CEAS-GNC-2024-080

EuroGNC Navigation Guidance Control CEAS Bristol, UK June 11th-13th 2024
University of BRISTOL ROYAL AERONAUTICAL SOCIETY AIAA

Flight Testing Advanced Control Functions on a Passenger Aircraft

Christian Weiser Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Christian.Weiser@dlr.de

Daniel Milz Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Daniel.Milz@dlr.de

Marc May Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Marc.May@dlr.de

Ramesh Konatala Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Ramesh.Konatala@dlr.de

Stefan Langen Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Stefan.Langen@dlr.de

Reiko Müller Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Reiko.Mueller@dlr.de

Christina Schreppel Research Associate, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Christina.Schreppel@dlr.de

Gertjan Looye Head of Department, German Aerospace Center (DLR), Institute of System Dynamics and Control, 82234, Weßling, Germany. Gertjan.Looye@dlr.de

ABSTRACT

Early testing of new flight control law architectures and functions or new control methods is important, as it gives relevant clues about their practicability in many respects. These include proper functioning on target hardware under relevant environmental conditions, validation of the translation of requirements into method-specific design objectives, and the incentive to go through all relevant verification and validation steps from a systems and software engineering perspective. This paper describes our processes, ways of working, methods, and tools for flight testing our new control functions and methods on a CS-25 class passenger aircraft. Particular challenges are posed by the fact that we test multiple highly different and dissimilar functions in each flight test and campaign. Therefore, we focus on how we organize parallel development, integrate into and implement experimental control software, and cost-efficiently test these functions in the limited flight time available.

Keywords: Flight Testing; Fly-by-wire; Validation & Verification; Flight Control

1 Introduction

Research on flight control algorithms at the German Aerospace Center (DLR) Institute of System Dynamics and Control (DLR-SR) focuses on control functions for new (types of) flight vehicles and new control functions, as well as on new methods to design and verify these. New types of flight vehicles include eVTOLs (electric Vertical Take-Off and Landing) configurations [1, 2], for which in addition to essential functions for automatic and manual control, such aircraft may require new features, such as e.g. transition from and to vertical flight. In addition, conventional aircraft configurations, like highly efficient transport aircraft with high-aspect-ratio wings [3] and e.g. High-Altitude-Long-Endurance aircraft [4] are considered. For both of these configuration types, load alleviation and interaction of structural loads and flight mechanics, or even flutter suppression (in the case of transport aircraft) need to be considered. Eventually, seeing such functions certified through an industrial design process in the frame of an aircraft program requires a multi-faceted view from their early conception. This view includes, apart from control engineering, systems, and software engineering, avionics, as well as all relevant aspects of flight physics. Our approach to turning concepts and theories into (candidate) solutions for future aircraft programs is to address these facets appropriately, at least to an extent proportional to our available resources.

From a control engineering perspective, new emerging methods may either be enablers for new functionalities, simplify the control architecture (and software), and/or make the overall design process more efficient. A typical example is the nonlinear control method incremental non-linear dynamic inversion (INDI) [5, 6], which allows addressing nonlinear effects and couplings from the start of the design. This saves manual scheduling of gains and, therefore, design time. By using direct measurements or estimates of angular accelerations, internal model computations can be eliminated, providing an architectural advantage compared to preceding NDI methods.

From a flight physical perspective, we have built a solid capability to develop loop-capable flight dynamic models that allow for analyses typically covered by other engineering domains. These models include airframe flexibility, flight loads, and relevant onboard system dynamics [3]. The underlying modeling process is integrative, which means that model data or computational methods as used in other disciplines are incorporated, rather than being developed from scratch. This ensures compatibility with acceptable computational means of compliance in those disciplines. An example of loads analysis with an INDI control law can be found in [7].

Flight testing of new functions and methods is an indispensable step in increasing their technology readiness level (TRL). The first and most basic reason, of course, is to demonstrate that a new control algorithm can be translated into software on a target flight control computer (FCC), runs correctly in real-time, can interact with its surrounding hardware and software as intended. Furthermore, it is verified that the compiled flight control software (FCSW) runs as predicted in simulations, and, implicitly, can cope with uncertainties that come with moving from modeling towards the intended operational environment. Referring to the INDI example again, synchronizing estimated or measured control surface deflections and angular accelerations turned out to be the key to realize INDI for the very first time on a un-manned areal vehicle (UAV) in 2013 (this principle was later formally proven [8]). A second reason for early flight testing is the validation aspect by itself. Each new function starts from newly derived requirements, and each new control methodology has its own way of addressing requirements (e.g. weighting functions, placement of poles, formulation of optimization criteria). Requirements can be extensively verified in computational analyses. Flight testing is the most reliable way, though, to validate new requirements as well as their translation into control method-specific criteria. Very common examples are flight performance and handling qualities [9, 10]. A third reason is that due to its cost and risks, flight testing arguably provides a strong incentive to address a design application from a consistent systems and software engineering perspective. This comes with traceability of requirements, sticking to agreed-on processes for design and testing, and developing and executing verification and validation (V&V) and test plans, part of which may be control method dependent. Although, in our case, it is usually not intended for

certification, the process may certainly give relevant clues in this direction and address the aforementioned facet of systems and software engineering relevant to the new function or method at hand. A final reason is the human aspect. Especially in the case of new control methods, part of the development process is in the aforementioned aspect of translating design requirements into method-dependent objectives. This sometimes involves a steep learning curve. The experience gained towards flight tests is invaluable for a design engineer in this respect and results in highly valuable feedback to control method development and application.

A number of examples of flight testing of new functions and methods have been documented in [11–20]. Notable are [21] and [22], describing first flights of INDI on a UAV and a CS-25 aircraft.

1.1 The experimental aircraft

Since 2017, the platform we regularly use is the Cessna Citation 550 PH-LAB (Fig. 1), which we charter for a fixed number of flight hours and days of ground preparation time during each campaign. The aircraft is jointly operated by TU Delft and the Dutch Aerospace Center (NLR) and serves as a multi-functional research platform. It is certified according to specifications for large airplanes (CS 25) and equipped with a conventional, fully reversible flight control system providing a fixed-geared link between the pilot's controls and the control surfaces of the aircraft. Additionally, an autopilot, which has authority over the primary flight controls (elevator, aileron, rudder), is available. For flight control testing campaigns, the aircraft is equipped with an experimental fly-by-wire (FBW) system [23], which was developed by the TU Delft and uses the autopilot servos as control actuators. This setup has been thoroughly tested and certified under CS 25 [24]. In addition, a flight test instrumentation system (FTIS) [25] is available for data acquisition and logging. Sensor data available for both, data logging and use in the flight control laws, includes the aircraft's original sensors (Attitude and Heading Reference System (AHRS), airdata computer, Global Positioning System (GPS)), but also further sensors (angle of attack (AoA), angle of sideslip (AoS), angular acceleration, etc.), which are only part of the experimental setup. The hardware setup is described in more detail in [22, 23, 25]. We perform our campaigns closely cooperating with TU-Delft staff, who oversee all operational, piloting, technical, and safety-related aspects.

For the development and testing of novel control methods, a baseline aircraft model is needed, which is the basis for the later comparison of the controller performance. For the control of rigid, six degrees of freedom (DoF) aircraft, the Cessna Citation is a suitable platform since there exists a detailed, high fidelity aerodynamic model, as well as detailed models of many sub-components (e.g., fly-by-wire system, weight and balance, engine dynamics, ...). Furthermore, the possibility to flight test on this same aircraft type is unique, and therefore, this aircraft type is chosen as the baseline for primary flight control law development.

Fig. 1 Cessna Citation II 'PH-LAB' by Alan Wilson, licensed under CC BY-SA 2.0.

