# Virtual flight testing of a controller for gust load alleviation using FMI for cosimulation

Reiko Müller[1] Markus Ritter[2]

[1]DLR, Institute of System Dynamics and Control, Oberpfaffenhofen, Germany, reiko.mueller@dlr.de
[2]DLR, Institute of Aeroelasticity, Göttingen, Germany, markus.ritter@dlr.de

## Abstract

During aircraft design and certification, one of the most vital development tasks is the calculation of loads and stresses, subsequent structural sizing and iterative mutual adaptation with respect to the aircraft's systems. In an effort to build up a so called virtual flight testing capability in the DLR-wide project *Digital-X* (2012 - 2016), a simulation of a flexible aircraft model coupled with CFD based aerodynamics and a flight control system with included Gust Load Alleviation (GLA) was developed and subjected to a certification relevant gust encounter scenario. Due to the diversity of modeling and simulation tools present in the DLR, the Functional Mockup Interface (FMI) 2.0 model interfacing standard has been successfully employed to cosimulate the control system inside the enclosing simulation framework. *Keywords: Virtual flight testing, Gust load alleviation, Flight control, FMI, Cosimulation*

## 1 Introduction

An aircraft's flight envelope expresses the admissible region of flight depending on the current state (e.g. variables like angle of attack, Mach number and altitude), with upon exceeding, the aircraft will no longer be flyable (high/low speed stalling, buffeting). In analogy to this, the loads envelope specifies the corresponding limits which the aircraft structure can handle. With the advent of electronic flight control systems, an appropriate means for regulating loads automatically was found and is used to limit the maximum design loads to increase flight safety, as well as the ones due to maneuvering or environmental disturbances like gusts. The benefits are manifold, as for example structural stress and fatigue is reduced on the airframe, passenger comfort is increased and overall aircraft performance can be improved by structural design optimization.

In the following contribution, a novel application for loads analysis is introduced, combining hitherto disconnected simulation steps and forming a high - fidelity "virtual flight testing" - capability. In detail, an aircraft is discretized as Finite Element Method (FEM) - model, with the element's elastic motion solved by methods from Computational Structural Mechanics (CSM). The forces and moments acting on the airframe due to aerodynamics are calculated from the conservation laws of mass, momentum, and energy. These have no closed form analytical solution and can only be solved by employing numerical methods from Computational Fluid Dynamics (CFD). The *Python* - based software framework *FlowSimulator* (Meinel and Einarsson, 2010) and CFD solver *TAU* (Schwamborn et al., 2006) were developed and utilized in the *Digital-X* project for multidisciplinary simulation of transport aircraft with aerodynamics calculated by CFD (Kroll et al., 2016). A *Modelica* - based flight control system with added gust load alleviation functionality had to be integrated in the *FlowSimulator* setup to conduct the virtual flight tests by means of cosimulation using the FMI 2.0 - standard (Mod, 2014). The principal layout of this approach is shown in figure 1. It benefits from the advantages of FMI, that are the time-savings due to omission of user-driven API development, interoperability for various tools and efficient simulation and event/error handling. Due to the large amount of simulations necessary for design and tuning of the controller to a specific test case, a second model based on a faster executing approximative method was established using the loads analysis software *Varloads* (Hofstee et al., 2003). The application scenario is an encounter of a frontal vertical gust, with the control objective of reducing the vertical accelerations and loads on the structure.

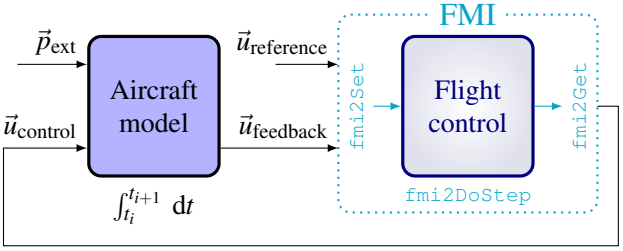


**Figure 1.** Integration loop of the controller using FMI for cosimulation to conduct virtual flight tests. $\vec{u}_{\text{reference}}$ contains controller reference values e.g. from a Flight Management System (FMS). As well, the aircraft model can depend on external parameters and inputs $\vec{p}_{\text{ext}}$ that are not part of the cosimulation loop.

The paper is structured as follows: In section 2, the governing equations of motion and the elastic deformation of the aircraft are discussed. The two aerodynamic models necessary for the controller synthesis as well as the high

fidelity simulation are derived in 2.1. The controller is laid out in section 3, including the design in *Modelica* and the gust load alleviation functionality in sections 3.1 and 3.3, along with the integration in the cosimulation setup in section 3.4. Section 4 discusses the application to the gust encounter scenario, while conclusions and an outlook for future work are given in the final section 5.

