COAST - A SIMULATION AND CONTROL FRAMEWORK TO SUPPORT MULTIDISCIPLINARY OPTIMIZATION AND AIRCRAFT DESIGN WITH CPACS

Daniel Kiehn[1], Johannes Autenrieb[1] & Nicolas Fezans[1]

[1]DLR (German Aerospace Center), Institute of Flight Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany

Abstract

This paper describes COAST (CPACS-oriented aircraft simulation tool), a fixed-wing aircraft simulation framework tailored to an XML-based open source common exchange format for multidisciplinary aircraft design, called CPACS (Common Parametric Aircraft Configuration Schema). COAST enables designers to utilize flight simulation on desktop computers or even on full motion simulators in early stages of aircraft design, which facilitates the assessment of flight characteristics and handling qualities as well as early flight control design. The core model of COAST is presented along with its interface to the data exchange format CPACS, which is implemented via import functions based on XML parsers. Due to the necessity of generically working autopilot functionalities, a nonlinear flight control concept based on the idea of the nonlinear model following control (NMFC) methodology is proposed. In order to handle novel aircraft configurations with a redundant set of control effectors, an optimization-based control allocation module is integrated. The process of integrating the model into the German Aerospace Center's full motion flight simulator AVES (Air Vehicle Simulator) in Braunschweig is discussed.

Keywords: CPACS, multidisciplinary optimization, full flight simulation, nonlinear model-following control

1. Introduction

Collaborative design and multidisciplinary optimization (MDO) are applied in an increasing number of projects in the aerospace industry and academia [1, 2]. By connecting different knowledge disciplines and software tools, MDO holds the potential to significantly reduce aircraft development costs and time to market, and it facilitates more innovative solutions and unconventional or disruptive concepts by enabling an optimized integration on system level [2]. In the past decade, the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR) has been employing and continuously developing an integration environment to connect software tools of different disciplines into more extensive, multidisciplinary process chains, called Remote Component Environment (RCE). The individual tools available in the RCE framework are provided via servers located at different DLR sites [3]. To simplify the exchange between the different MDO tools, DLR developed the common exchange format CPACS (Common Parametric Aircraft Configuration Schema) [1, 4]. CPACS is an open-source project which is freely available on www.cpacs.de under the Apache license 2.0. It is primarily developed by DLR, but any interested party is welcome to use CPACS or to contribute to its development. By using a common data exchange model such as CPACS, the number of required interfaces between tools can be significantly reduced, as shown in Figure 1: if all N tools communicate with each other via individual interfaces, the potential number of required interfaces is $N(N-1)$, whereas this number is only $2N$ if all tools use the same standard for data exchange. CPACS is developed on the basis of the Extended Markup Language (XML) and is being used internationally in multiple collaborative projects [2, 5]. CPACS files can generally be accessed with any XML parser, but there are also two individual libraries specifically tailored to CPACS: TiXI (TIVA XML Interface) [6], which is a generic XML parser extended by methods to account for CPACS specific conventions (such as reading and writing vectors or multidimensional arrays), and TiGL (TIVA geometric library) [7], which is designed to read and interpret geometric data of a CPACS file, for example to calculate the surface area of a wing or the cross section of the fuselage. Another functionality of TiGL is the export of the three-dimensional model geometry to IGES/STEP files, which can be interpreted with most CAD (computer-aided design) software.

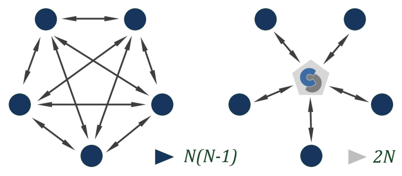

Figure 1 – Reduction of the possible number of interfaces by using a central data format [1]

The usefulness of stability and control analysis in early stages of aircraft design was shown in [8–11]. In [12], stability and control analyses were performed for two different aircraft configurations described in the CPACS format. The next logical step for the evaluation of flying characteristics is the pilot-in-the-loop flight simulation. Especially for unconventional aircraft configurations, flight simulation enables the identification of potential controllability or stability issues already in early stages of aircraft design, as was shown in various examples: in [13], flight simulation was used to assess the controllability of a remotely piloted high-altitude platform. In [14], the flight characteristics of an aircraft configuration with distributed electric propulsion were investigated. When flown in a full-motion flight simulator, severe differences in its flying characteristics compared to conventional aircraft were noticeable, which were caused by the propeller slipstream effects of the twelve electrical engines distributed along the wing. The authors derived from their observations that the discovered problems for this kind of aircraft could be mitigated with new specially tailored flight control laws [14]. The research showed again the considerable benefit of stability and control investigations and piloted flight simulations in the early stages of aircraft design.
To ease the generation of flight simulation software for aircraft configurations described in the CPACS format, DLR developed the simulation framework COAST (CPACS-oriented Aircraft Simulation Tool). COAST offers the potential to be used as the core flight mechanics model for stability and control analysis, but can also be used for pilot-in-the-loop flight simulations due to its interface with DLR's full motion cockpit flight simulator platform AVES (Air Vehicle Simulator). Since a CPACS file can, in principle, hold the definition of an aircraft with an arbitrary level of fidelity and complexity due to the hierarchical data structure of CPACS [12], COAST can be used in different stages of aircraft design. The usage of COAST to enhance a generalized MDO toolchain is shown in Figure 2.

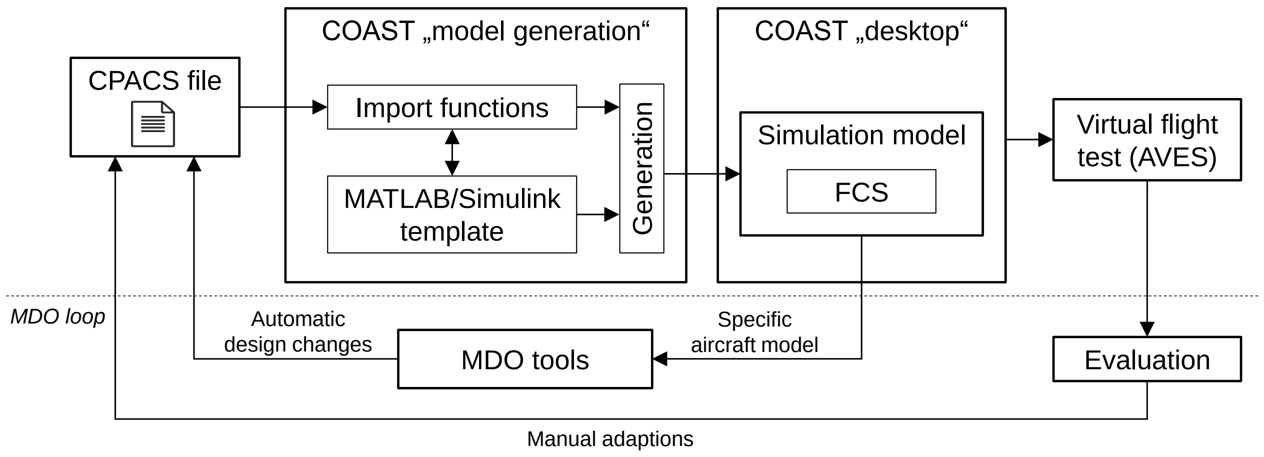

Figure 2 – Enhancing an MDO toolchain with COAST

In order to enable controlled flight also for unstable aircraft configurations and to achieve a certain degree of comparability between aircraft configurations, a flight control system with automatically

tuned gains and a central control allocation algorithm is developed.
The structure of the COAST model, including the import process of CPACS data, is introduced in section 2. In section 3, the developed flight control system is presented, and section 4 discusses the integration of COAST into the flight simulator AVES.