1.2 Scope of the Paper

The paper structure follows a classical V-Model which is sketched in Fig. 2, where the steps are linked to the sections in the paper. Sec. 2, our control law development process together with the controller architecture is presented. This allows us to implement various types of functions in a single framework such that all of these functions may be tested sequentially during a single flight, without requiring a software reload. Additionally, examples for functions tested during the latest flight campaign are briefly

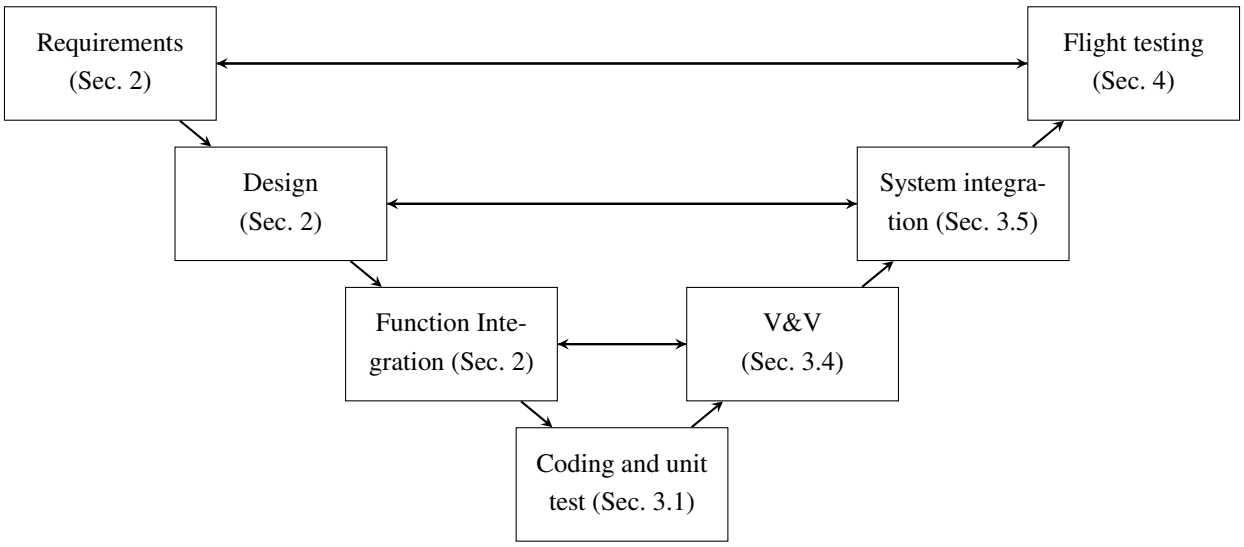


Fig. 2 V-model of the flight control design and validation process.

discussed. As the main focus of the paper is the overall process, not single functions, for details the reader is referred to references provided. In Sec. 3, the applied testing, implementation and computation process for flight test clearance of functions is described. We furthermore present pre-flight verification, including preparation of the test program on a simulator, as well as ground tests with the aircraft and control software. It further focuses on hardware and software implementation and related verification steps (H/W testing, Code checking). Finally, Sec. 4 presents example data of the latest flight test campaign and relevant lessons learned.

2 Development and Integration of Flight Control Functions

As the availability of the aircraft and resources are limited, the aim is to test multiple, or all, newly developed functions in each test flight.

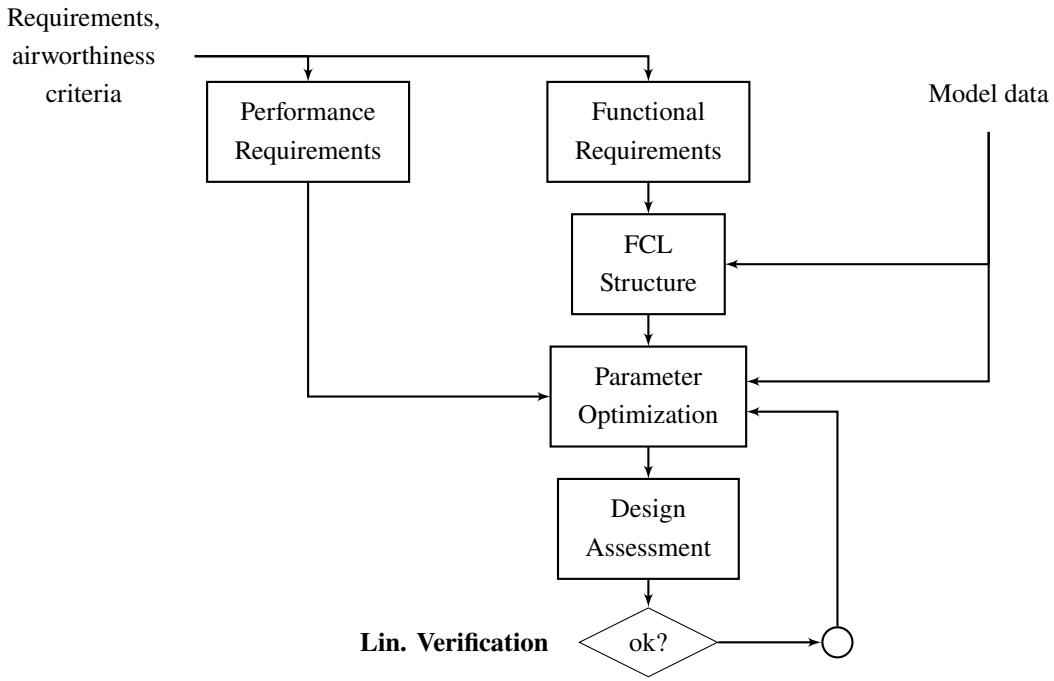


Fig. 3 Flight control law development process.

As a complete reloading of software during flight is not preferable, we integrate all functions in a single control law architecture. Although all controller features are developed self-contained and independently, these are combined in one SIMULINK® system (via references) and auto-coded in precisely this setup. Fig. 3 depicts the layout of the requirement-based control law design which is sketched in this section, starting from functional requirements and the architecture design in Sec. 2.1, control law development and tuning / optimization in Sec. 2.2 and giving some examples from the latest flight test campaign in Sec. 2.3.

2.1 Generic control law architecture

A first step in the process towards a flight control architecture is the definition of functional requirements, e.g. which functions shall the flight control system provide. In our example, we show the basic needs of any flight controller which are listed in Table 1: control and stabilization of the aircraft's attitude and flight path variables. From these requirements, the control structure as depicted in Fig. 4 is derived.

	Requirement	**Scope**	**Components**
1.	Attitude Tracking	Track e.g. aircraft attitudes / angular rates (Φ, Θ, $\dot{\Phi}$, $\dot{\Theta}$)	Inner Loop
2.	Flight path tracking	Track flight path (V, γ, χ)	Outer Loop
3.	Signal Processing	Signal conditioning for use in feedback loops: state estimation, (complementary) filtering.	

Table 1 Functional control design requirements.

Even though functions and their interfaces may change from campaign to campaign, the chosen flight control system (FCS) architecture is generic, see Fig. 4. It has multiple loops that include inner loop attitude control, automatic flight, Flight Management System (FMS), as well as signal processing. Individual functions to be tested are integrated in their appropriate hierarchical loop.

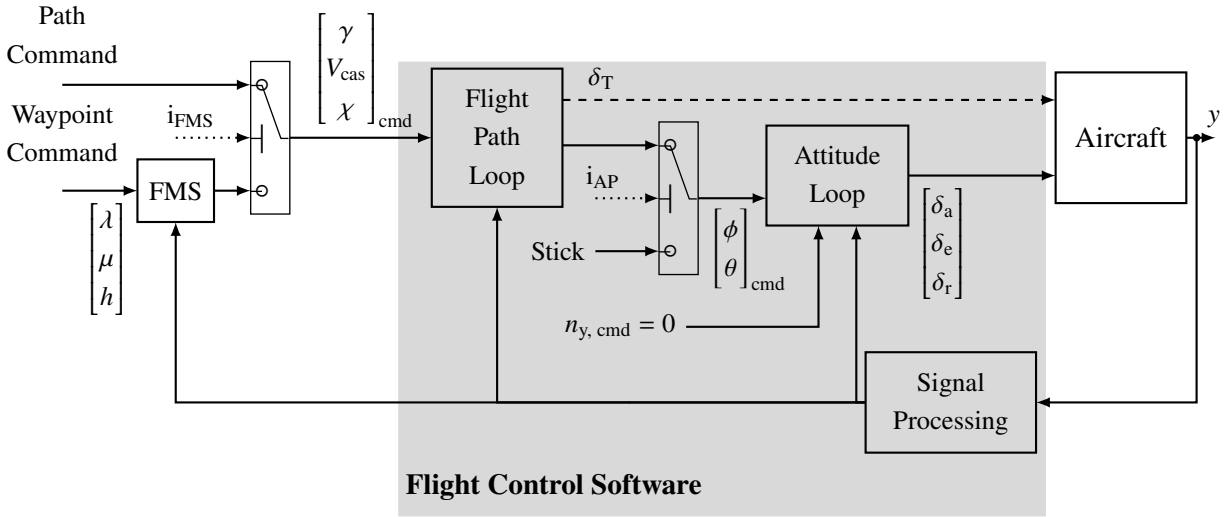


Fig. 4 Architecture of the research FCS with inner-loop attitude control (roll angle ϕ and pitch angle θ), outer-loop flight path control (flight path angle γ, calculated airspeed V_{cas}, course angle χ, and throttle/thrust command δ_T), and FMS tracking waypoints (latitude λ, longitude μ, and altitude h).