## 2 Aircraft modeling

The motion of an aircraft through the air can be described in different levels of detail. The simplest notion is of a point on which the aircraft mass is concentrated, that translates due to external forces and the weight, given by the dynamic equilibrium of forces (d'Alembert principle). When taking into account distributed masses and the rotational movement of the aircraft, one arrives at the rigid-body equations of motion, yielding six degrees of freedom. These are defined in aircraft mean body axes with respect to a ground-fixed inertial Cartesian coordinate system on a local tangent plane (flat earth assumption) with uniform gravity, also known as the Newton-Euler equations (1):

$$\begin{bmatrix} \mathbf{M}_{bb}\left(\dot{\vec{V}}_b + \vec{\omega}_b \times \vec{V}_b\right) \\ \mathbf{I}_{bb}\dot{\vec{\omega}}_b + \vec{\omega}_b \times (\mathbf{I}_{bb}\vec{\omega}_b) \end{bmatrix} = \mathbf{T}_{rb}^{\mathrm{T}} \Phi_{gr}^{\mathrm{T}} \vec{P}_g^{\mathrm{ext}} \tag{1}$$

with

| | |
|---|---|
| $\mathbf{M}_{bb}$ | Mass matrix |
| $\mathbf{I}_{bb}$ | Inertia tensor |
| $\vec{V}_b = [u\,v\,w]^{\mathrm{T}}$ | Body-fixed velocity vector |
| $\vec{\omega}_b = [p\,q\,r]^{\mathrm{T}}$ | Rotational velocity vector w.r.t. body fixed system |
| $\mathbf{T}_{rb}$ | Transformation of Center of Gravity (CG) to grid reference point |

In (Waszak and Schmidt, 1988) the equations of motion of the elastic aircraft are derived using the mean axis conditions. These are fulfilled easily by using mode shapes (eigenvectors) of an unconstrained (free-free) structural model and ensure that the rigid body equations (1), and the linear elastic equations of structural mechanics in a modally reduced form (2), are inertially decoupled.

$$\begin{aligned} &\Phi_{gf}^{\mathrm{T}}\mathbf{M}_{gg}\Phi_{gf}\ddot{\vec{u}}_f + \Phi_{gf}^{\mathrm{T}}\mathbf{B}_{gg}\Phi_{gf}\dot{\vec{u}}_f \\ &+\Phi_{gf}^{\mathrm{T}}\mathbf{K}_{gg}\Phi_{gf}\vec{u}_f = \Phi_{gf}^{\mathrm{T}}\vec{P}_g^{\mathrm{ext}} \end{aligned} \tag{2}$$

with

| | |
|---|---|
| $\Phi_{gr}$ | Modal matrix rigid body modes |
| $\Phi_{gf}$ | Modal matrix of flexible modes |
| $\vec{P}_g^{\mathrm{ext}}$ | Vector of external forces to structural grid points |
| $\mathbf{M}_{gg}$ | Physical mass matrix |
| $\mathbf{B}_{gg}$ | Damping matrix |
| $\mathbf{K}_{gg}$ | Stiffness matrix |
| $\vec{u}_f$ | Generalized coordinates of elastic modes |

Hence equations (1) and (2) are only coupled by means of the external forces $\vec{P}_g^{\mathrm{ext}}$, which in the end allows that both the large nonlinear motions of a maneuver, and the small linear perturbation introduced by the flexible structure, be taken into account.

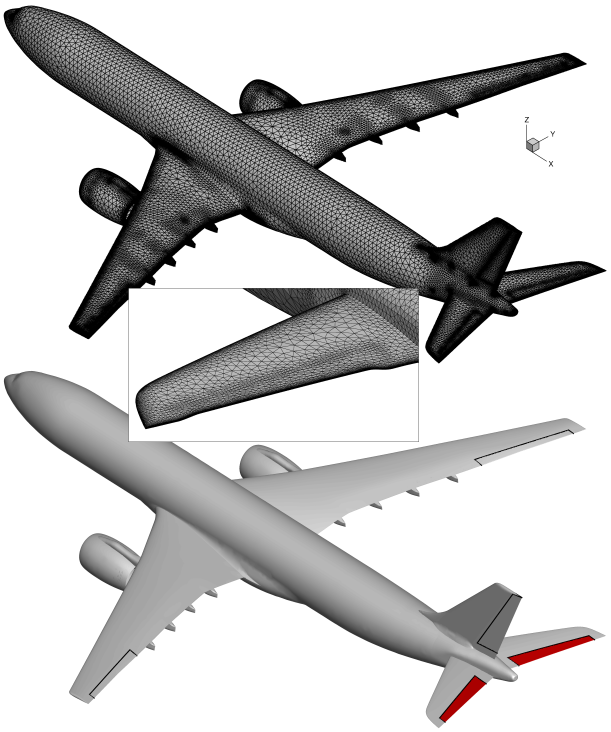

**Figure 2.** *Digital-X XRF-1* CFD computation mesh with control surfaces and exemplary deflection of the horizontal tail plane control surface of 5 degrees on top. A blending technique was used to obtain a smooth transition at the boundaries of the elevators, which are the only control surfaces used during the gust encounter cosimulation.

### 2.1 Aerodynamic models

The aerodynamic forces included in $\vec{P}_g^{\mathrm{ext}}$ are derived from the conservation laws for mass, momentum and energy. While the continuity equation depicts the mass flow through a control volume in the airflow, the Navier-Stokes equations describe the equilibrium of forces, taking into account viscosity, volume forces (e.g. due to gravity) and the momentum flow through the volume. Compressibility of the flow field requires the introduction of the energy equations, formulating the equilibrium between energy flow through the volume, energy produced due to the forces and moments, external energy contributions and inner and kinetic energy of the medium.