2. Model structure

COAST is mainly implemented in MATLAB®/Simulink® R2007b and is upwards compatible up to the most recent release (R2022a). In the initialization phase of COAST, all data required to run COAST are imported from the CPACS file via so-called *wrapper functions*. Each field (e.g., propulsion, aerodynamics) has its own designated wrapper function to specifically parse and import the corresponding data into MATLAB and bring it into the required format. The wrapper functions rely on the TiXI and TiGL libraries and their respective MATLAB interfaces to read out the relevant information from the CPACS file.
Since the Simulink model has to function for any given aircraft configuration, tailoring certain model components to the aircraft configuration can be cumbersome (e.g., the number of aerodynamic control surfaces and corresponding actuators may generally vary between models), hence there would be generally two main options for a pure Simulink implementation of COAST: in option 1, the model would have a pre-defined structure and be designed for a fixed number of engines, control surfaces etc., and unnecessary blocks would be filled with dummy data. This option has two drawbacks: the model cannot be used if the aircraft exceeds the pre-defined structure (e.g., by exceeding the number of modeled engines), and the simulation of dummy devices such as superfluous control surfaces would cause unnecessary computational load. In option 2, the Simulink model would be created on-the-fly using MATLAB scripts, utilizing MATLAB's ability to programmatically add/remove/change blocks in Simulink models. This option provides more flexibility, but it makes the model creation computationally more demanding and adds complexity due to the required signal routing (e.g., mapping of bus creators and bus selectors). It was hence decided to not use a pure MATLAB/Simulink solution, but to use an alternative option by implementing all model components whose structure is directly dependent on CPACS (i.e. the aerodynamics module, the propulsion module, and the nonlinear dynamic inversion used for control allocation and controller design) in C++ with S-function interfaces to the encompassing Simulink model.[1] Using S-functions also yields a slight performance gain compared to the pure Simulink implementation, at least on desktop computers.
In the following sections, the individual modules of the COAST model and their interactions with the corresponding CPACS data content are described. CPACS is under constant development and the stated information is hence partially dependent on the CPACS standard. In this paper, the most recent CPACS standard (version 3.4) is used as a reference unless stated otherwise.

2.1 Common modules

While some components of the COAST model are specifically tailored to CPACS, several modules are implemented generically and do not share any direct interface to the CPACS structure, as shown in Figure 3. These CPACS-independent modules include the 6-degrees-of-freedom (6-DoF) equations of motion, an atmospheric model which is an implementation of the US Standard Atmosphere model from 1976 [16], and the geodetic reference system which is modeled after the World Geodetic System 1984 (WGS84) [17]. In the current version of COAST, the equations of motion assume a rigid body, although they might be extended to include flexible degrees of freedom in a future update since the required data can generally be provided via the CPACS standard [18].

The CPACS standard does not provide any means to specify sensors on the aircraft. In order to still achieve a certain degree of realism, COAST relies on default models for an inertial navigation system, an air data system (consisting of a pressure sensor, an altitude sensor, and angle of attack and angle of sideslip sensors), and an altitude radar. These sensor models are implemented as second-order systems with configurable gains, bandwidths and biases. Due to the lack of corresponding data in CPACS, their respective positions on the aircraft follow standard positions on airliner aircraft. This

[1] An S-function is a computer language description of a Simulink block written in MATLAB®, C, C++, or Fortran [15].

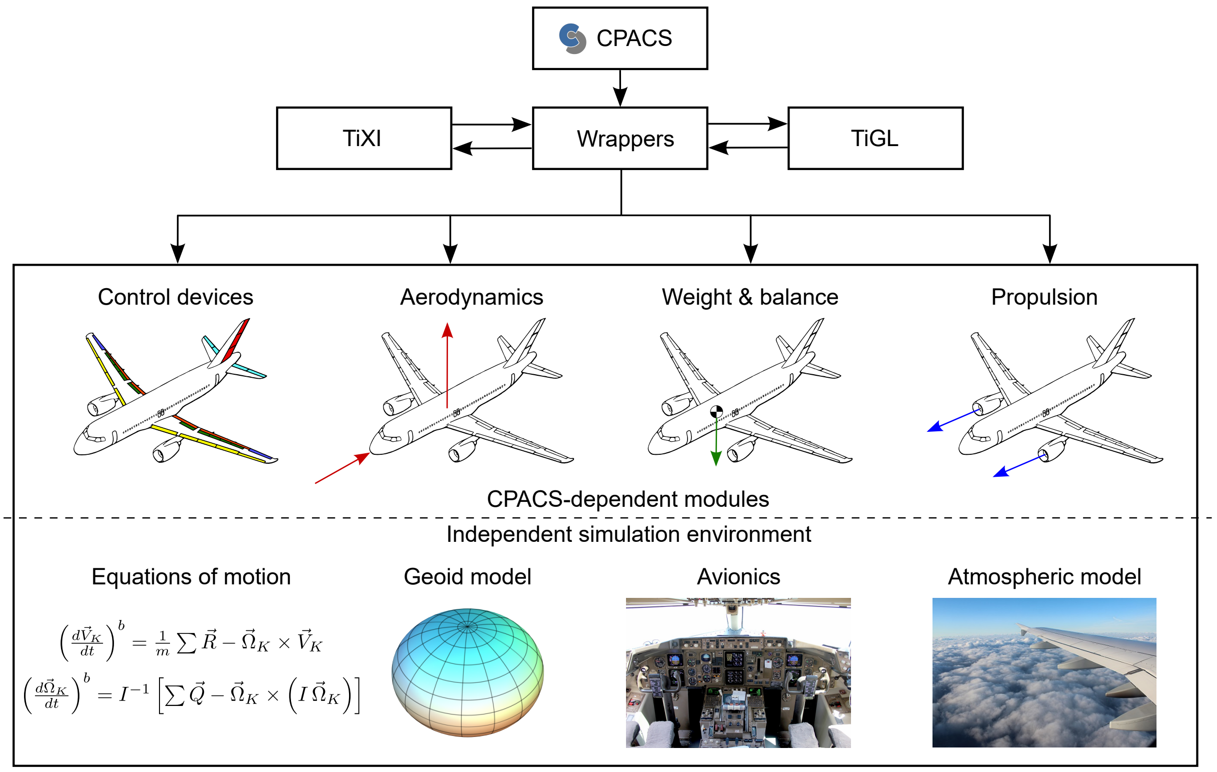
CPACS
TiXI
Wrappers
TiGL
Control devices
Aerodynamics
Weight & balance
Propulsion
CPACS-dependent modules
Independent simulation environment
Equations of motion
$\left(\frac{d\vec{V}_K}{dt}\right)^b = \frac{1}{m}\sum\vec{R} - \vec{\Omega}_K \times \vec{V}_K$
$\left(\frac{d\vec{\Omega}_K}{dt}\right)^b = I^{-1}\left[\sum\vec{Q} - \vec{\Omega}_K \times \left(I\,\vec{\Omega}_K\right)\right]$
Geoid model
Avionics
Atmospheric model

Figure 3 – COAST model structure

coarse degree of realism is assumed to suffice for simulation in early stages of aircraft design and can be enhanced manually by adding more sensors or tuning the sensor characteristics and adapting the positions, if necessary.
COAST does not yet include a model for the landing gear, partially because the implementation of a realistic gear model requires a high degree of detail which the CPACS format does not yet provide. If a landing gear model is required, for example for simulated landings, a custom model for the individual aircraft can be produced as discussed in [14].
Since the CPACS standard does not yet allow for the definition of actuator dynamics, COAST uses second-order models with default values for damping and angular frequency for all aerodynamic control surface actuators. The respective deflection limits are provided in the CPACS file, and default values for rate and acceleration limits are used. As is the case for the sensors or the landing gear, these default values can be tuned to more closely match an individual aircraft if a higher degree of realism is needed.