The interfaces depicted are those that were used during the latest campaign, but are constantly updated for upcoming campaigns. From right to left, the block *Aircraft* represents the flight and system

dynamics of the Cessna Citation research aircraft, *Inner Loop* is the container for a rate command attitude hold (RCAH) attitude control law, where it is possible to choose between implemented controller types, which are:

- INDI [15, 22],
- linear parameter varying (LPV) [14]
- hybrid incremental nonlinear dynamic inversion (hINDI) [17]
- incremental approximate dynamic programming (iADP) [19]

In this paper, hINDI [17] is discussed as a novel control architecture and is used to demonstrate the workflow. Further to the left, the *Outer Loop* in Fig. 4 is implemented as Total Energy Control System (TECS) autopilot [26], which controls the flight path angle (FPA) and airspeed. On the lateral axis, a conventional course/heading controller is implemented. Similar to the inner loop, it would be possible to have several autopilot variants on board and change the variant during flight with one click. The FMS, which is the outermost control loop, can guide the aircraft along initially defined trajectories. The aircraft does not have an auto-throttle servo, for this reason, we have a standard function that translates auto throttle control laws thrust commands into N1 references for both engines, which then need to be tracked by the safety pilot. The architecture shown in in Fig. 4 depicts the functions tested in the August 2023 campaign. Four different control functions have been developed: the nonlinear dynamic inversion (NDI) (see Sec. 2.3.1) and iADP (see Sec. 2.3.2) are standalone controllers and do not interact with each other. The FMS (see Sec. 2.3.4) uses the TECS (see Sec. 2.3.3) controller for flight path guidance, and therefore they are closely coupled. Furthermore, TECS uses either INDI or LPV as an inner loop . All controllers with their related functions are placed in one single model, thus development and especially testing is strictly connected.

2.2 Control law function development

For the development of individual functions, a process as described in [27] is used. Starting from the architecture in Fig. 4 and function requirements (see e.g. Table 1), this development includes a detailed design of the function architecture, translation of design requirements into method-relevant criteria, parameter optimization, and detailed verification of performance and robustness by means of model-based analyses. The process is sketched in Fig. 3 and in this section, the performance requirements are discussed. An example for performance requirements used for development is listed in Table 2:

ID	Requirement	Scope	Components	Value
X.1	Performance	Specified as e.g. rise time	Inner Loop, Outer Loop	e.g. 3 s for a 30 deg Φ step
X.2	Stability	Gain & Phase Margins (GM, PM)	Inner Loop, Outer Loop	GM < 6 dB PM < 45 deg
X.3	Control bandwidth	Available bandwidth of the actuation system for the respective loop	Inner Loop, Outer loop	XX rad/s
X.4	Smooth enabling	Bump-less transfer during enabling for each loop	Inner Loop, Outer loop	-
X.5	Anti- wind-up	Protection of all integrators against wind-up	Inner Loop, Outer loop	-

Table 2 Typical control design requirements.

In our example, the essential requirements for performance and stability are addressed, which are also used in the validation later on in Sec. 3. To ensure that all developers can collaborate efficiently

without obstructing each other's work, a framework has been set up. This framework consists of version control with Git and a generalized interface between the controller and the plant. A decision was made to split the project into several sub-projects (realized as *git submodules*), including the flight dynamics model (FDM), controller, and FMS. Furthermore, there is one overall wrapping project, which brings all sub-projects together and contains the closed-loop simulation model, which corresponds to Fig. 4. All testing, e.g. unit testing, software-in-the-loop (SIL), Monte-Carlo simulation is done with this closed loop model containing all components. However, only the controller and FMS projects are used for code generation.

One crucial thing when developing in a team on a monolithic controller is to define the interfaces. However, during the development, there is often a need for slight changes to allow new functionalities. Thus, specifying an interface control document (ICD) separately and ensuring that everyone complies with it would drastically increase development efforts. We went with a more straightforward but comparably safe option: all interfaces between the different subsystems are realized via a bus (in SIMULINK®) or struct (in **C++**). The definition of the bus is a MATLAB® code included in a separate *Common* sub-project to allow a standalone definition of the interfaces and shared data where all modules can access the latest bus definitions and test if the signals used are currently available on the bus. Those are not embedded in the main project to allow standalone controller and FMS code generation but are imported into all projects. By using submodules, a handy feature is included to keep track of interface changes and to ensure compatibility of all subsystems: every sub-project is linked to one commit (one point in development history) of the interface definition. If two sub-projects with different interfaces were used together, there would be a clash in the referenced commit.

2.3 Example control functions

Our experience is that this architecture and way of working allows for integrating diverse functions. As an example, for the latest campaign in August 2023, the following controllers and subsystems were designed and integrated:

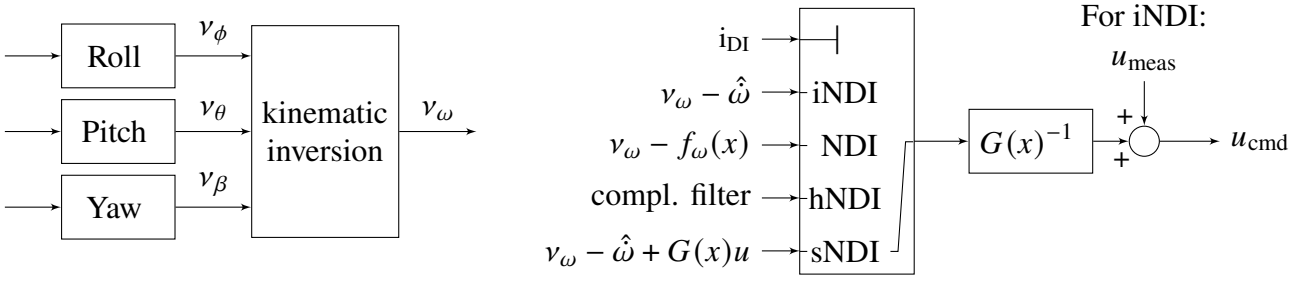
2.3.1 Dynamic Inversion-based Control Laws

Non-linear dynamic inversion (NDI) is an established method to control non-linear systems, especially aircraft attitude and rates. However, to decouple the different axes of the system, in-depth system knowledge is required. This includes an in-depth understanding and quantitative approximation of the aerodynamic effects during flight. New developments in the last decades strive to decrease this dependency, either by adaptive elements [28], robust control methods [29], approximations [5], or sensor measurements [30]. Both latter approaches led to a new branch of NDI: incremental NDI. A detailed description and analysis of this method can be found in, e.g., [5, 6, 31]. New developments include a hybrid approach of NDI and sensor-based NDI, as shown in [17].

The overall control system originated from the first in-flight trials of incremental NDI [22]. From there on, the controller evolved into the current holistic state. Several features were added over time, including Pseudo Control Hedging (PCH). During flight testing, there is an urgent need to switch between different functionalities and en-/disable certain functionalities since flight time is valuable and features may be erroneous. This way, the dynamic inversion controller grew to an integrated block with easy and bump-less switching.

NDI-based attitude control systems are, in general, split into the *inversion core*, the *linear compensator*, and the *reference model*, as shown in Fig. 5. For a detailed explanation of the subsystems, see e.g. [22, 32].

Note the specialty in this implementation: NDI, as well as incremental, sensory, and hybrid NDI, can be continuously switched during execution.



(a) Outer-loop dynamic inversion controller with uncoupled channels.

(b) Dynamic inversion core with switch for the different controllers. hNDI does a complementary filtering of the NDI and sNDI signal.

Fig. 5 Architecture of the dynamic inversion control law.

2.3.2 Incremental Approximate Dynamic Programming

From the perspective of a resilient fault-tolerant flight control system, an online learning-based adaptive control law, which could adapt to system failures and handle non-linearities, is interesting. A Reinforcement Learning(RL) based iADP flight control law [18] has the potential to address this issue, where a local, linearized incremental model is identified to estimate and minimize an infinite horizon quadratic cost-to-go, exclusively using the collected aircraft state data. The non-linear discrete system is locally linearized using Taylor series expansion, as shown in equation (1). The identification process employs a Recursive Least Squares (RLS) approach, providing a Linear Time Variant (LTV) approximation to the original model.

$$\Delta X_{k+1} \approx T_{k-1}\Delta X_k + G_{k-1}\Delta u_k \tag{1}$$

The dynamic programming technique is then utilized to approximate an infinite horizon quadratic cost-to-go function, incorporating observed one-step costs in tracking error and control input, along with the identified incremental model as shown in equation (2).

$$X_k^T P X_k = (y_k - y_k^r)^T Q (y_k - y_k^r) + (u_{k-1} + \Delta u_k)^T R (u_{k-1} + \Delta u_k) + \gamma (X_k + \Delta X_{k+1})^T P (X_k + \Delta X_{k+1}) \tag{2}$$

By Bellman's optimality principle, an optimal control law or policy can be obtained by minimizing this cost-to-go concerning the incremental control input(Δu_k), as shown in equation (3)

$$\Delta u_k = -(R + \gamma G_{k-1}^T P G_{k-1})^{-1} [R u_{k-1} + \gamma G_{k-1}^T P X_k + \gamma G_{k-1} P T_{k-1} \Delta X_k] \tag{3}$$

A control design requirement of achieving an angular rate tracking task with the capability of online adaptation is defined, followed by the development of the controller architecture as shown in Fig. 6. Decoupled longitudinal and lateral rate tracking control loops are designed for the preliminary Verification and Validation(V&V) of the Flight Control Law. This architecture has the potential to address the failures, as any failure to the aircraft that changes the rotational dynamics of the aircraft could be identified online, and the controller should still be able to recover the aircraft from the off-nominal condition, as the controller is agnostic to any specific model.