In combination, these form an equation system to calculate the forces / the pressure distribution on the aircraft's surface, for which however no closed form analytical solution exists. Numerical methods to solve this kind of problems are grouped under the term of Computational Fluid Dynamics (CFD), with DLR's *TAU* code being a comprehensive software environment for this task and therefore an obvious choice as CFD - solver for the *FlowSimulator*


DOI 10.3384/ecp17132921

framework (see description of the simulation setup in 3.4). Due to the nature of turbulent flow, changes can happen on a very small scale, which is why the computational grid for numerically solving the Navier-Stokes equations also may need a very fine resolution locally, to include all turbulent phenomena. As can be seen in figure 2, a higher grid density has been applied especially at geometry changes or regions of expected turbulence.

The complex grids in turn cause a large increase in computation time for calculation of the aerodynamic forces and moments, while during controller and aircraft design, quite often thousands of simulation runs are performed, e.g. to iteratively tune controller gains or to investigate aircraft response to stresses dependent on multidimensional parameter spaces. Due to their high demand on computational power, the Navier-Stokes equations are generally not viable for these kind of tasks and have to be simplified. A first step is to solve only for the unknowns that are most relevant to those applications, e.g. the pressure distribution on the object's surface.

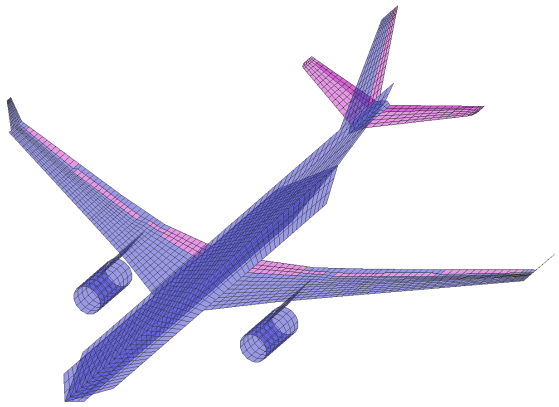

**Figure 3.** Aircraft aerodynamic model composed of lift surfaces, for use in the Vortex Lattice Method (VLM) or Doublet Lattice Method (DLM), generated by *VarLoads*. Blue panels belong to the aircraft body, purple ones to the control surfaces.

In the beginning of the 20th century, Prandtl found that for flows at higher Reynolds numbers $Re > 10^5$, the effect of viscosity is approximatively limited to a thin boundary layer encompassing the object's body. Consequently, the flow beyond the boundary layer can be considered as inviscid and importantly, the pressure gradient through it normal to the surface as zero ($\frac{\partial p}{\partial z} \approx 0$). In order to obtain the pressure distribution on the object's surface, it is therefore sufficient to calculate it in the inviscid flow just outside of the boundary layer using the inviscid Navier-Stokes or Euler equations. The assumption of isentropic (no energy contribution/drain) and irrotational flow allows to define a velocity potential function

$$\vec{v} = \text{grad}\,\Phi = \left[u, v, w\right] = \left[\frac{\partial \Phi}{\partial x}, \frac{\partial \Phi}{\partial y}, \frac{\partial \Phi}{\partial z}\right] \tag{3}$$

which is inserted into the Euler equations. These can then be linearized around $\vec{v}$, with the disturbance velocities $\left[u', v', w'\right]$

$$\vec{v} = \begin{bmatrix} u_\infty \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} u_\infty + \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \\ \frac{\partial \varphi}{\partial z} \end{bmatrix} \tag{4}$$

to arrive at the unsteady Prandtl-Glauert equation:

$$(1 - Ma^2)\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} - \frac{2U}{a^2}\frac{\partial^2 \varphi}{\partial x \partial t} - \frac{1}{a^2}\frac{\partial^2 \varphi}{\partial t^2} = 0 \tag{5}$$

When neglecting the time-dependent terms, the linear second order Laplace equation for the Vortex Lattice Method (VLM) (Hedman, 1966) is obtained:

$$(1 - Ma^2)\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = 0. \tag{6}$$

This method calculates a matrix of Aerodynamic Influence Coefficients (AIC) based on (6) to model lift distributed on an approximation of the aircraft consisting of several lift surfaces as shown in figure 3. The unsteady counterpart (in the frequency domain) for solving (5) is the Doublet Lattice Method (DLM). External aerodynamic and propulsive forces are added to the inertial forces by means of the Force Summation method to calculate resultant forces and moments on the aircraft. The loads analysis software *VarLoads*, which was jointly developed by Airbus and DLR (Hofstee et al., 2003), implements all of these modeling and simulation paradigms and was used to prepare the model with simplified aerodynamics for the controller synthesis.