2.2 Open-loop (direct law) control inputs

In CPACS, the chain of pilot control inputs to aerodynamic control surface deflections is defined via a four-level control mixing hierarchy. The first level are the *cockpitControls*, which represent the controls directly accessible by the pilot. Each *cockpitControl* is linked to a *commandCase* via an input and an output vector, where linear interpolation is applied to obtain intermediate values. Each *commandCase* is then connected to a *controlDistributor* via a multiplicative gain. The *controlDistributor* describes an input/output relationship from the *commandCase* input to nondimensional deflections of each connected aerodynamic control surface. Again, linear interpolation is used to determine intermediate values.
The four-level control mixing hierarchy as defined in CPACS can be well explained using the example of the UCAV (unmanned combat aerial vehicle) MULDICON (Multi-disciplinary configuration) which was designed in the DLR research project Mephisto [19]. The control setup of this aircraft is shown in Figure 4, and the specific control mixing hierarchy of the MULDICON aircraft is shown in Figure 5. The exact mapping of control distributors to control surfaces is given in Table 1. Positive deflections of the inner and outer flaps correspond to downward deflections, while a full positive deflection of the

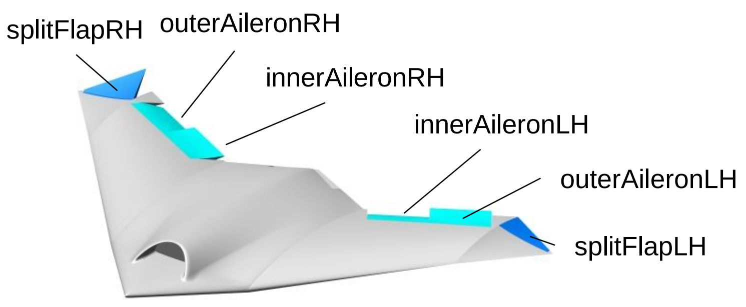

Figure 4 – Aerodynamic control surfaces of the MULDICON aircraft [19]

split flaps represents a fully split state. For example, a full deflection of the *rollDistributor* (indicated by a deflection of “1”) corresponds to a deflection of -1 of the right outer aileron, 0.8 of the left outer aileron, -0.5 of the right inner aileron, and 0.4 of the left inner aileron.

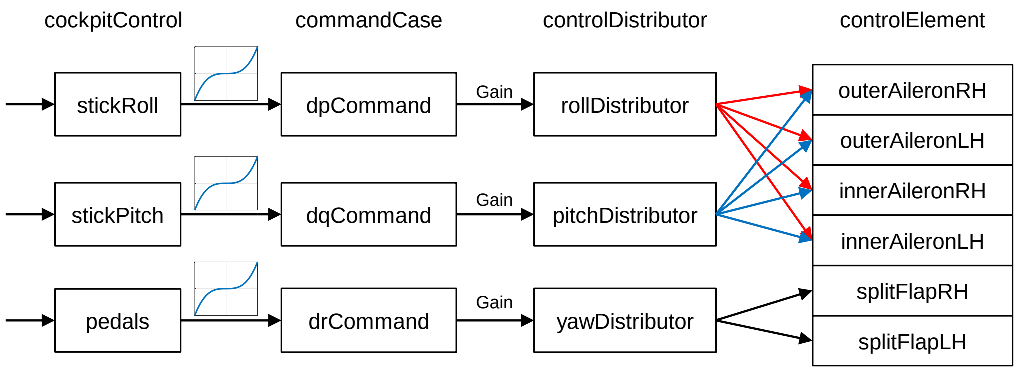

Figure 5 – Mapping of open-loop controls from cockpit controls to aerodynamic control surfaces

Table 1 – Mapping of control distributors to aerodynamic control surfaces of the MULDICON

Control surface	rollDistributor $[-1,0,1]$	pitchDistributor $[-1,0,1]$	yawDistributor $[-1,0,1]$
outerAileronRH	$[0.8,0,-1]$	$[-1,0,1]$	–
outerAileronLH	$[-1,0,0.8]$	$[-1,0,1]$	–
innerAileronRH	$[0.4,0,-0.5]$	$[-1,0,1]$	–
innerAileronLH	$[-0.5,0,0.4]$	$[-1,0,1]$	–
splitFlapRH	–	–	$[0,0,1]$
splitFlapLH	–	–	$[1,0,0]$

In COAST, this chain of control inputs is only used when flying in direct law. The more complex control laws, which are based on standard control laws as used by Airbus, are presented in section 3.2.

2.3 Aerodynamics

The aerodynamic data in CPACS is stored in multidimensional lookup tables, and intermediate values can be determined via linear interpolation. The newer versions of the CPACS standard (3.0 and upwards) are able to support multiple aerodynamic data maps for the same aircraft. This feature adds additional flexibility to the overall aircraft MDO design. However, COAST is not yet able to switch between different maps during a simulation, and therefore the model can only be compiled for a single map at a time. When using COAST, the user has to define which aerodynamic data map is to be imported using either the *uID* (unique identifier) of the map or its numerical index within the CPACS file.

Each aerodynamic coefficient $C_{(.)}$ is calculated as a sum of three contributions: the basic coefficient $C_{(.),base}$ which includes the aircraft with undeflected control surfaces and without dynamic motion, the contributions of dynamic derivatives $\Delta C_{(.),dyn}$, and the contributions of the control surfaces ΔC_{CS}:

$$C_{(.)} = C_{(.),base} + \Delta C_{(.),dyn} + \Delta C_{(.),CS} \tag{1}$$

where the index (.) is a generic placeholder for the force or moment the coefficient designates: *d* for the drag force, *s* for the side force, *l* for the lift force, and *md/ms/ml* for the moments about the respective axes. The designations used here follow the convention of CPACS 3.4. All coefficients are defined w.r.t. the moment reference point $\vec{r}_{rp}$ and in a frame denoted here by $\tilde{a}$, which is rotated by 180° around the y-axis compared to the aerodynamic frame as defined in the ISO 1151-1 standard [20], denoted by a, in the x and z direction:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = \begin{pmatrix} -x_{\tilde{a}} \\ y_{\tilde{a}} \\ -z_{\tilde{a}} \end{pmatrix} \tag{2}$$

The angle of attack $\alpha \in (-\pi, \pi]$ and the angle of sideslip $\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ used in COAST follow the standard definition in flight mechanics:

$$\alpha = \operatorname{atan2}(w_a, u_a) \tag{3}$$

$$\beta = \arcsin \frac{v_a}{V} \tag{4}$$

where $\operatorname{atan2} : \mathbb{R}^2 \to (-\pi, \pi]$ is the four-quadrant inverse tangent function.
The base coefficient $C_{(.),base}$ is a function of the angle of attack α, the angle of sideslip β, the altitude H, and the Mach number Ma:

$$C_{(.),base} = f(\alpha, \beta, H, \mathrm{Ma}) \tag{5}$$

The contributions of the dynamic derivatives $\Delta C_{(.),dyn}$ can be expressed as

$$\Delta C_{(.),dyn} = C_{(.),p} p^*_{\tilde{a}} + C_{(.),q} q^*_{\tilde{a}} + C_{(.),r} r^*_{\tilde{a}} \tag{6}$$

where $(p^*_{\tilde{a}}, q^*_{\tilde{a}}, r^*_{\tilde{a}})^T$ are the nondimensional rotational rates of the aircraft w.r.t. the $\tilde{a}$ frame:

$$\begin{pmatrix} p^*_{\tilde{a}} \\ q^*_{\tilde{a}} \\ r^*_{\tilde{a}} \end{pmatrix} = \frac{l_\mu}{V} \begin{pmatrix} p_{\tilde{a}} \\ q_{\tilde{a}} \\ r_{\tilde{a}} \end{pmatrix} \quad , \tag{7}$$

with $(p_{\tilde{a}}, q_{\tilde{a}}, r_{\tilde{a}})^T$ being the rotational rates of the aircraft w.r.t. the $\tilde{a}$ frame, and:

$$C_{(.),p} = \frac{\partial C_{(.)}}{\partial p_{\tilde{a}}} = g_1(\alpha, \beta, H, \mathrm{Ma}) \tag{8}$$

$$C_{(.),q} = \frac{\partial C_{(.)}}{\partial q_{\tilde{a}}} = g_2(\alpha, \beta, H, \mathrm{Ma}) \tag{9}$$

$$C_{(.),r} = \frac{\partial C_{(.)}}{\partial r_{\tilde{a}}} = g_3(\alpha, \beta, H, \mathrm{Ma}) \tag{10}$$