As there are no standard V&V procedures defined for self-learning adaptive flight controllers, an overlap in the methods and tools used for other controllers being tested are first identified and are then adapted along with some custom V&V methods ([20]). The development of this controller, alongside other control designers working on the same aircraft, expedited the design process for the iADP controller.

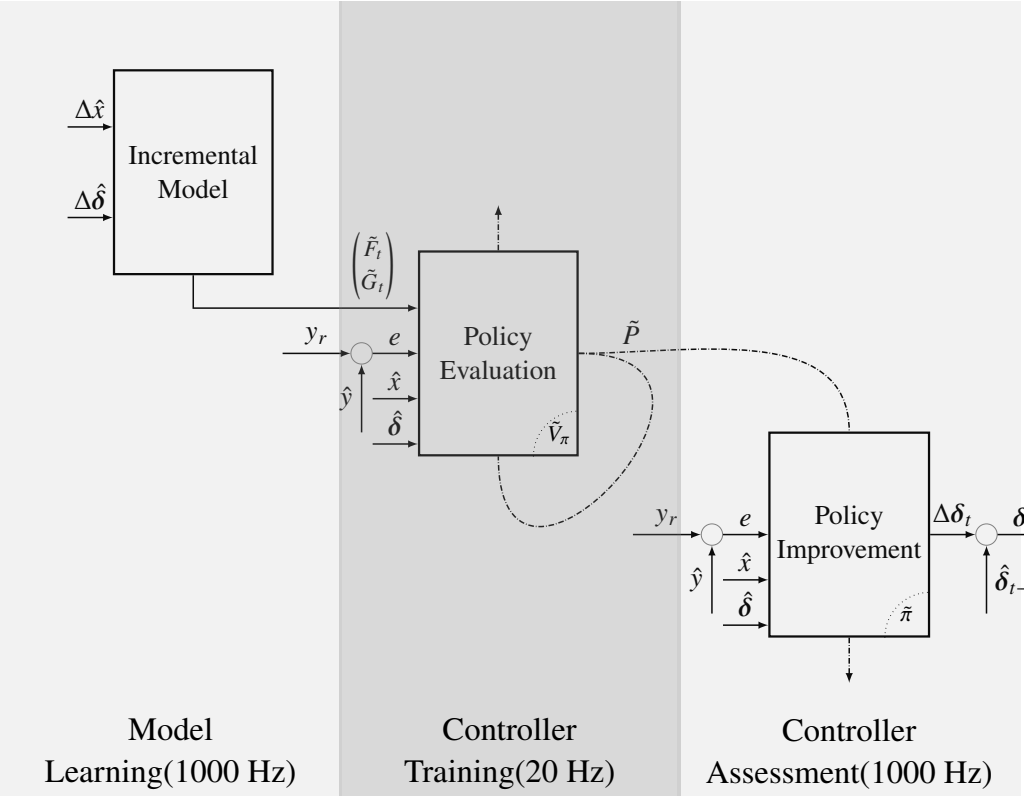


Fig. 6 Structure of the Reinforcement Learning Agent of iADP Flight Control Law. Model Learning provides the latest model estimates using the RLS algorithm. Controller Training evaluates(V_π) the Control Policy using incremental model estimates and one-step Cost. Controller Assessment takes actions and improves Control Policy (π) based on policy evaluation. The frequency at which each subsystem on the Flight Control Computer runs is indicated below. Refer to Nomenclature for the corresponding notation of the symbols used.

This collaborative effort led to effective management of time and effort, ensuring efficient utilization of resources, which finally resulted in a successful maiden flight testing of this controller [19].

2.3.3 TECS Autopilot

DLR is currently developing a solar-powered high altitude long endurance (HALE), which is having its maiden flight in Summer 2025. DLR-SR is responsible for developing and implementing auto flight control laws, allowing it to automatically track course, altitude/flight path, and speed commands. For the longitudinal autopilot concept, the well-known TECS architecture has been chosen, as it naturally and very efficiently provides decoupled tracking of airspeed and FPA commands. Furthermore, its architecture inherently allows prioritizing speed control via the pitch command loop in case of thrust saturation. It has been applied in previous HALE design projects at Boeing (Condor) and DLR [33].

Although TECS has been flight tested before [11, 27, 34, 35] and the Citation is quite a different type of aircraft, we decided to test the current status of development in order to validate functionality and the design process to achieve it, and to gain implementation and test experience in order to reduce risk before implementing the control laws on the intended target HALE platform for its first flight. The TECS controller architecture as implemented for the campaign is depicted in Fig. 7. The proportional and integral gains are tuned via an optimization process as described in [26, 36]. The optimization targets were a bandwidth of 25% of the inner loop bandwidth, together with satisfactory gain and phase margins as well as disturbance rejection specifications.

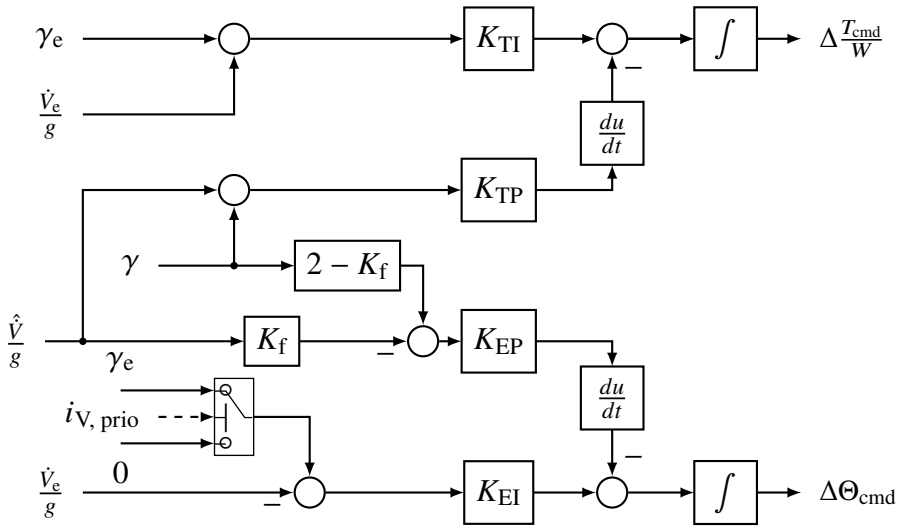
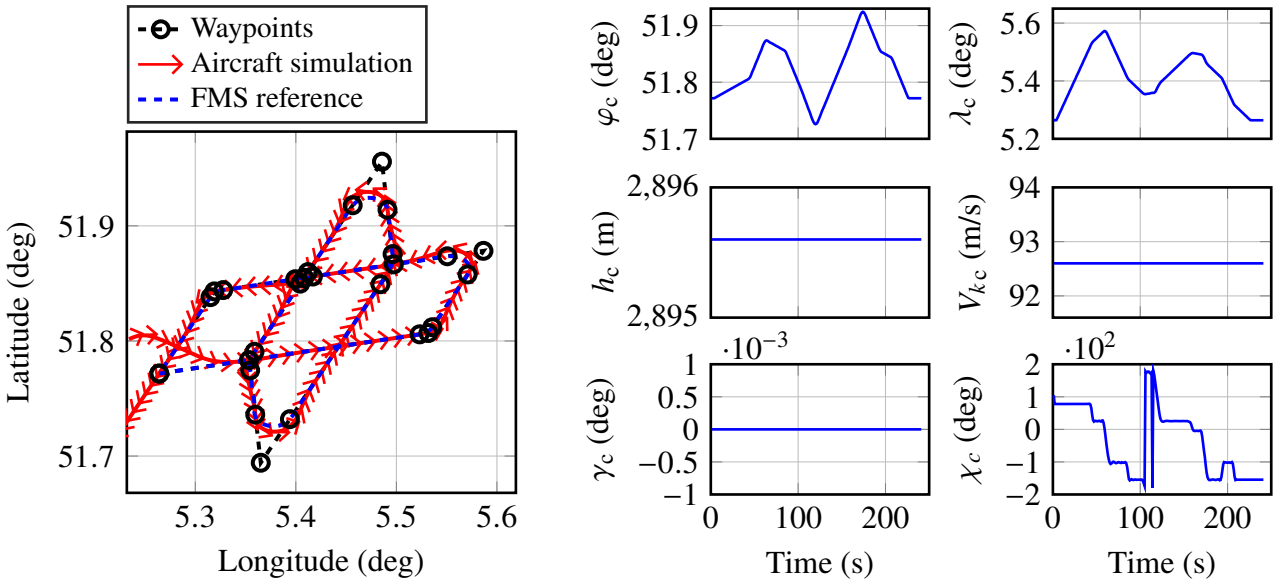


Fig. 7 TECS controller architecture.