# 3 Controller design and integration

The Flight Control System (FCS) or short "controller", follows the classical cascaded design layout that is well studied in both theory and practice (see e.g. (Brockhaus et al., 2011)). This layout is based upon the fact that the aircraft's equations of motion can be separated into parts that play a role on different timescales (timescale separation principle). For example, the body-fixed rotational rates $[p\ q\ r]_B$ as fast states are directly linked to the deflection of the control surfaces and resulting moments. On the other hand, states referring to orientation and even more position have a slower progression. This allows to dissect the flight control system into smaller parts, as shown in figure 4: The inner loop or Stability and Control Augmentation (SCA) - block can be designed to stabilize the aircraft and to dampen the dynamic aircraft modes (e.g. phugoid, dutch roll - modes). The autopilot in turn generates a reference orientation for the modified plant of the aircraft stabilized by the inner loop. It is designed to achieve a high tracking precision for the desired trajectory variables. The additional Gust Load Alleviation (GLA) is

arranged at the level of the SCA, since it is assumed here that the gusts cannot be sensed ahead of the aircraft (e.g. by LIDAR like in Hecker and Hahn (2007)) and necessitate fast reactions of the controller / the control surfaces.

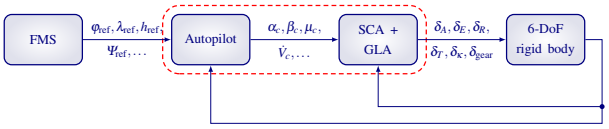


**Figure 4.** Structure of an electronic flight control system, consisting of the FMS and the FCS with autopilot and inner loop SCA. The framed blocks are the considered parts in this work.

As the scenario only considers a frontal vertical gust encounter, the GLA operates on the longitudinal dynamics, and can symmetrically deflect ailerons and elevators to attenuate the gust. Discrimination between inner and outer ailerons and elevators as well as distributed spoilers are incorporated in the GLA - layout, but only uniform and symmetric deflections of likewise $\delta_A$ and $\delta_E$ act as inputs. No actuator dynamics are modeled, due to their absence in the aircraft model of the cosimulation. Acceleration measurements are available at the Center of Gravity (CG) and form the single feedback variable:

$$n_{z,m} = \frac{\text{Lift}}{\text{Weight}} = \frac{V_K \dot{\gamma}}{g \cdot \cos(\Phi)} + \cos(\gamma) \tag{7}$$

with load factor $n_{z,m}$, kinematic velocity $V_K$, trajectory pitch angle $\gamma$ and roll angle $\Phi$. The load factor is fed into the parameterized GLA filter structure which generates control surface deflection commands, for the elevator $\delta_E^g$ with a filter structure containing e.g. a tunable time-constant. These are then super-positioned to the commands of the flight controller:

$$\delta_i = \delta_i^c + \delta_i^g, \quad i \in [A, E]. \tag{8}$$

Hence the autopilot and inner loop also contribute to the load alleviation due to the gust, by acting to hold altitude and speed.

## 3.1 Controller model in *Modelica*

The resulting Flight Control System (FCS) was implemented in *Modelica* using *Dymola* 2016, and is shown in figure 5. The *Modelica Standard* - and *LinearSystems* - libraries provide all of the needed models, with which a flight control library had been established. It consists of modules arranged in longitudinal, lateral, inner and outer loop controllers and is also prepared for use in conjunction with DLR's *FlightDynamics* library (Looye, 2008). The *Dymola* simulation tool makes use of the object oriented features of *Modelica*, allowing easy testing and interchange of different modules and furthermore offers an implementation of the FMI standard.

In the diagram view depicted in figure 5, the middle left and the lower center grey rectangular blocks represent the FCS and the GLA respectively. The FCS consists of four channels for the four individual control effectors of the airplane (throttle, elevator, aileron and rudder). The autopilot modes are set to speed -, altitude - and course - hold, while the commanded sideslip angle $\beta_c$ is zero. The inner loop receives orientation commands from the autopilot and calculates corresponding rates and control surface deflections. Each of the channels includes a set of several cascaded linear controllers. To ensure robustness over the flight envelope, multiple gust - and load - cases, a robust controller synthesis process would normally be appropriate. However, since only one gust encounter case is considered in this study, a simple tuning of the controller gains has been performed to minimize the effect on the wing root bending moment (see section 3.3).

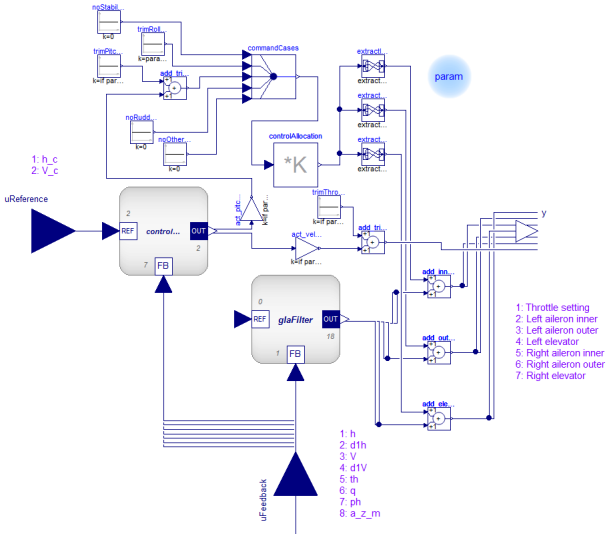


**Figure 5.** *Modelica* model of the longitudinal controller with gust load alleviation.

## 3.2 Initialization of the FMU

Each of the four channel's inner loops mentioned in the last section contain either integrator or derivative blocks with internal states that have to be initialized correctly to avoid transient oscillations in the beginning of the simulation. Furthermore the cosimulation must be compatible with both aircraft models and their respective trim algorithms. The given variables of the initialization process are the reference inputs $\vec{u}_{\text{reference}}$ and feedback inputs $\vec{u}_{\text{feedback}}$ from the aircraft, while the unknowns are the FMU outputs $\vec{u}_{\text{control}}$ (see figure 1). Hence a two-step initialization of the closed-loop simulation setup is performed:

- At first, the aircraft is trimmed separately for steady horizontal flight at a given speed and altitude (see table 1 for a set of characteristic state values), without the controller. This yields values for e.g. $\alpha$ and $\Theta$ and also for elevator deflection $\delta_E$ and throttle $\delta_T$[1].