The structure of CPACS does not yet account for any cross-coupling effects between different control surfaces. The effect of the control surface deflections $\Delta C_{(.),CS}$ is hence described as a sum of contributions from each individual control surface i:

$$\Delta C_{(.),CS} = \sum_{i=1}^{n} \Delta C_{(.),CS,i} \tag{11}$$

where i is the control surface index, n is the total number of control surfaces, and $\Delta C_{(.),CS,i}$ is defined by

$$\Delta C_{(.),CS,i} = h(\delta_i, \alpha, \beta, H, \mathrm{Ma}) \tag{12}$$

with the deflection angle δ_i.
The vector of aerodynamic forces expressed in the body frame $\vec{R}_A^b$ is thus obtained via:

$$\vec{R}_A^b = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_A^b = \bar{q}S \underbrace{\begin{pmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{pmatrix}}_{M_{ba}} \underbrace{\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}}_{M_{a\tilde{a}}} \begin{pmatrix} C_d \\ C_s \\ C_l \end{pmatrix} \tag{13}$$

where $\bar{q}$ is the dynamic pressure, S is the reference area, and M_{ba} is the transformation matrix from the aerodynamic frame (index a) to the body frame (index b).
The vector of aerodynamic moments in the body frame w.r.t. the moment reference point $\vec{r}_{rp}$ is obtained via:

$$\vec{Q}_{A,rp}^b = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_{A,rp}^b = \bar{q}Sl_\mu M_{ba} M_{a\tilde{a}} \begin{pmatrix} C_{md} \\ C_{ms} \\ C_{ml} \end{pmatrix} \tag{14}$$

where l_μ is the reference length. The vector of aerodynamic moments in the center of gravity $\vec{Q}_{A,cg}^b$ is then calculated as:

$$\vec{Q}_{A,cg}^b = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_{A,cg}^b = \vec{Q}_{A,rp}^b + (\vec{r}_{rp} - \vec{r}_{cg})^b \times \vec{R}_A^b \tag{15}$$

2.4 Propulsion

CPACS provides the ability to store all relevant information about the engines in two distinct nodes. In one node, the relevant information for each engine regarding the aircraft integration is stored, such as the designated position, an optional *rotation* node to define tilt angles, and a reference to an engine type via an *engineUID* identifier. More detailed information about the engines are further stored in a separate node. In this node, detailed information about the thrust T and the corresponding fuel mass flow of each engine type is stored in three-dimensional lookup tables as a function of the Mach number, the altitude, and the throttle lever position. At the current point, CPACS does not support any propeller-wing interactions or further moments due to the rotational inertia of the engines. If the modeling of such effects is required, the implementation has to be made independent of the CPACS standard: this was done for example in the DLR project SynergIE, where COAST was enhanced by an aerodynamics/propulsion interaction module to account for the propeller slipstream effect on the lift of the investigated aircraft [14]. Since CPACS does not include a way to define engine dynamics, COAST uses second-order models for the thrust with default values for damping and angular frequency which include limits for position, rate, and acceleration.
The combined vector of moments around the center of gravity produced by all engines is obtained by:

$$\vec{Q}_{P,cg}^b = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_{P,cg}^b = \sum_{i=1}^{n_E} (\vec{r}_i - \vec{r}_{cg})^b \times \begin{pmatrix} X_{P,i} \\ Y_{P,i} \\ Z_{P,i} \end{pmatrix}^b \tag{16}$$

where $\vec{r}_i$ is the position vector of the i-th engine in body-fixed axes, $\vec{r}_{cg}$ is the center of gravity, and n_E is the total number of engines. The forces X, Y, and Z represent the projection of the thrust force T into the $x/y/z$ axis, respectively. In case the thrust axis is aligned with the body-fixed x-axis of the aircraft, these projections are simply $X_{P,i} = T_i$ and $Y_{P,i} = Z_{P,i} = 0$.

2.5 Toolboxes and post-processing

COAST provides users with a broad set of flight dynamical and simulation-based analysis tools. For example, a trimming function provides a simple way to trim the regarded vehicle at a user-defined operating point while considering the desired flight conditions (such as steady, horizontal flight or a constant turn). After the vehicle is trimmed, the provided linearization functionalities can be used to linearize the trimmed system around the equilibrium point. Stability and control derivatives can be extracted from the resulting linear system using designated functions. Among the obtainable derivatives are, for example, classic stability derivatives such as $C_{m\alpha}$ and $C_{n\beta}$, but also controllability quantities such as the lateral control divergence parameter (LCDP) [21]. In order to enable easy and unambiguous access of the model's states, inputs, and outputs, COAST provides the user with corresponding lists of names which allow direct access to each variable via its name: e.g., the user can directly access the pitch angle via the state name *Theta* instead of having to use fixed indices such as "x[2]".

3. Control design

The design and validation of flight control systems for the full flight envelope of an aircraft is a time-consuming task and, in the context of aircraft pre-design projects, controllers have to be constantly adapted to each configuration change since the underlying algorithms are mostly based on linear system assumptions and hence are only valid for specific operating points. This leads to the need of control design workflows in which the system needs to be trimmed, linearized and tuned for each operating point within the defined flight envelope. Later, the different control configurations needed during the flight are connected by utilizing gain-scheduling approaches. Especially for the integration of simulator capabilities into the multidisciplinary preliminary design workflow, a flight control system is needed that is generic enough to allow early simulator test flights for knowledge acquisition. At the same time, the flight control system needs to be customizable enough to be able to augment certain desired system dynamics and response behaviors.
In order to perform pilot-in-the-loop scenarios, it is often required to include control laws or autopilot functions. For instance, in the framework of the DLR project SynergIE an electrical distributed propulsion aircraft with twelve propellers, shown in Figure 6, was brought to AVES [14]. Such configurations might exhibit uncommon flight characteristics, as was the case in this example: the pilot-in-the-loop simulations conducted in the AVES revealed that the aircraft showed a very strong thrust/lift interaction, which led the authors to conclude a potential need of completely new flight control laws for this kind of aircraft [14].
Since the main use of the CPACS format is for preliminary design studies of possibly uncommon configurations, COAST is currently being extended to ease as much as possible the design of such flight control functions for the generated models. The currently considered flight control architecture is discussed in the following subsections.

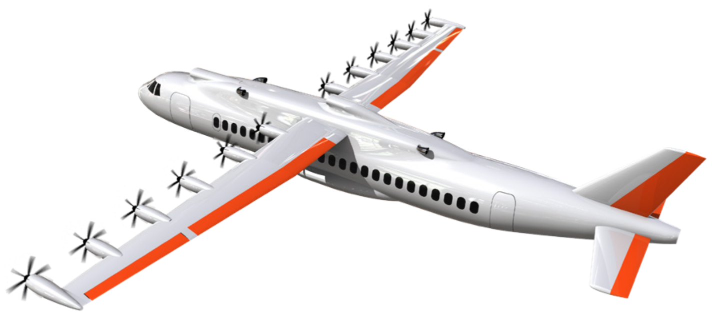
Figure 6 – Final configuration of the SynergIE aircraft as simulated in AVES [14]