2.3.4 Flight management system and trajectory guidance

If activated, instead of the pilot command, a Flight Management System (FMS) has the role of generating smooth reference inputs for the following Outer-loop/autopilot, as shown in Figure 4. These incorporate altitude, velocity, course angle, pitch angle, and derivatives for a reference trajectory in space (3-D) or space and time (4-D). The method of generating these trajectories for use in direct optimal control problems was described in [37, 38] and has been slightly adapted to fit within the process of code generation, deployment on the actual hardware and flight testing. In the following, the inner workings of the FMS module will be outlined, along with details on the implementation during the campaign. The most intuitive parameterization approach uses waypoints, which form a sequence to depict an aircraft trajectory from the start to the endpoint. Each of those is assumed to have a position (latitude φ_{wp}, longitude λ_{wp} and altitude over the ellipsoid h_{wp}) along with a time annotation, either given by an inertial velocity value V_{wp} or time instant t_{wp}. The preparation of a waypoint sequence can happen in different ways - for instance, using a map and collecting the list of coordinates from documents or databases. Another convenient approach is to employ web-based planning tools like SkyVector®[1] and SimBrief®[2] to plan the missions and generate input files for the FMS module, which was adopted in this case. An input eXtensible Markup Language (XML) file is then passed over to the actual FMS module, which is implemented as a MATLAB® system object. This allows stating a time-dependent dynamical system using MATLAB® code that can be easily integrated into the overall Simulink® project (see Sec. 3.3). The waypoint sequence or polygon given in the XML is then enriched with auxiliary waypoints to model arc transitions at heading changes, for instance, a flyover or a flyby of the target waypoint when switching to the next flight leg, while also taking into account kinematic constraints (for example a roll angle limit). An interpolation of this polygon in projected coordinates (using the gnomonic projection) with either B-Splines or a combination of linear legs and circular arcs provides minimum distance routes. In the final step, the result is degree-elevated (using a nonlinear least-squares B-Spline approximation), obtaining higher-order splines for position, velocity, time, and associated derivatives (for example, rate of climb/descent $\dot{h}$, acceleration $\dot{V}_k$, course angle rate $\dot{\chi}$, etc.). In the actual simulation or on the test system, the FMS module can provide the 4-D reference values at the current (Dueca-) time step by just evaluating the B-Splines and their analytical derivatives (for the 3-D case, a root-finding algorithm (e.g., `fzero` function) is necessary, which usually converges very quickly/within sampling time limits). An example simulation result is shown in Figure 8, where the aircraft captures a predefined trajectory polygon constructed from several flyby waypoints.

[1] `www.skyvector.com`
[2] `www.simbrief.com`



(a) FMS trajectory using flyby waypoints. **(b) FMS reference signals.**

Fig. 8 Example simulation with aircraft and FCS.

3 Flight Control Law Implementation Process

The presented monolithic, single model structure in Sec. 2 has the advantage that all necessary settings for the export and testing of the controller can be supplied on the overall model level. This section deals with the implementation, testing (SIL & Hardware-in-the-Loop (HIL)), and code generation of the flight control laws. It is shown in Fig. 9 that the requirements defined in Sec. 2 are the basis for the FCS structure and controller design. This section starts off after the controller tuning and first stability checks with linear models. First, the aspects of unit testing and SIL simulation are addressed. This verifies the requirements and checks if all parts of the control algorithm have been tested. After the simulation-based unit tests have passed, in Sec. 3.2 further (offline) testing can be done in order to explore worst cases or even instabilities in case of model uncertainties / parameter variations. This can be achieved via (MOPS) anti-optimization or Monte-Carlo analysis. Finally, the steps of emitting C Code and testing it on the hardware platform are presented, followed by hardware integration and HIL testing.

3.1 (Unit) Testing

The final stage within the controller design is class-based unit testing, which shall evaluate each functionality. Therefore, each controller function shall provide tests of the implemented functionalities. Each unit test calls multiple simulations with different inputs, and for each case, tracking, stability and robustness requirements of the closed loop system are evaluated. This is shown through linking the requirements from Table 2 to the tests in Table 3. The notation in the last column "X." relates to the fact, that these properties are usually linked to multiple functions, and thus the requirements (and unit tests) exist for each controller function that inherits these performance properties. Furthermore, the unit tests shall also cover all possible logical conditions (if the feature contains multiple signal paths or logical switches). In this way, the (model) coverage of the tests is analyzed using the SIMULINK® coverage software, and especially the decision and execution coverage should be close to 100% as a sign that the implemented unit tests cover all states of the feature under test. If desired, similar coverage tests using the same in- / output data could be implemented in the C++ code to assess code coverage in the compiled controller. Table 3 explains the different unit test cases which are considered for clearance testing. Those test cases can be collected in a test suite, which has many advantages, e.g., when generating clearance reports or using continuous integration (CI).

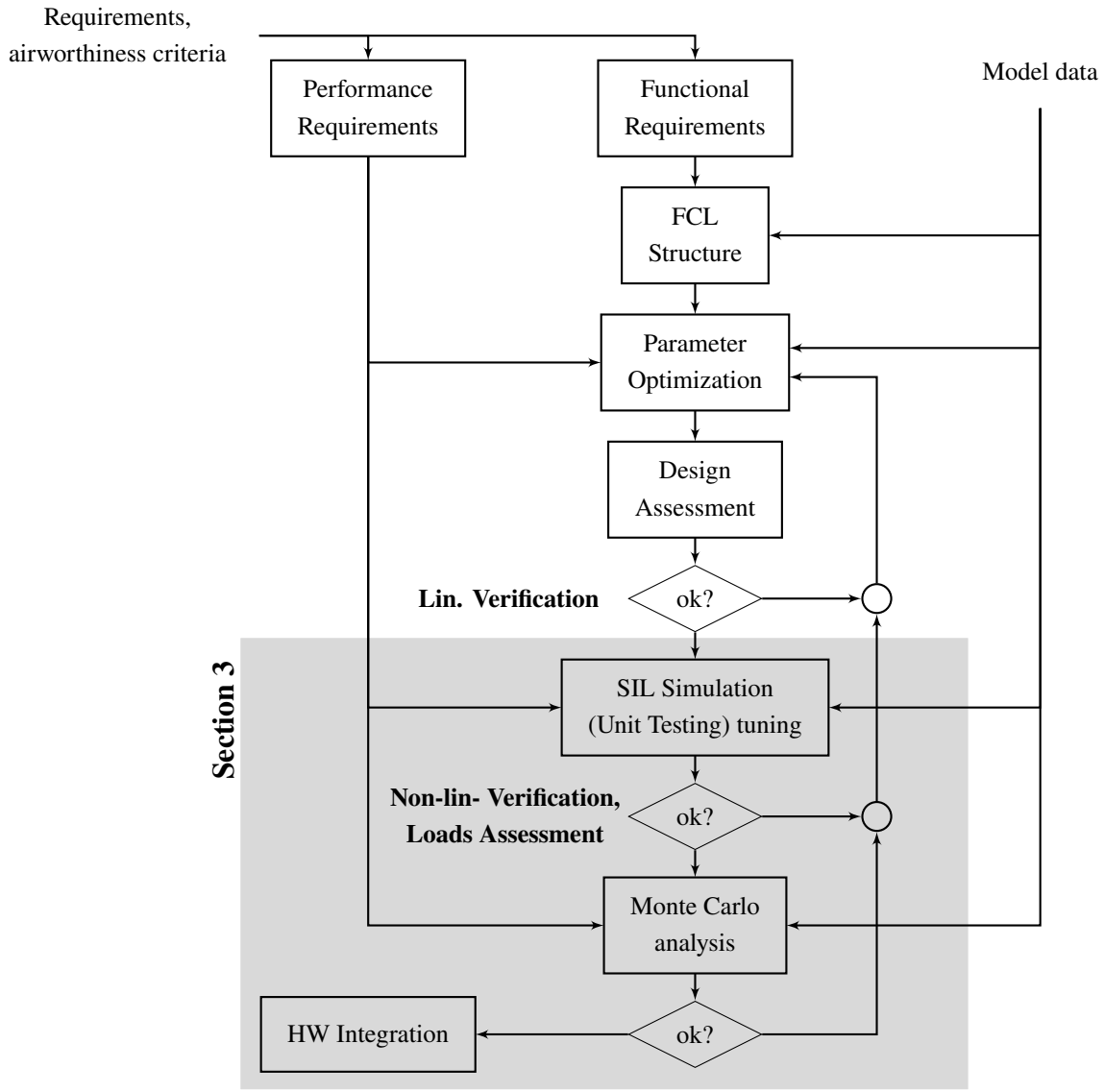


Fig. 9 Control design and clearance process.