[1]The initial values of the lateral effectors $\delta_A$ and $\delta_R$ are zero.


DOI 10.3384/ecp17132921

- In the second step, the Functional Mockup Unit (FMU) outputs need to be set to the aircraft control input values ($\vec{u}_{\text{control}}$) obtained in step 1. Yet due to FMI design, which prevents variables with output causality from the assignment of any value, additional trim parameters have to be defined.

**Table 1.** Trim values for aircraft model used for controller synthesis

| *Property* | *Unit* | *Value* |
|---|---|---|
| Mach number | - | 0.83 |
| Altitude | ft | 35000 |
| Reference velocity | m/s | 246.1 |
| Aircraft mass | kg | 198540 |
| Angle of attack | ° | 4.55 |
| Gust gradient $H$ | m | 85.9 |
| Gust velocity in $z$ - direction | m/s | −4.296 |

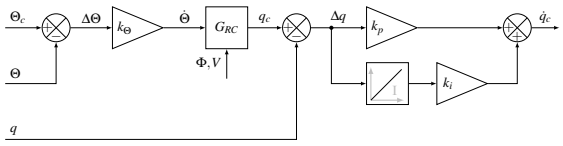


**Figure 6.** Inner loop pitch channel, with pitch angle Θ, pitch rate $q$ and $G_{RC}$ as transfer function containing the correction for turning flight (increase in pitch due to rotated lift vector).

This second step is further explained using the example of the inner loop pitch channel shown in figure 6: With given $\delta_{E,\text{trim}}$ and $\Theta_{\text{trim}}$, the single degree of freedom is the initial state value of the integrator. The trim pitch angle is added to the autopilot command

$$\Theta_c = \Delta\Theta_{\text{AP}} + \Theta_{\text{trim}}, \tag{9}$$

where $\Delta\Theta_{\text{AP}}$ is zero here due to initial $h = h_c$. Likewise the elevator command consists of

$$\delta_E = \dot{q}_c + \delta_{E,\text{trim}} + \delta_{E,\text{GLA}}. \tag{10}$$

With the constraint of steady state integrator initialization ($\dot{x}_{\text{int}} = 0$), and similar provisions for the velocity channel, the initial equation of the controller model has to be specified as shown in listing 1. By calling the `initialize()` - method of the FMU, the controller can then match the preceding aircraft trim.

**Listing 1.** Initial equation of the controller model

```
initial equation
controllerLongitudinal.y[1] = trim_de_T;
controllerLongitudinal.y[2] = trim_de_E;
```

## 3.3 Synthesis of the GLA controller

The controller and GLA gains were adapted to the gust encounter scenario using the fast executing simplified model of section 2. The single objective of this process was the minimization of the bending moment around the aircraft's longitudinal $x$ - axis (see figure 2) at the wing root station, $M_x$. In contrast to the high-fidelity simulation, both the elevators and the ailerons were actuated by the GLA. Figure 7 compares three gust encounters, one open loop, one with flight controller only, and one combined with additional GLA. The undisturbed case is added for reference and shows the bending moment at the trim condition.

The control objective is to reduce the initial maximum amplitudes of $M_x$. This is satisfied by the FCS and the GLA as expected, with the most notable difference in the second peak at $t \approx 0.7$ s. Due to several filter time-constants, the GLA does not act against the first falling peak, which is why the controller - and GLA - variants reduce the moment about the same amount. At the second rising gust peak, the GLA is able to reduce the moment around 45 %, however the GLA inputs generate an increased preceding moment. Using this highest GLA - peak, the reduction over the open loop case is still as large as 39 % with the steady trim moment value as baseline.

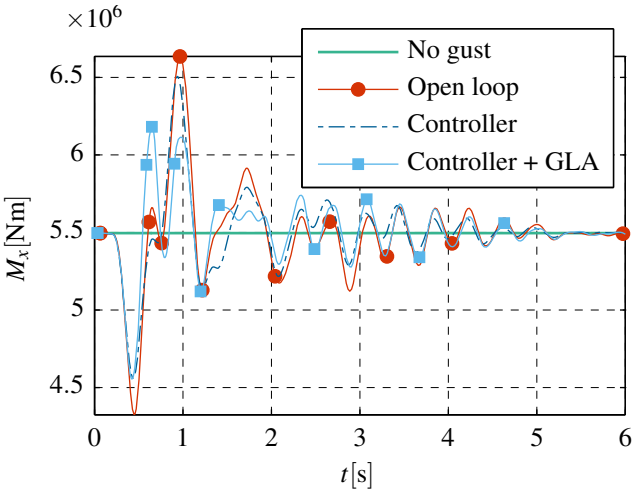


**Figure 7.** Wing-root bending moment for uncontrolled and two controlled gust encounter simulations.

## 3.4 Integration in simulation environment

The *FlowSimulator* framework allows to specify, integrate and simulate all sub-models necessary for the controlled coupled CSM - CFD application. A special *FlowSimulator* - plugin called *FSDynafly* has been developed at DLR's Institute of Aeroelasticity to model the process chain for the controlled cosimulation, see figure 8.