3.1 Control allocation

The inner loop of the attitude controller controls the three rotational degrees of freedom of the aircraft using only the aerodynamic control surfaces. If the number of available control surfaces n is equal to

the number of degrees of freedom to be controlled, there is only one combination of control surface deflections which achieves the desired control effort (given it can be achieved at all). If the number of available control surfaces is higher (i.e., $n > 3$), and if each degree of freedom is affected by at least one independent control surface, the control surfaces are redundant and the desired control effort can be achieved with an infinite number of control surface deflection combinations.
The main task of the control allocation module is to determine the optimal way of distributing the control effort within the redundant set of control surfaces to produce the moments required to follow the rotational accelerations $(\dot{p},\dot{q},\dot{r})_c^T$ commanded by the flight control system. The benefit of a central control allocation module in COAST is that the encompassing flight controller can be implemented generically and independently of the setup of control surfaces on the individual aircraft configuration in the CPACS file. Using the desired angular accelerations of the aircraft $(\dot{p},\dot{q},\dot{r})^T$ as the input to the control allocation enables the flight control design to focus purely on the control laws, while the desired response of the aircraft can be shaped independently of the aircraft configuration via the angular accelerations, which significantly simplifies the control design process.
The required deflections can be obtained by inverting the aerodynamic model and the rotational equations of motion, as will be explained in the following. In the first step, the vector of required (i.e., commanded) total moments $\vec{Q}_c$ is calculated from the angular accelerations $(\dot{p},\dot{q},\dot{r})_c^T$ commanded by the flight control system:

$$\vec{Q}_c = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_c = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix}_c + \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times I \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{17}$$

where $(p,q,r)^T$ are the rotational rates of the aircraft and I is the inertia matrix w.r.t. the center of gravity in body axes. Since the control allocation uses only the aerodynamic control surfaces to achieve the commanded moments, the required aerodynamic moments $\vec{Q}_{A,\text{req}}$ have to exactly compensate the difference between the commanded moments and the contributions of the engines (index P):

$$\vec{Q}_{A,\text{req}} = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_{A,\text{req}} = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_c - \begin{pmatrix} L \\ M \\ N \end{pmatrix}_P \tag{18}$$

Note that there is no contribution of gravitational forces because the formulation is defined in the center of gravity. The goal of the control allocation is to find the set of deflections δ which achieves $\vec{Q}_A = \vec{Q}_{A,\text{req}}$, or in other terms, which minimizes the Euclidean norm of the residual vector $\tilde{r}$:

$$\tilde{r} = \begin{pmatrix} L \\ M \\ N \end{pmatrix}_A - \left[\begin{pmatrix} L \\ M \\ N \end{pmatrix}_c - \begin{pmatrix} L \\ M \\ N \end{pmatrix}_P \right] \tag{19}$$

As explained in section 2.3, the aerodynamic forces and moments are a function of the angle of attack α, the sideslip angle β, the altitude H, the Mach number, the rotational rates $(p,q,r)^T$, and the set of control surface deflections δ. While δ is the free variable of the optimization, the other dependencies are fixed parameters (from the perspective of the allocation module; they can still be used as control variables by the encompassing flight controller) and are therefore summarized by the parameter set $P = \{\alpha, \beta, H, \text{Ma}, p, q, r\}$. The resulting optimization problem is constrained, since the deflections δ must be within their respective lower and upper deflection limits $\underline{\delta}$ and $\overline{\delta}$. The resulting nonlinear least-squares optimization problem hence can be written as:

$$\begin{gathered} \delta_{\text{opt}} = \underset{\delta}{\operatorname{argmin}} \, \|\tilde{r}(P,\delta)\| = \underset{\delta}{\operatorname{argmin}} \left(\sum_{k=1}^{3} (\tilde{r}_k(P,\delta))^2 \right) \\ \text{such that:} \quad \forall i \in [1,n], \quad \underline{\delta_i} \leq \delta_i \leq \overline{\delta_i} \end{gathered} \tag{20}$$

where n is the number of control surfaces. Such an optimization problem can be solved with various well-established methods. In the allocation module used in COAST, this optimization is solved iteratively using an enhanced Gauss-Newton algorithm with the following iteration rule:

$$\delta^{(k+1)} = \delta^k - \underbrace{\left(J^T J\right)^{-1} J^T}_{J^+} \tilde{r}(P, \delta^k) \tag{21}$$

where k is the iteration index and J is the Jacobian matrix, which describes the sensitivities of each residual component w.r.t. all individual control surface deflections:

$$J = \begin{pmatrix} \frac{\partial}{\delta_1} L_A & \frac{\partial}{\delta_2} L_A & \frac{\partial}{\delta_3} L_A & \dots & \frac{\partial}{\delta_n} L_A \\ \frac{\partial}{\delta_1} M_A & \frac{\partial}{\delta_2} M_A & \frac{\partial}{\delta_3} M_A & \dots & \frac{\partial}{\delta_n} M_A \\ \frac{\partial}{\delta_1} N_A & \frac{\partial}{\delta_2} N_A & \frac{\partial}{\delta_3} N_A & \dots & \frac{\partial}{\delta_n} N_A \end{pmatrix} \tag{22}$$

The presented form of the pseudoinverse J^+ (also known as Moore-Penrose inverse) causes the algorithm to find the solution with the smallest Euclidean norm out of all possible solutions (i.e., the one with the smallest control surface deflections in total). The Jacobian matrix is determined via numerical linearization of the aerodynamic model and has to be re-computed in every iteration, since it is a function of the control surface deflections δ. The aforementioned enhancement of the Gauss-Newton method is the implementation of an *active set* logic as described in [22]: If the limits of an individual control surface are reached during the iteration, this element is taken out of the optimization process until it can contribute to the control effort within its valid deflection range again.

3.2 Control laws

As previously discussed, automated simulation-based analysis methods are a practical approach for evaluating the stability and control properties in the early design stages. This approach helps identify wrong design decisions in the early development phases and consequently minimizes overall development costs. Especially for novel aircraft configurations, additional flight control systems are sometimes needed because the open-loop characteristics show poor handling qualities and stability attributes. Further, these systems are occasionally needed to provide certain autopilot modes to allow early-stage evaluations of specific parameters during certain flight operational procedures, such as the noise emission during the approach or landing of an aircraft. Nevertheless, developing suitable flight control systems can often be tedious work and would conflict with the idea of easily performable virtual flight tests. In order to cope with this problem, it was decided to facilitate the implementation of generalized autopilot modes based on automatically generated flight control laws. Figure 7 shows the overall conceptual architecture of the currently partially integrated flight control system concept and its interface to the pilot via the autopilot cockpit interface.

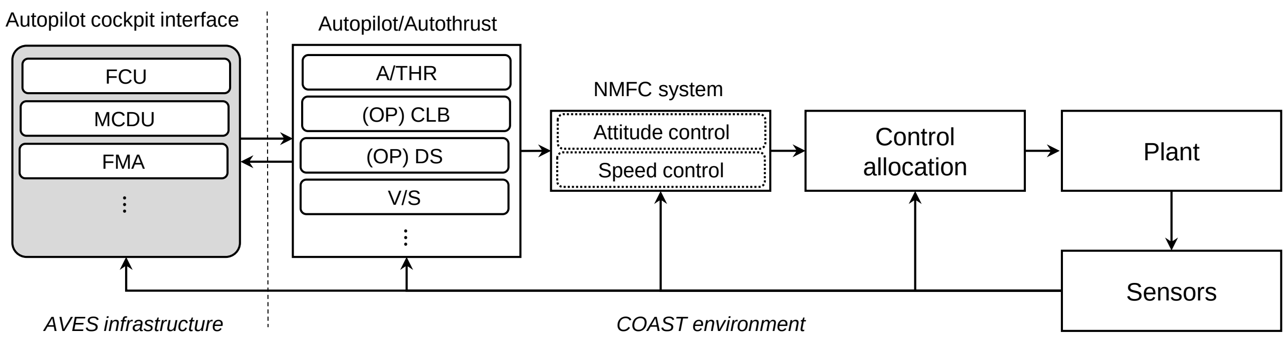

Figure 7 – Overview of the COAST flight control system

The autopilot cockpit interface is provided by the AVES infrastructure and contains the flight control unit (FCU), multipurpose control and display unit (MCDU), and the flight mode annunciator (FMA). The integrated autopilot functionalities, such as the autothrust ("A/THR"), (open) climb ("(OP) CLB"), (open) descent ("(OP) DS"), and vertical speed mode ("V/S"), match with the autopilot modes known from civil Airbus aircraft [23, ch. DSC-22_30]. The given autopilot functions can be developed so that mostly no explicit model knowledge is needed beforehand. This simplifies the development of the functionalities and generalizes the developed modules for a broad set of different aircraft. For example, most autopilot modes only depend on kinematic relationships, which are identical for all airplanes. For cases where explicit model knowledge is needed, relevant information can usually be either directly accessed over the CPACS interfaces or by using additional pre-processing routines within the COAST framework. Using such a generalized autopilot approach allows centralizing the use of model knowledge for the derivation of the inner-loop flight control laws.