Test	Scope	Link to RQs
Enabling	Check for bump-less enabling of all controllers	X.4
Step Response	Check tracking performance in all possible mode combinations	X.1, X.2, X.5
(Gust) Disturbance	all possible mode combinations, check disturbance rejection & Stability margins, disturbance rejection	X.2, X.3

Table 3 Unit test types used for controller clearance.

After the simulation-based unit tests defined in Table 3 have succeeded, in Sec. 3.2 further testing can be done in order to explore worst cases or even instabilities in case of model uncertainties / parameter variations.

3.2 Optimization-based Clearance of Control Laws

Before implementation on the target platform and testing in flight, flight control laws need to be extensively verified with respect to their functional and performance requirements (see Sec. 3),

especially in presence of uncertainties. Although the Citation FBW system is certified (within its operational constraints) for use with experimental control algorithms, thorough verification saves time and unnecessary cost. Performing this task efficiently is especially important in a research context, since resources are limited compared to industrial-scale flight tests.

Apart from control method-specific ones, like gain and phase margins, μ-analysis etc., most analyses are simulation-based. To this end, the aircraft model has been parameterized, allowing for variation of aerodynamic, propulsion, mass, and systems related parameters. Relevant design requirements are verified by means of selected simulation scenarios. This may vary from basic step inputs to imposing relevant disturbances, failure scenarios, etc.

Based on the parameterization and selected analysis scenarios, various types of clearance analyses may be performed. These may include simulating all extreme parameter cases (if their relevant number is not too high), optimization-based worst-case search [39–41], as well as Monte-Carlo analyses.

The DLR software Multi-Objective Parameter Synthesis (MOPS) is used for this purpose, featuring functionality for multi-case parameter optimization, worst-case search, design of experiments, and Monte Carlo analysis. These tasks can be applied within the software on a common parameterized model or run-script, thus significantly reducing coding effort and making extensive analysis (-chains) possible. Written in MATLAB®, it also seamlessly integrates into the MATLAB® project, testing, and reporting tool-chain and can be employed during different stages of the development process straightforwardly.

3.3 Real-Time Capable Implementation and Code Verification

Once the SIMULINK® controller model has been thoroughly tested in the simulation environment and has successfully passed the offline unit tests, it is suitable for further processing. This first step until now is the left hand side ("model in the loop") step in Fig. 10.

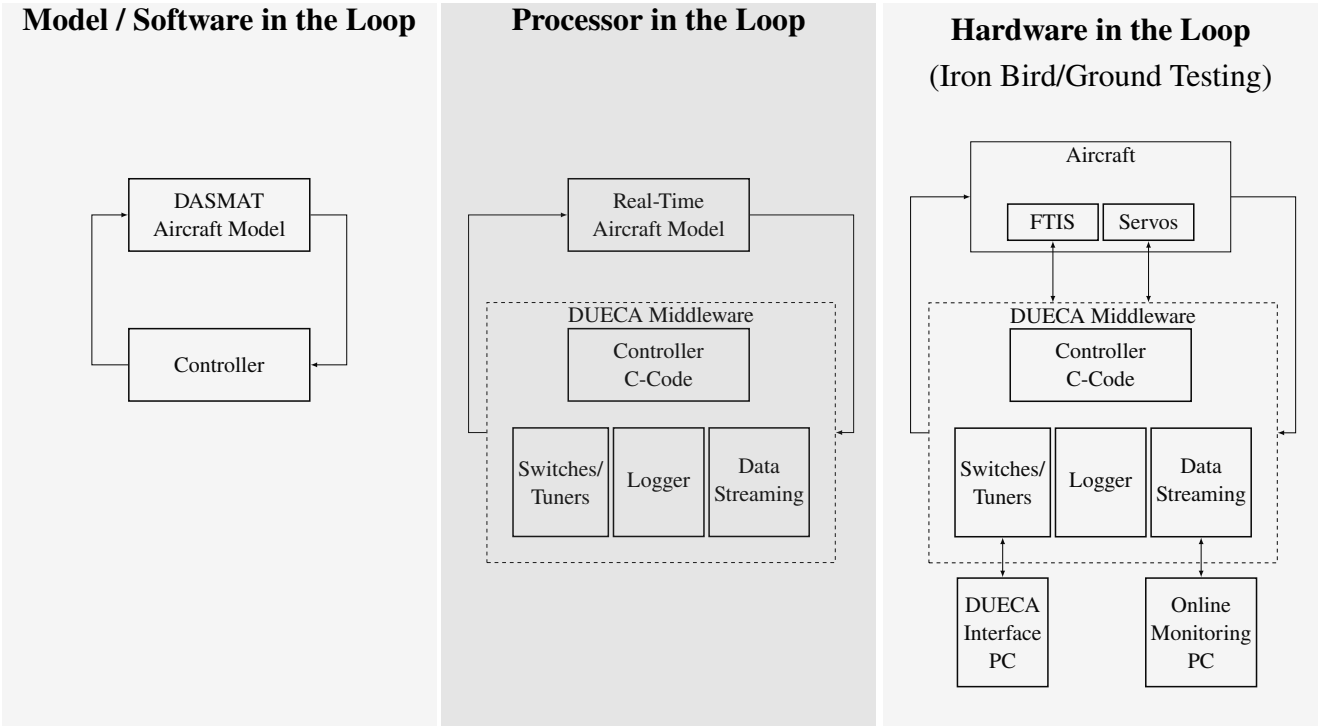


Fig. 10 Simulation setup indicating various fidelity levels of the simulation setup from Model in the Loop towards Hardware in the Loop.

For the deployment onto the hardware and the integration into the aircraft software framework, it needs to be available as C code. This is automatically generated out of the model by using the

Embedded Coder[3]. It exports code that is optimized for use on embedded systems. The SIMULINK® controller model must meet specific requirements so that real-time capable code for the execution on the embedded target can be derived from it. The complete model must be discrete and no longer include any continuous components. A fixed step size needs to be specified so that the execution on the target platform is performed periodically within a constant time interval. With the generated code (or an Functional mock-up unit (FMU) compiled from the code) the second step, SIL can be performed as well as processor-in-the-loop (PIL). The difference between these scenarios is, that for the first one, the compiled code / FMU can run on the same machine as the aircraft model. For the PIL setup, the code is already deployed inside the Delft University Environment for Communication and Activation (DUECA) framework and placed on the flight computer or a similar machine, while the flight model runs as a separate DUECA node. The final step in Fig. 10, HIL then connects the actual aircraft sensors with the flight control computer and the tests in this stage are discussed in Sec. 3.5.

3.4 Pilot-in-the-loop testing

Before the flight test, piloted real-time simulation studies are performed to verify and validate design requirements such as handling qualities (HQ). Moreover, the studies can be used in order to familiarize ourselves with general in-flight behavior, test behavior during switching on and off the control system, as well as during switching between functions. In addition, simulator sessions are very helpful in preparing test procedures to ensure a smooth operation for the actual flight tests. The Institute of System Dynamics and Control's Robotic Motion Simulator (RMS) [42] is used as a platform for flight test preparation via non-linear simulations [43].

The RMS is a seven DoF industrial robot, depicted in Fig. 11, which seats one person and can be equipped with aircraft controls to provide the user with a real-time response in terms of accelerations, body rates, and attitudes. The pilot input, given via side-stick and thrust levers, is processed in the primary simulation model, and its 6 DoF outputs are delivered as inputs to the RMS. Filtering algorithms ensure the gravity vector is always pointing in the direction of the stationary acceleration vector and higher frequency accelerations are depicted via translation and rotation of the gondola seen in Fig. 11. Visual information is provided to the user via virtual reality glasses or in-cabin projections. It includes a cockpit view perspective of the flight trajectory and a primary flight display, which can be customized with test-relevant parameters. As already mentioned, switching behavior is of particular importance. Unlike control laws on production aircraft, experimental ones are switched on and off multiple times during a test flight, as there may be intervention by the safety pilot, air traffic control (ATC) may request a change of course while no corresponding function is active, or reconfiguration for a next test point may be needed. A basic requirement on any control function tested is that switching it on does not cause any transients. Especially in the case of the first flights of algorithms, initial transients may give cause for concern, leading to premature disengagement. The same holds to a lesser extent for disengaging functions, as trim loading of control surfaces may occur.