After initialization, the governing equations of motion of the free-flying elastic aircraft (equations (1) and (2)) are solved by *FSDynafly* for the current time step. Their outputs and the command references are passed on to the

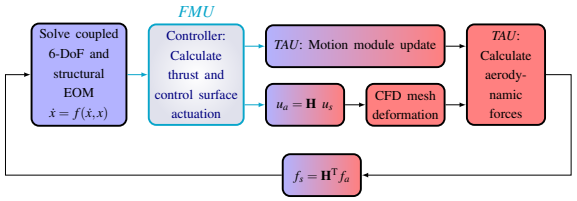


**Figure 8.** Time domain solution process of *FSDynafly_6DOF* including the flight controller.

FMU to form the control error. The deflection commands calculated by the FMU are mapped onto the structural grid at the respective control surface positions, yielding modal deformations $u_s$. An unstructured mesh has been built for the *Digital-X XRF-1* configuration using the meshing software *CENTAUR*, with the control surfaces being cut into the CAD geometry based on locations provided by Airbus. Each control surface thus has a separate boundary marker in order to be deflected properly in the unsteady gust encounter simulation. As the structural grid does not coincide with the aerodynamic one, the deformations are multiplied with the splining matrix $\mathbf{H}$, which is built from Radial Basis Functions (RBFs). It is then morphed according to the deformations $u_a$ using the submodule *FS-Deformation*, as is shown in figure 2 with the example of the Horizontal Tail Plane (HTP) - deflection. In parallel, another submodule of the CFD solver *TAU* calculates an update of the aircraft motion, followed by the actual call of *TAU* to solve for the new aerodynamic forces $f_a$ of the next time step. To solve the equations of motion, these are transformed back into forces $f_s$ relating to the structural grid by multiplication with $\mathbf{H}^T$.

The controller interfaces to *FSDynafly* through the Functional Mockup Interface (FMI), in the working principle shown in figure 1. The FMI for cosimulation methodology was adopted, since deployment as model exchange - type FMU would have been far more complicated (e.g for integration and event handling). The complete controller model shown in figure 5 is exported together with the Sundials *CVode* ODE - solver compiled in a FMI - compliant library (64-bit .so for UNIX - type target simulation environment). As the application and interfacing layer of *FSDynafly* is written in *Python*, the DLR - developed *Python* FMI - API of *PySimulator* (Pfeiffer et al., 2012) is used to address the FMU. Finally, a fixed-time step master algorithm enables communication between the two cosimulated models, a *Python* code representation is given in listing 2.

**Listing 2.** Master algorithm for the cosimulation of aircraft with the controller in *Python* (only the most relevant commands are displayed).

```
# Load the FMU
fcs = FMUInterface.FMUInterface(
        "Controller_GLA.fmu")
fcs.fmiInstantiate()
# Trim the aircraft
[x_tr, u_tr, dx_tr] = aircraft.trim(
                                x0, u0, dx0,
                                ix, iu, idx0)
# Set the trim parameters in the FMU ...
# ... to achieve u equal to u_tr
fmu_setReal_inValueAndReference(
    fcs, pars_trim,
    ["trim_de_T","trim_de_E",
     "trim_de_A","trim_de_R"], u_tr)
# Initialize the FMU
fcs.initialize()
# Integration loop
while aircraftODE.successful()
    and status == 0
    and t <= stopTime:
    # Set u to u_tr for the first ...
    # ... timestep
    if t == t0:
        u_in = u_tr
    else:
        u_in = u
    # Evaluate aircraft right hand side ...
    # ... and retrieve feedback
    der_x, out = aircraftODE.f(
    t, aircraftODE.y, u_in)
    # Set controller reference inputs
    fmu_setReal_inValueAndReference(
        fcs, u_ref, ["h", "V"], u_ref.val)
    # Set controller feedback inputs
    fmu_setReal_inValueAndReference(
        fcs, u_feedback,
        u_feedback.varNames, out)
    # Integrate one step for FMU
    status = fcs.do_step(t, dt, True)
    # Retrieve controller commands
    u = fmu_getReal_fromValueAndReference(
        fcs, y_out, y_out.varNames)
    # Set aircraft model inputs
    aircraftODE.set_f_params(u)
    # Integrate one step for aircraft
    aircraftODE.integrate(t + dt)
    # Increment the master time
    t = t + dt
# End of integration
```

# 4 Vertical gust encounter simulation

As mentioned before, the only scenario covered in this contribution is an encounter of a discrete vertical gust with the assumption that all points in planes normal to the aircraft's velocity are affected (as defined in (Joint Aviation Authorities, 1994)). The vertical velocity profile is shaped according to equation (11)

$$w_{\text{wind}} = \frac{U_{ds}}{2}\left[1 - \cos\left(\frac{V}{H}\pi \cdot t\right)\right], \qquad (11)$$

and therefore denoted as "One-minus-cosine" - gust. The parameters of this function are the design gust velocity $U_{ds}$, the gust gradient $H$, which is the distance parallel to the aircraft's flight path for the gust to reach its peak velocity, and $V \cdot t$ as the distance traveled into the gust.