The control strategy is based on the idea of nonlinear model-following control (NMFC) and is displayed in Figure 8. The NMFC controller is designed to allow the tracking and regulation of the bank angle Φ_c, the body-fixed acceleration in z-direction $n_{z,c}$ and the body-fixed acceleration in y-direction $n_{y,c}$.

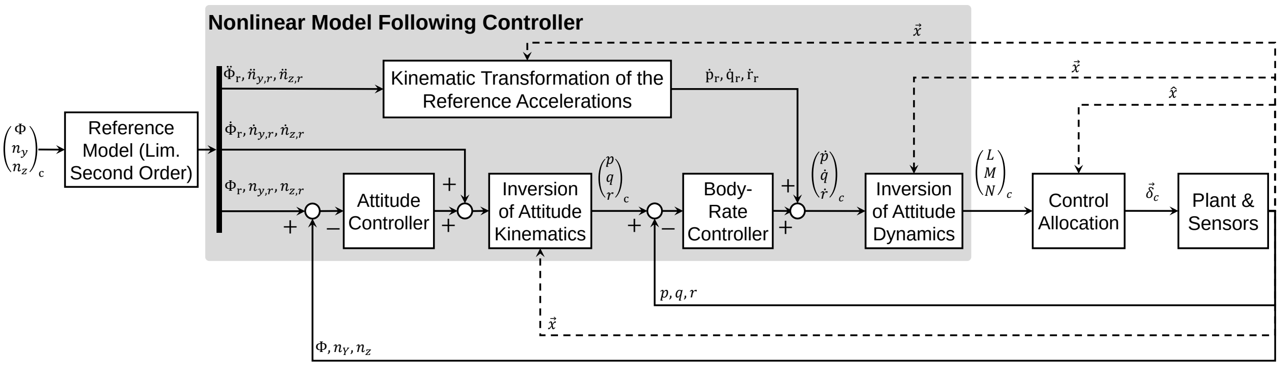
Nonlinear Model Following Controller
$\begin{pmatrix}\Phi \\ n_y \\ n_z\end{pmatrix}_c$
Reference Model (Lim. Second Order)
$\ddot{\Phi}_r, \ddot{n}_{y,r}, \ddot{n}_{z,r}$
$\dot{\Phi}_r, \dot{n}_{y,r}, \dot{n}_{z,r}$
$\Phi_r, n_{y,r}, n_{z,r}$
Kinematic Transformation of the Reference Accelerations
$\dot{p}_r, \dot{q}_r, \dot{r}_r$
$\vec{x}$
$\vec{x}$
$\hat{x}$
Attitude Controller
Inversion of Attitude Kinematics
$\begin{pmatrix}p \\ q \\ r\end{pmatrix}_c$
Body-Rate Controller
$\begin{pmatrix}\dot{p} \\ \dot{q} \\ \dot{r}\end{pmatrix}_c$
Inversion of Attitude Dynamics
$\begin{pmatrix}L \\ M \\ N\end{pmatrix}_c$
Control Allocation
$\vec{\delta}_c$
Plant & Sensors
p, q, r
$\vec{x}$
Φ, n_Y, n_z
+
−
+
+
+
−
+
+

Figure 8 – Overview of the integrated nonlinear model following control architecture [24]

The illustration indicates the interface to the control allocation module, which was presented in section 3.1. The use of a control allocation system drastically simplifies the automized flight control system (FCS) design, since no further emphasis has to be given to the control effector properties of the regarded vehicle. Applying an NMFC-based methodology allows using a model-based approach to address flight control considerations such as stability and control augmentation. At its core, the established NMFC architecture is quite similar to the architecture of nowadays often used time-scale separated NDI feedback control systems, as described, for instance, in [25]. Therefore, no further considerations concerning gain scheduling are explicitly needed.

The proposed system uses a two-loop control approach with a distinct cascaded outer-loop controller ("Attitude Controller") and an inner-loop controller ("Body-Rate Controller") to handle the slow and fast dynamics of the aircraft. In addition, a second-order reference model is used to generate reference signals. The generated reference commands are the low-pass filtered reference signal $(\Phi_r, n_{z,r}, n_{y,r})^T$ and the signals of the corresponding first and second-order time derivatives of the desired model response ($(\ddot{\Phi}_r, \ddot{n}_{y,r}, \ddot{n}_{z,r})^T$ and $(\dot{\Phi}_r, \dot{n}_{y,r}, \dot{n}_{z,r})^T$, respectively). The two higher-order signals are utilized as feedforward signals to augment the inner- and outer-loop controller signals, while the low-pass filtered signal is used for regulating the error terms of the controlled bank angle and accelerations.

The desired handling characteristics for each controlled state can be shaped by adjusting the reference model. The desired response may vary depending on the considered aircraft but must be consistent with the relevant standard and certification specifications to provide appropriate handling qualities for the pilots. The gains for the feedback controllers can be derived automatically from the COAST aircraft model by using a pole placement-based approach. To do so, COAST provides default pole locations, which can offer acceptable and robust control performance for most conventional aircraft. In cases where the default pole locations are seen as inappropriate, better suitable pole locations can be found experimentally by analyzing specific aircraft characteristics during virtual flight tests and adjusting the pole locations until acceptable behavior is achieved.

3.3 Handling of unusual or non-CPACS-supported aircraft behaviors

The CPACS file format allows the definition of a large variety of configurations in a quite generic and flexible way. However, there are some implicit assumptions in the format which limit the capability to express certain effects. For instance, no cross-coupling between the aerodynamic effectiveness of control surfaces is foreseen, so if control surfaces have significant interactions, these cannot be represented. This limitation is not really a problem for typical airliner configurations, but can be limiting for other configurations (e.g., UCAV). Some extensions of the format have been tested and used in specific projects, but they have not been integrated to the CPACS standard yet. Another limitation applies to the interaction between aerodynamics and propulsion, which cannot be represented yet. This limitation results from the fact that CPACS has originally been used for investigating aircraft configurations powered by turbofan engines (e.g., classical tube-and-wing or blended-wing-body configurations).

For such cases, data or models are exchanged aside of the main CPACS file. The usual process would be to use COAST to generate a model for all components that are described in the CPACS file and to augment it in a second step to include the other components, as shown in Figure 9. Depending on the case considered, the second step might be more or less automated and generic. Such cases may also cause additional work for the controller design, as each model augmentation may cause different problems (e.g., introducing couplings that were not present before) or change the objectives and constraints of the control design problem (e.g., having to consider differential thrust for yaw control or to couple thrust variations with pitching behavior, as described in [14]).

The 12-propeller configuration of the SynergIE project (cf. Figure 6) is an example of a configuration for which specific add-ons had to be used [14]. Such configurations do require some manual work, as COAST can hardly automatize the model generation and controller design that cannot be described in CPACS file. By handling all the CPACS-compliant parts of the model, COAST still eases the work of the flight dynamics specialists and control designers, who can focus on the less common parts of the model. Configurations requiring unusual model structures may often be the ones exhibiting rather unusual flying characteristics and for which it is worth investigating their handling qualities, controllability, and testing control laws/autopilot functions as early as possible in the MDO process.