Fig. 11 Robotic Motion Simulator at the DLR Institute of System Dynamics and Control in a multi-domain flight simulation with coupled flight dynamics and energy systems (shown on the displays) [43]

[3] https://de.mathworks.com/products/embedded-coder.html

3.5 HIL Ground Testing

After successfully executing the piloted simulation studies, the controller code is prepared to be embedded in the overall aircraft software. Thereby, the C code generated out of the SIMULINK® controller model by using the Embedded Coder is integrated into the DUECA framework of the aircraft. DUECA is a middle-ware layer written in C++ that enables real-time implementation and communication [44]. It supports a modular design and takes care of the synchronization and exchange of data. The controller code is an additional module with the appropriate interfaces and sampling rate in the existing aircraft framework.

For the HIL test, the aircraft is powered on and all sensors, i.e. inertial measurement system (INS), air data and control surface position measurement deliver signals to the flight control algorithm. A test program is defined for each flight test, depending on the functions present in the flight control software and their original design requirements (Sec. 3). An example HIL test program of the latest flight test campaign is listed in Table 4.

Test Item	**Procedure**	**Scope**
1. Enabling check	Enable the controller on ground in steady state.	Jolt-free enabling, bump-less transfer from open loop to closed loop.
2. Air Data Sensors	Move Alpha & Beta vane, check control surface response (direction).	Check correct wiring & signs of the sensor feedback inside FCS.
3. Control command check	Give stick & step inputs in all mode combinations which shall be tested	Check signs & responses of control system, do a qualitative check of anti-windup / hedging algorithms.

Table 4 HIL test items.

In item one, the enabling of the controller is carefully observed, i.e. when activating the controller, the algorithm should set the actual sensor values as initial, steady state, and thus no reactions (jolts) in the control surface reactions shall be observed. In item two, sensors such as angle of attack and side slip vanes can be moved to see if the control function aims to correct this in the right direction. This can easily be observed by looking at the control surface deflections. Even ground movements have been carried out to validate inertial measurements. Finally, side stick inputs can be given from the cockpit or a WiFi-linked computer [45] to validate directions of control surface movement. Additionally, when holding the input in this test case, the control surface position will integrate up to the saturation limits at some point since the aircraft's sensor signals stay constant. When changing the direction of the control input now, the functionality of the used hedging / anti-windup algorithms can be observed.

In the graphical interface of DUECA, the available parameters of the controller can be manipulated, and a switch between different controller variants is provided. For these tests described above and in Table 4, the controller software is executed on the flight hardware, while sensor data is read, and the actuators are commanded directly. Although this is obviously not a closed-loop test, very important sanity checks can be carried out. In this way, it is examined whether the software components are interfaced properly and whether the access of the actuators and processing of the incoming data works correctly. The tests take place up to right before each flight test to check the controller functionality. The data of the ground tests is also logged to be examined in detail to determine the controller's performance. Finally, the maneuvers planned for the flight test are already being run through on the ground. Any controller or interface issues can be detected by HIL testing and resolved before the flight. In this way, straightforward controller testing in different maneuvers in the air is enabled. A basic rule is that the final software loaded onto the FCC goes through the full ground test before flight.

4 Flight Testing

Flight test planning starts with compiling all test objectives for the functions to be tested (Sec. 2). Although procedures and best practices for control law flight testing are well established, combining multiple very different functions with different test crews in one flight is challenging. As the number of flight hours is limited, it is usually attempted to perform basic tests on all functions during the first flight already. Planning following flights with at least two days in between allows time to analyze and rectify any problems.

4.1 Flight Test Procedures

Since the experiments are very intensive in monetary and staff resources, it is desired to utilize as much as possible of the available flight time for testing and reduce idle time spent for maneuvering without an experiment running or re-configuring the hardware setup. Therefore, the flight test program is optimized to order each test case optimally so that the airspace is used efficiently (e.g., planning turns where the boundary of the flight test area demands a turn anyway). For that purpose, flight test cards are generated with information about each flight test case. A small example from the iADP related flight test card is presented in Table 5.

ID	Tag	Synopsis	Time	Init	CH	AP	CFG	Settings
A.1	iADP Lon Chk	3-2-1-1 elevator	110 s	IAS = 160 kts, FL=100	Pitch	AT off	clean	iADP var=LON, IMAR=on
A.2	iADP Lat Chk	aileron / rudder doublets	140 s		Roll + Yaw	AT off, pitch hold		iADP var=Lat, IMAR=on
A.3.1	iADP Lat	learning	140 s					
A.3.2		learning w equal model	140 s					
A.3.3		learning w new model	140 s				Gear dn, Flaps Up	
A.3.4		learning w equal model(2)	140 s				Gear dn, Flaps 15	

Table 5 Flight test card example.

From left to right, the first three columns provide a unique case identifier, a tag to group similar tests, as well as a short description of the test scenario. The flight tag should give a brief overview of what this test case is about and the synopsis describes the test in more detail. Additionally, it has been proven useful to estimate the time needed for each test (determined by e.g. simulations plus a small buffer for controller activation etc.) in order to calculate the total testing time and provide a reasonable amount of test cases per flight. The time column gives the expected duration of the test. Most tests would usually start at the same indicated airspeed and pressure altitude, but the trajectory following experiment (see section 2.3.4) requires a custom starting point since a pre-programmed path is to be followed. The next piece of information is mostly intended for the pilots and the flight test engineer in charge of coordinating the tests. Those are the initial aircraft state ("Init"), controller settings ("CH" and "AP") and aircraft configuration ("CFG"). The "AT" in the controller settings mentions, if the pilots should enable any of the aircraft's built-in autopilot modes on the currently un-commanded axis, or if they should follow the auto-throttle target on their additional display in Fig. 12. The term "CFG" indicates the configuration (e.g. gear down, flaps 15, clean) of the aircraft. The "Settings" column shows the flight test engineer what settings should be selected for the specific controller for each test case. Additionally, there is some

space for hand written comments at the end of the table. Both, pilots and flight test engineers use the same flight test card. The pilots do not see the "Special Conditions" and "Comments" column since they cannot change any controller modes nor do they have enough time to write down comments (they would communicate their comments via intercom).

A further important element are the displays used by the pilots and the flight test engineers. With the help of the DLR Visualization [46] and the Flight Visualization [47] libraries, a dedicated graphical instrument is developed to display all relevant sensor data for each function tested, see Fig. 12. Furthermore, additional information, such as internal controller variables and set points are marked, e.g the thrust demand in the lower central part of the display. During flight in autopilot mode, the pilots need to make the actual N1 engine speed (in percent) match the demand by moving the throttle levers.

Shortly before getting ready to enter the plane, the engineers should perform a pre-flight check to ensure all necessary equipment is onboard and functional, and all software is up to date, in the desired configuration, and fully ground-tested as in the ground test plan. During flight planning and pre-flight briefing, special attention is paid to communication onboard during experiments. As already said, with multiple functions with individual test engineers, this posed a challenge to take several iterations to optimize.