DOI 10.3384/ecp17132921

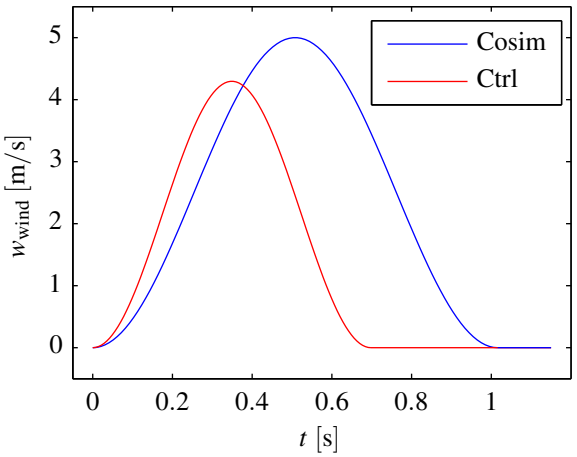


**Figure 9.** One minus cosine gust definitions used for the complete cosimulation and for the controller synthesis.

The gust parameters are slightly differing between the high fidelity cosimulation and controller simulation as shown in figure 9, similarly the angle of attack and HTP trim values, see table 2. Furthermore, in the high fidelity simulation results presented in the following, only the control surfaces of the horizontal tail plane were used as primary control surfaces to reduce the loads acting on the airframe. This was partly due to project time constraints and availability of other control surface geometries like spoilers and ailerons. The gains in overall load reduction can therefore not be compared between the high- and lower fidelity models as of now, yet this was not the goal of this specific application anyway.

**Table 2.** Trim values for high fidelity simulation, only values differing from those in table 1 are listed.

| *Property* | *Unit* | *Value* |
|---|---|---|
| Angle of attack | ° | 3.39 |
| HTP trim angle | ° | 2.58 |
| Gust gradient | m | 125 |
| Gust velocity in $z$ - direction | m/s | −5 |
| Communication time stepsize | s | 0.01 |

Two gust encounter simulations are presented in the following, one without gust attenuation, and another one with the flight controller in the loop. The results are shown in figure 10 in terms of selected states measured in the body fixed coordinate system with the pitch rate $q$, its time derivative $\frac{\mathrm{d}q}{\mathrm{d}t}$, the velocity in the $z$ - direction $w$ and the acceleration in the $z$ - direction $\frac{\mathrm{d}w}{\mathrm{d}t}$.

As can be seen from the time function of the states plotted, the actuation of the controller markedly reduces the accelerations of the airframe's center of gravity, thereby reducing structural loads as well. A reduction of the heave accelerations of more than 20% is achieved. This simulation is a purely symmetric maneuver, meaning that no

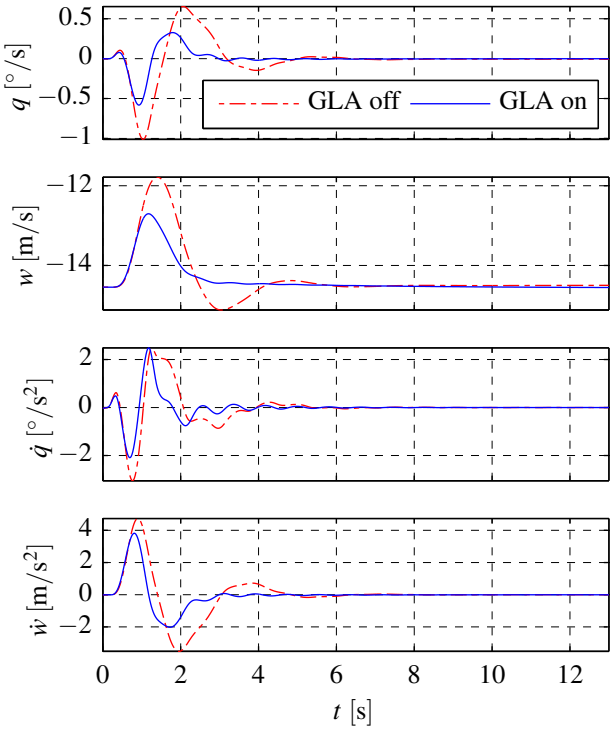


**Figure 10.** Selected states of the aircraft as function of time, showing a reduction of accelerations due to the gust load alleviation.

distinctive lateral motions are excited during the gust encounter. Small but negligible lateral motions occur due to non-negative values for $I_{xy}$ , and $I_{yz}$ of the tensor of inertia of the aircraft. These entries can be attributed to the fact that the mass model is not purely symmetric. The output of the controller in terms of the time dependent rotation of the horizontal tail plane's control surface is shown in figure 11. The maximum deflection of the HTP is about 1.3°. This value is comparatively low, but the gust disturbance velocity is small as well.