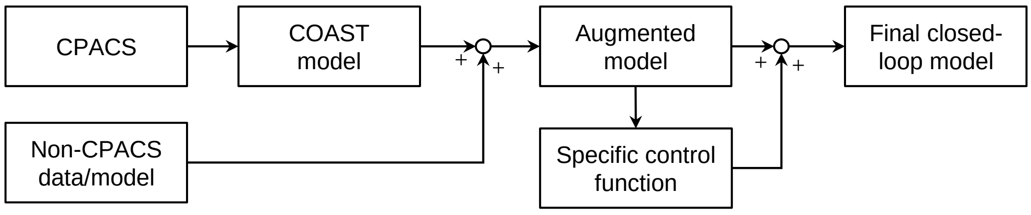

Figure 9 – Model augmentation for non-CPACS-conform configurations

4. AVES integration

The simulation center AVES (Air Vehicle Simulator) consists of a fixed-based and a motion-based simulator with exchangeable cockpits of the Airbus A320 and the Airbus Helicopters EC135. It is operated by the DLR Institute of Flight Systems in Braunschweig since 2013 [26]. An electropneumatic hexapod system reproduces the motion sensation and thereby increases the immersion. The hexapod system is shown in Figure 10a. A large dome provides a visual projection field of 240° in horizontal and 95° in vertical direction [27]. The AVES A320 cockpit shown in Figure 10b is a highly detailed replica of the original A320 cockpit, combined with an operator station for simulator control in the rear (not shown in Figure 10b). With the exception of the software for the motion platform, all AVES software was developed in-house [27], which enables a high level of flexibility and simplifies the implementation of custom systems and interfaces as well as the integration of all kinds of flight simulation models and scenarios. AVES was, for instance, used for developing and testing of airline-oriented training scenarios and techniques [28, 29], for investigating the impact of spoofing attacks on GNSS systems [30], simulating air-to-air refueling scenarios [31, 32], investigating performance-based ice detection systems [33], or developing an energy-management/noise abatement assistance system and training the flight crews how to use it [34]. For these these simulator campaigns a variety of different models were implemented and integrated into the AVES infrastructure. The integration of dynamic models generated by COAST is not much different. The main challenges are to deal with the large input/output interfaces to and from the cockpit, to map them to the model internal variables, and to implement widely automated processes to permit regular updates during the MDO design loop.

(a) Simulator cabin on the motion system (b) Cockpit view in the A320 simulator cabin

Figure 10 – The AVES motion simulator with the A320 cockpit

The A320 simulator cockpit provides typical Airbus-like cockpit hardware such as side sticks, thrust levers, pedals, flight control unit, multi-purpose control and display unit, and displays. In pre-design stages however the data required to simulate such functionalities are limited, and the cockpit functions have to be restricted to relatively generic functions. The cockpit interface with COAST is presently limited to the side sticks and the pedals, the thrust levers, the gear and flap levers, and a simplified version of the primary flight display and navigational display. All other cockpit controls are currently inoperative.
A *minimum implementation* of a CPACS-defined model via COAST would typically only include simple manual control modes, such as a direct law and a normal law similar to those used in Airbus aircraft. Thanks to model-based techniques described in section 3, it would be possible, with a reasonable amount of additional work, to adapt a complete autopilot and autothrust (similar to those of Airbus aircraft, with all managed and unmanaged modes) permitting to design and evaluate specific flight procedures. Such investigations are relevant for aircraft configurations with flight performance characteristics that deviate from the usual ones. This might, for instance, be the case for short-take-off-and-landing (STOL) aircraft, for aircraft of unusual size (e.g., very large blended wing bodies), or for aircraft approaching and departing with unusual speeds (e.g., supersonic aircraft). This would also be required for assessing the noise of a new aircraft type with regular procedures or with specific assistance functions as in [34]. Simulating an aircraft with a cockpit that was not specifically designed for it, might bring some limitations. For instance, the two thrust levers of the A320 cockpit are configured to control the left and right engines, respectively, even if the considered aircraft has more engines on each side. The mapping of engines to their respective sides (respective throttle levers) is done automatically in the initialization phase of COAST. Keeping these limitations in mind, being able to adapt fairly quickly a full flight simulator to assess an aircraft described in CPACS, including adaptations for scenarios which require operation-relevant procedures, is a pretty unique capability offered by combining COAST with the flexibility of the AVES simulation framework.

5. Conclusions

This paper presented COAST, a 6-DoF flight simulation tool for aircraft configurations described in the CPACS standard. It can be used to enhance multidisciplinary preliminary aircraft design processes by providing information about stability and controllability of the configurations early in the design process, thereby accelerating the design process and mitigating risks. The structure of the model including its interface to CPACS was outlined, and the aspects of its integration with the AVES full flight simulator was shown. The developed flight control laws were presented, which use pole placement and a control allocation algorithm to ensure a consistent control behavior independent of the individual aircraft configuration.

5.1 Future Work

As mentioned in section 2.1, CPACS does not yet provide possibilities to define landing gear or actuator characteristics in a level of detail which could be sufficient for a 6-DoF flight simulation. Developments to enhance CPACS in those aspects are currently in progress, and the authors are involved to help finding a compromise between the required flexibility of the data structure and the practicality and feasibility of the resulting implementation. Once these changes are fixed in the CPACS standard, the corresponding models and wrapper functions for the landing gear and control surface actuators will be implemented.

In section 2 it was explained that the CPACS-dependent modules of COAST are already implemented in C++ S-function. It is planned to implement the non-CPACS-related model components in C++, too, in order to obtain a closed S-function containing the full model, which will bring a performance gain for application on desktop configurations.

Regarding the flight control laws (subject of section 3.2), some of the desired autopilot functionalities are yet not fully implemented. It is planned to implement further A320-like autopilot functionalities in future development efforts. Additionally, it is planned to analyze the possibility of further automizing the current flight control design with methodologies which mitigate the problem that manual adjustment steps are still needed for some aircraft, for the case that the default closed-loop pole locations are in conflict with the flight physical properties of the regarded vehicle. This problem might be solved with the extended use of further model knowledge available in the CPACS files, which allows the a priori consideration of the physical limitation of an aircraft.

6. Contact Author Email Address

daniel.kiehn@dlr.de

7. Copyright Statement

References

[1] Alder, M., Moerland, E., Jepsen, J., and Nagel, B., “Recent Advances in Establishing a Common Language for Aircraft Design with CPACS,” in *Aerospace Europe Conference, 25–28 February 2020, Bordeaux, France*, Feb. 2020.

[2] Ciampa, P. D. and Nagel, B., “AGILE Paradigm: The Next Generation Collaborative MDO for the Development of Aeronautical Systems,” *Progress in Aerospace Sciences*, vol. 119, Nov. 2020. DOI: `10.1016/j.paerosci.2020.100643`.

[3] Boden, B. *et al.*, “Distributed Multidisciplinary Optimization and Collaborative Process Development using RCE,” in *AIAA Aviation Forum 17–21 June 2019, Dallas, Texas*, Jun. 2019. DOI: `10.2514/6.2019-2989`.

[4] Nagel, B. *et al.*, “Communication in Aircraft Design: Can we establish a Common Language?” In *28th Congress of the International Council of Aeronautical Sciences (ICAS), 23-28 September, 2012, Brisbane, Australia*, Sep. 2012.

[5] Moerland, E. *et al.*, “On the Design of a Strut-Braced Wing Configuration in a Collaborative Design Environment,” in *AIAA Aviation Forum 2017 – 17th AIAA Aviation Technology, Integration, and Operations Conference*, Jun. 2017. DOI: `10.2514/6.2017-4397`.

[6] DLR (German Aerospace Center) Institute for Software Technology, *TIVA XML interface (TiXI)*, URL: https://github.com/DLR-SC/tixi, Accessed: 03.05.2022.

