such control oriented, uncertain LPV model serves for the flutter suppression controller. An additional model LPV model is created for the baseline control design. For this model, the structural dynamics modes and the aerodynamic lag states are residualized. Therefore, the model for the baseline control design contains only the 12 rigid body dynamics states and the actuator dynamics.

5.5 Baseline control design

The baseline control design is based on the LPV model obtained in the model integration block, that has 12 rigid body state and the actuator dynamics. 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 32 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.

5.6 Flutter control design

The flutter controller design is done based on the uncertain LPV ASE model of the aircraft. 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.

6 Performance evaluation

6.1 RCE with modeling and control design blocks

The performance evaluation is done in two steps. First, it is critical to evaluate the RCE implementation. This is presented for the modeling and control design blocks for the flutter suppression control design. The RCE implementation of these two blocks is shown in Figure 20.

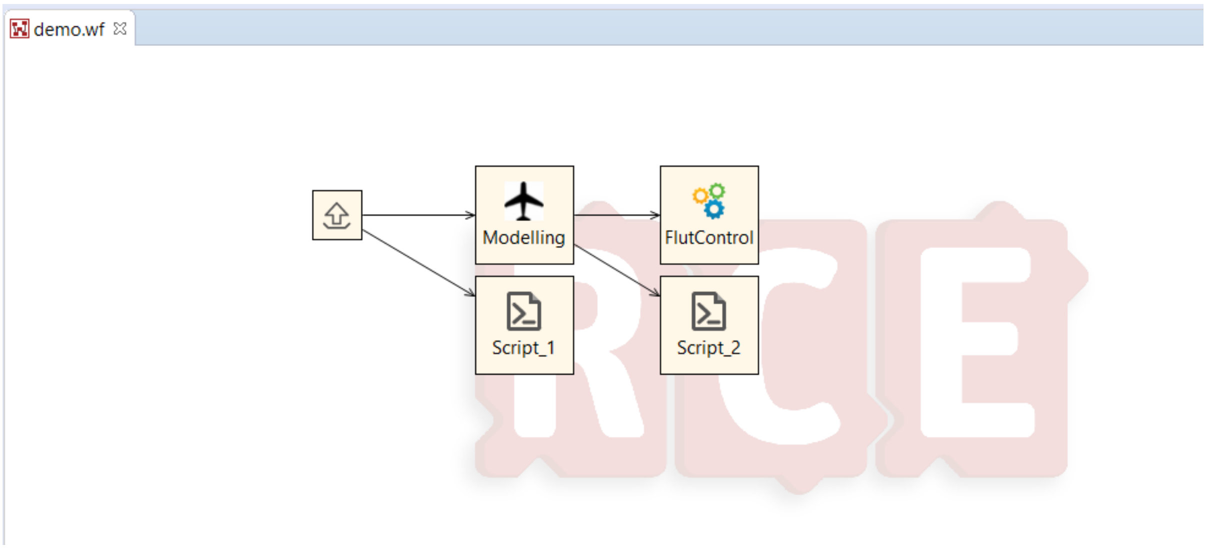

Figure 20: RCE implementation of the modeling and flutter control design blocks.

First, an 'Input Provider' is used to send the initial CPACS file, then the Modeling component start processing and sets an output based on the actual modelling script, the output is forwarded to the Flutter Controller component. Both the modeling and the flutter controller design components function with a help from external scripts which act like wrappers between them and the actual Matlab files. The scheduling between the blocks is based on the data that is the output of the preceding block. The output is set using the post-execution commands of the modeling block. The output is written in the output directory in accordance with the wrapper, so when the current block finishes, the post execution commands are executed.

All RCE block communications and data sharing needs to be specified in addition to the scheduling of the RCE blocks. The control oriented modeling blocks output files are referenced in CPACS. These output files are given in the ToolSpecific field of the CPACS xml file. The control structure specifications of the four controllers needs to be defined in advance and these need to be set up in the ToolSpecific filed as advance. Besides the main control structure, the field needs to contain the sensors used by each controller as well as the control input signals.

The second step is to evaluate the results of the control design blocks. This step is carried out for each controller individually first. For the baseline controller the first step is to evaluate if the handling qualities are satisfied or not. If this can not be achieved by the resulting controllers then the handling qualities need to be relaxed. In addition to the handling qualities, robustness, gain and phase margins of the resulting controller is evaluated. The analysis results are also written in the corresponding ToolSpecific field of the CPACS xml file. The flutter controller is also analyzed if it satisfies the robustness analysis criteria.

A crucial aspect of the performance evaluation is to verify that the controllers do not degrade each others performance when all 4 controllers are connected with the aircraft model. Such behavior of the controllers needs to be considered already in the control design process blocks. The frequency grid separation, as presented in the previous section should prevent such behavior of the controllers. Namely, the bandwidths of each controllers should be clearly separated. If such frequency separation is not possible, then a remedy could be to design the controllers with overlapping frequency grids either as integrated controllers or by successive loop closures.

Finally, time domain simulations are also checked to see if the controllers work well with the nonlinear models.

7 Conclusion

The major output of all the activities is enhanced and matured tools in each discipline, which are compatible within a collaborative design toolchain.

The adapted tools will be integrated into RCE framework by respective tool owners to build up a first version of MDO toolchain.

8 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] Thomas Klimmek. Parameterization of Topology and Geometry for the Multidisciplinary Optimization of Wing Structures. In *Proceedings - CEAS 2009*, 2009.

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

[8] 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.

[9] Gustavo H. C. Silva and Johannes K. S. Dillinger. Advancement of a Composite Wing Optimization Process - Status Report 02 / 2019, DLR - Institute of Aeroelasticity, 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] Voß, Arne. "Loads Kernel User Guide," Institut für Aeroelastik, Deutsches Zentrum für Luft- und Raumfahrt, Göttingen, Germany, Technical Report DLR-IB-AE-GO-2020-136, Oktober 2020, 2020.

[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.