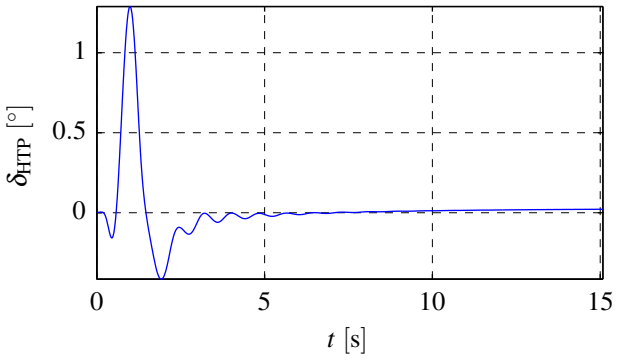


**Figure 11.** Controller output in terms of the rotation of the HTP control surface.

# 5 Conclusion and outlook

In this contribution, a new methodology for loads analysis and virtual flight testing of flight controllers is presented. Usually disconnected simulation steps are combined into a single process chain, including elastic structural aircraft modeling, full Navier-Stokes aerodynamics and a flight control system with gust load alleviation.

A flight controller with outer and inner loop, as well as gust load alleviation system was set up in *Modelica* using *Dymola*. It was tuned for a gust encounter scenario using a Vortex Lattice Method (VLM) based aerodynamic model, allowing the required large number of simulations during controller synthesis. A final reduction of up to 45% in the wing root bending moment was found there.

A setup for the cosimulation was developed in *Python*, including a fixed-step master algorithm connecting the simulation framework *FSDynafly* with the controller. By employing the FMI standard to interface the controller, dedicated API development for the dissimilar aircraft models could be omitted. Furthermore the functionalities of FMI for cosimulation allowed an easy setup and efficient operation of the controlled high fidelity simulation. Reductions in the vertical and the pitch accelerations of up to 20 % were achieved, also consequentially leading to a reduction in the structural loads.

After successfully completing this first proof of concept, future work will be directed towards functionality in larger simulation studies with multi-parameter or even multi-model test cases and different scenarios (e.g. maneuver loads, flight performance analysis). Ensuring the robustness of the controller for the entire flight envelope will be an important prerequisite for these applications, and could be achieved by employing methods from robust control design (e.g. $H_\infty$ or robust LPV control).

An immediate next step will be the addition of new control surfaces (ailerons and possibly spoilers) to the CFD mesh, since currently the only means of controlling the aircraft and the loads is the horizontal tail plane. Based on the results of the simplified aerodynamics simulation, it is expected that loads on the main wing can be further reduced by this approach. As well, it should be worthwhile to incorporate additional design criteria in the controller synthesis process. By treating it as a multi-objective optimization problem, the investigation of trade offs between e.g. load reduction, passenger comfort and flying qualities is made possible.

# 6 Acknowledgments

This work was prepared during the course of DLR project *Digital-X*. The authors would like to express their thanks to the following colleagues for their support and input: Martin Leitner, Hans-Dieter Joos, Andreas Pfeiffer and Thiemo Kier.

# References

Rudolf Brockhaus, Wolfgang Alles, and Robert Luckner. *Flugregelung*. Springer, 2011. ISBN 9783642014437. URL `http://books.google.de/books?id=2IKXH3skXBwC`.

Simon Hecker and Klaus-Uwe Hahn. Advanced gust load alleviation system for large flexible aircraft. In *Proceeding 1st CEAS Konferenz*, 2007.

Sven G Hedman. Vortex lattice method for calculation of quasi steady state loadings on thin elastic wings in subsonic flow. Technical report, DTIC Document, 1966.

Jeroen Hofstee, Thiemo Kier, Chiara Cerulli, and Gertjan Looye. A variable, fully flexible dynamic response tool for special investigations (VarLoads). In *2003 CEAS/AIAA/NVvL International Forum on Aeroelasticity and Structural Dynamics*, 2003.

Joint Aviation Authorities. Joint aviation requirements. JAR-25. Large aeroplanes. *Civil Aviation Authority Printing & Publication Services, Greville House*, 37, 1994.

Norbert Kroll, Mohammad Abu-Zurayk, Diliana Dimitrov, T Franz, Tanja Führer, Thomas Gerhold, Stefan Görtz, Ralf Heinrich, Caslav Ilic, Jonas Jepsen, et al. DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods. *CEAS Aeronautical Journal*, 7(1):3–27, 2016.

Gertjan Looye. The new DLR flight dynamics library. In *Proceedings of the 6th International Modelica Conference*, volume 1, pages 193–202, 2008.

Michael Meinel and Gunnar O Einarsson. The FlowSimulator framework for massively parallel CFD applications. *PARA 2010*, 2010.

*Functional Mock-up Interface for Model Exchange and Co-Simulation, Version 2.0*. Modelica Association, July 2014.

Andreas Pfeiffer, Matthias Hellerer, Stefan Hartweg, Martin Otter, and Matthias Reiner. PySimulator-A Simulation and Analysis Environment in Python with Plugin Infrastructure. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, pages 523–536. Linköping University Electronic Press, 2012. 76.

Dieter Schwamborn, Thomas Gerhold, and Ralf Heinrich. The DLR TAU-Code: recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.

Martin R Waszak and David K Schmidt. Flight dynamics of aeroelastic vehicles. *Journal of Aircraft*, 25(6):563–571, 1988.