[7] DLR (German Aerospace Center) Institute for Software Technology, *TIVA Geometry Library (TiGL)*, URL: https://dlr-sc.github.io/tigl/, Accessed: 03.05.2022.

[8] Kaenel, R. von *et al.*, “CEASIOM: Simulating stability & control with CFD/CSM in Aircraft Conceptual Design,” in *26th Congress of International Council of the Aeronautical Sciences (ICAS), 14–19 September 2008, Anchorage, Alaska*, Sep. 2008.

[9] Rizzi, A., Eliasson, P., McFarlane, C., Goetzendorf-Grabowski, T., and Vos, J., “Virtual Aircraft Design and Control of TransCruiser - A Canard Configuration,” in *AIAA Atmospheric Flight Mechanics Conference, 2–5 August 2010, Toronto, Ontario, Canada*, Aug. 2010. DOI: 10.2514/6.2010-8245.

[10] Rizzi, A., “Modeling and simulating aircraft stability and control – the SimSAC project,” *Progress in Aerospace Sciences*, vol. 47, no. 8, pp. 573–588, 2011, Special Issue – Modeling and Simulating Aircraft Stability and Control. DOI: 10.1016/j.paerosci.2011.08.004.

[11] Rizzi, A., Zhang, M., Nagel, B., Böhnke, D., and Saquet, P, “Towards a Unified Framework using CPACS for Geometry Management in Aircraft Design,” in *50th AIAA Aerospace Sciences Meeting, 09–12 January 2012, Nashville, Tennessee*, Jan. 2012. DOI: 10.2514/6.2012-549.

[12] Hasan, Y. J. *et al.*, “Stability and Control Investigations in Early Stages of Aircraft Design,” in *36th AIAA Applied Aerodynamics Conference 2018, June 25–29, 2018, Atlanta, Georgia*, Jun. 2018. DOI: 10.2514/6.2018-2996.

[13] Hasan, Y. J., Roeser, M. S., and Voigt, A., “Evaluation of the Controllability of a High-Altitude Platform in Atmospheric Disturbances Based on Pilot-in-the-loop Simulations,” in *70th German Aerospace Congress, 31.08.–02.09.2021, Bremen, Germany and online*, 2021.

[14] Vechtel, D. and Buch, J.-P., “Aspects of Yaw Control Design of an Aircraft with Distributed Electric Propulsion,” in *70th German Aerospace Congress, 31.08.–02.09.2021, Bremen, Germany and online*, 2021.

[15] The MathWorks, *Matlab Documentation: What Is an S-Function?* URL: https://www.mathworks.com/help/simulink/sfg/what-is-an-s-function.html, Accessed: 03.05.2022.

[16] National Oceanic and Atmospheric Administration, *U.S. Standard Atmosphere, 1976 (NASA TM-X-74335)*, 1976.

[17] US Government, Department of Defense, *World Geodetic System 1984. Its definition and relationships with local geodetic systems*, Rockville, MD, 1991.

[18] Klimmek, T., Schulze, M., Abu-Zurayk, M., Ilic, C., and Merle, A., “cpacs-MONA – an independent and in high fidelity based MDO tasks integrated process for the structural and aeroelastic design for aircraft configurations,” in *IFASD 2019 - International Forum on Aeroelasticity and Structural Dynamics*, Jun. 2019.

[19] Liersch, C. M., Schütte, A., Siggel, M., and Dornwald, J., “Design studies and multi-disciplinary assessment of agile and highly swept flying wing configurations,” *CEAS Aeronautical Journal*, vol. 11, no. 3, pp. 781–802, May 2020. DOI: 10.1007/s13272-020-00453-y.

[20] International Organization for Standardization, *Flight dynamics – Concepts, quantities and symbols – part 1: Aircraft motion relative to the air (ISO 1151-1:1988)*, Geneva, Switzerland, Apr. 1988.

[21] Nguyen, L. T., Gilbert, W. P., and Ogburn, M. E., *Control-System Techniques for Improved Departure/Spin Resistance for Fighter Aircraft (NASA Technical Paper 1689)*. Hampton, VA, USA: National Aeronautics and Space Administration, 1980.

[22] Härkegård, O., “Dynamic Control Allocation Using Constrained Quadratic Programming,” *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 6, pp. 1028–1034, May 2004. DOI: 10.2514/1.11607.

[23] Airbus S.A.S. Flight Operations Support and Services, *A318/A319/A320/A321 Flight Crew Operating Manual (FCOM)*, 1 rond point Maurice Bellonte, 31707 Blagnac Cédex, France, Sep. 2012.

[24] Autenrieb, J. and Fezans, N., “Nonlinear Model Following Control Design for a Hypersonic Waverider Configuration,” in *CEAS EuroGNC "Conference on Guidance, Navigation and Control", 3–5 May 2022, TU Berlin, Germany*, May 2022.

[25] Snell, S. A., Enns, F. D., and Garrard, W. L., “Nonlinear Inversion Flight Control for a Supermaneuverable Aircraft,” *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 976–984, May 1992. DOI: 10.2514/3.20932.

[26] Duda, H., Gerlach, T., Advani, S., and Potter, M., “Design of the DLR AVES Research Flight Simulator,” in *AIAA Modeling and Simulation Technologies (MST) Conference, Boston, MA*, May 2013. DOI: 10.2514/6.2013-4737.

[27] Gerlach, T. and Durak, U., "AVES SDK: Bridging the Gap between Simulator and Flight Systems Designer," in *AIAA Modeling and Simulation Technologies Conference, 22–26 June 2015, Dallas, Texas, USA*, Jun. 2015. DOI: 10.2514/6.2015-2947.

[28] Buch, J.-P., Niedermeier, D., and Stepniczka, I., "Managing the Unexpected – Human-in-the-Loop Simulation as Effective Tool for the Assessment of the Risk Information System in an Operationally Relevant Context," in *AIAA Modeling and Simulation Technologies Conference 5–9 June 2017, Denver, Colorado*. DOI: 10.2514/6.2017-4155.

[29] Niedermeier, D., Buch, J.-P., Mohrmann, F., and Durak, U., "Simulating the Unexpected: Challenge-Centric Simulator Scenario Design for Advanced Flight Crew Training," in *AIAA Modeling and Simulation Technologies Conference, 8–12 January 2018, Kissimmee, Florida*, Jan. 2018. DOI: 10.2514/6.2018-1397.

[30] Buch, J.-P., Geister, R., Canzian, L., Gamba, G., and Pozzobon, O., "What the Hack Happened to the Flight Deck: Analyzing the Impact of Cyber Attacks on Commercial Flight Crews," in *AIAA Science and Technology Forum, 7–11 January 2019, San Diego, California*, Jan. 2019. DOI: 10.2514/6.2019-0060.

[31] Fezans, N. and Jann, T., "Modeling and Simulation for the Automation of Aerial Refueling of Military Transport Aircraft with the Probe-and-Drogue System," in *AIAA Modeling and Simulation Technologies Conference, 5–9 June 2017, Denver, Colorado*, Jan. 2017. DOI: 10.2514/6.2017-4008.

[32] Fezans, N. and Koloschin, A., "Extending Simulink with Blocks Managing Complex Objects Collaboratively — Application to Air-to-air Refueling Simulation," in *AIAA Science and Technology Forum, January 3–7, 2022 San Diego, CA & Virtual*, Jan. 2022. DOI: 10.2514/6.2022-0808.

[33] Deiler, C. and Fezans, N., "Performance-Based Ice Detection Methodology," *Journal of Aircraft*, vol. 57, no. 2, pp. 209–223, Jan. 2020. DOI: 10.2514/1.C034828.

[34] Abdelmoula, F. and Scholz, M., "LNAS – a pilot assistance system for low-noise approaches with minimal fuel consumption," in *31st Congress of the International Council of Aeronautical Sciences (ICAS), 09–14 September, 2012, Belo Horizonte, Brazil*, 2018.