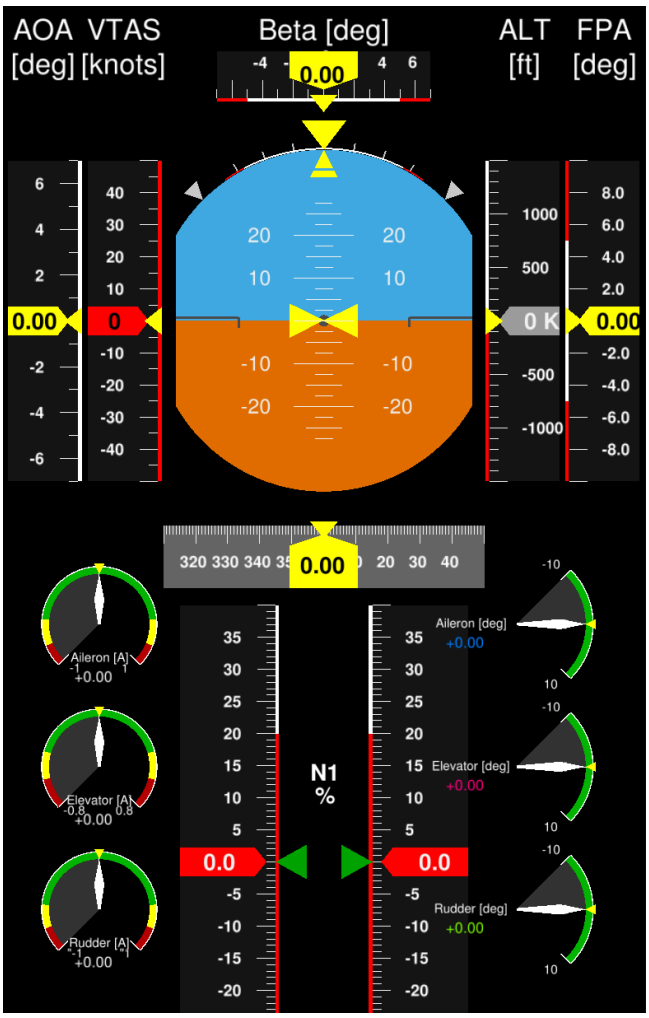
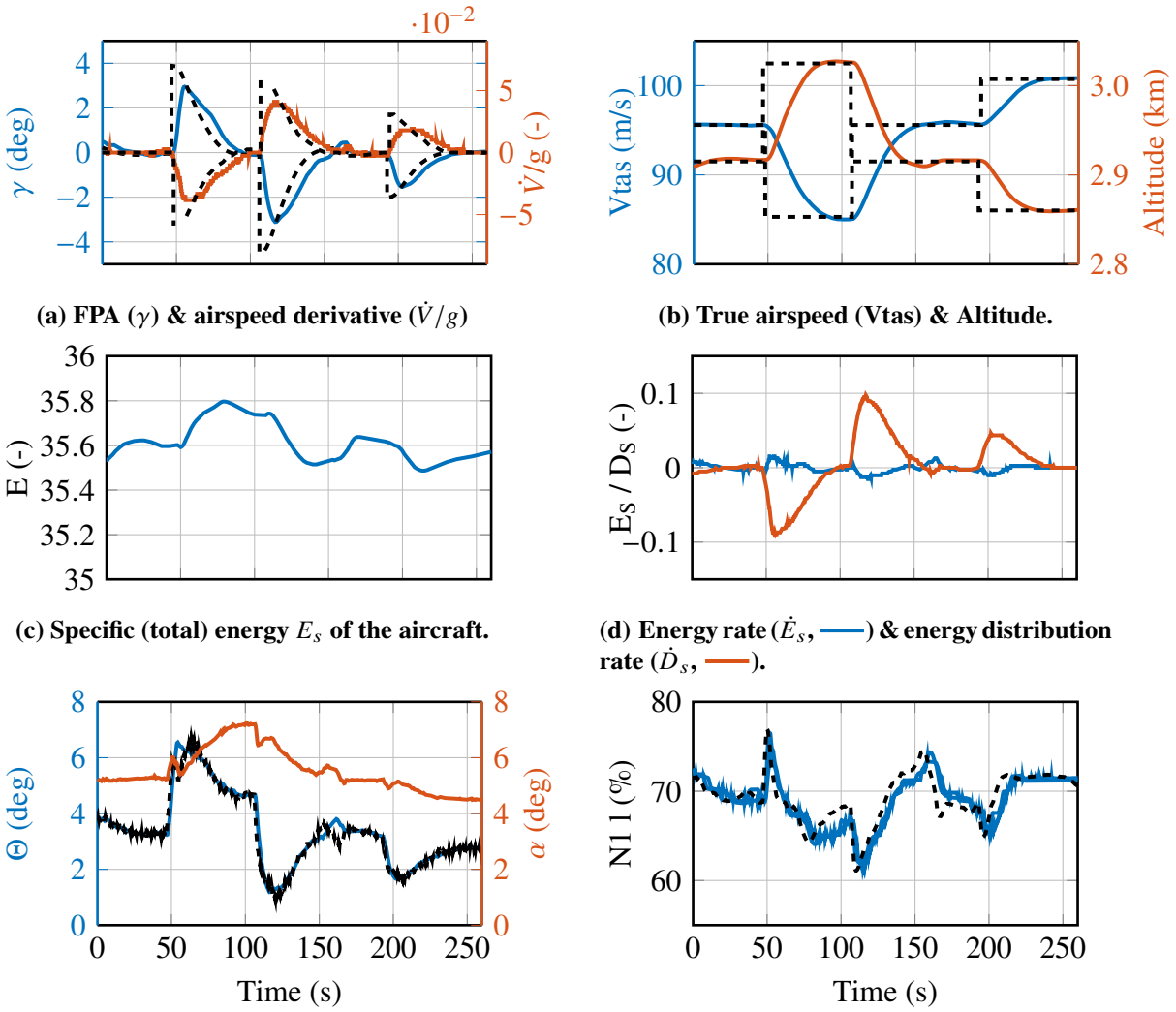


Fig. 12 Pilot display

Special attention is also given to functions or methods that have never been in a flight test before. Methods like INDI were found quite critical, as this method relies on the internal synchronization of sensor signals. This is difficult to test offline. For example, a procedure for a first-time test of a INDI-based control function is given in [6]. As no structural model of the aircraft is available, no computational analysis of flight loads and structural dynamics can be performed before flight. Although this is deemed relatively uncritical for the aircraft and the control authority of the FBW system, we do closely monitor the airframe and controls (surfaces and inceptors) for any suspect behavior in this respect. As an example, test results from the latest campaign are presented in the following Sec. 4.2. Further test data can be found in e.g. [15, 22] (INDI), [14] (LPV), [17] (hINDI) and [19] for the iADP flight tests. In order to analyze flight tests during the short breaks between flights, extensive plotting tools have been used to extract and display any relevant parameter or sensor output from recorded flight test data.

4.2 TECS Flight Test Data

For the TECS autopilot, a maneuver that exchanges potential with kinetic energy is performed as one of the vital flight test experiments. For this purpose, simultaneous steps in altitude (+100 m) and airspeed (-10 m/s) are commanded in Fig. 13. This is done such that the sum of the total and potential energy of the aircraft remains equal, implying that no change of throttle setting (as the source of energy) should be necessary.



(a) FPA (γ) & airspeed derivative ($\dot{V}/g$)

(b) True airspeed (Vtas) & Altitude.

(c) Specific (total) energy E_s of the aircraft.

(d) Energy rate ($\dot{E}_s$, ——) & energy distribution rate ($\dot{D}_s$, ——).

(e) Pitch attitude (Θ) & AoA (α).

(f) N1 command vs. actual values.

Fig. 13 Flight test data of three flown energy exchange maneuvers with TECS controller active.

From Fig. 13d, it can be seen that the time response characteristics of the total energy rate and distribution terms, $\dot{E}$ and $\dot{D}$, indeed show that the amount of additional requested power is quite small compared to the re-distribution rate of energy. The total energy and energy distribution rates are defined as follows:

$$\dot{E} = mg\dot{h} + mV\dot{V} \simeq mgV\left(\gamma + \frac{\dot{V}}{g}\right). \tag{4}$$

$$\dot{D} = mg\dot{h} - mV\dot{V} \simeq mgV\left(\gamma - \frac{\dot{V}}{g}\right). \tag{5}$$

5 Summary and Outlook

This paper provides detailed insight into current flight test activities to validate the performance of new control functions and methods. The principal aim is to mature these to a level that makes them valid options for future aircraft programs, or at least to explore the challenges that need to be tackled before achieving this. Considerable effort has been spent to develop procedures, ways of working, planning, methods, and tools that allow for valuable testing of multiple, possibly highly dissimilar functions in a single flight or campaign.

We find the TU-Delft Cessna Citation PH-LAB with its experimental Fly-by-Wire system a highly valuable platform for maturing and flight testing of new control functions and methods. So far, this has facilitated us to perform "maiden flights" of various of those.

Control authority is somewhat reduced due to limited control power of the installed servos that come with the original autopilot system. We often found this an advantage, as this forces us to a more mature design that is able to address these limitations in a safe way. As the servo control loops are closed on the experimental computer, this also allowed us to successfully demonstrate concepts like torque-controlled actuation [16].

Even though the aircraft configuration is fixed, we plan to extend the simulation model with redundant controls, structural dynamics, as well as loads computation. This would allow us to not only mature control laws towards flight test, but also to perform model-based assessment of fault tolerance and cross-disciplinary (e.g. flight loads) interactions within the same framework as described in this paper.

Acknowledgments

The authors want to acknowledge the teams of TU Delft that made the whole flight test campaign possible and provided their flight test framework and valuable expertise, namely Menno Klaassen, Fred den Toom, Olaf Stroosma, René van Paassen, and Ferdinand Postema. Last but not least the authors want to thank the TU Delft test pilots, Alexander in't Veld and Hans Mulder, who showed patience, calm, and fun and provided us with a great and safe flight test experience.

References

[1] Daniel Milz and Gertjan Looye. Tilt-wing control design for a unified control concept. In *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, 2022. DOI: 10.2514/6.2022-1084.

[2] Marc S. May, Daniel Milz, and Gertjan Looye. Semi-empirical aerodynamic modeling approach for tandem tilt-wing evtol control design applications. In *AIAA SciTech 2023 Forum*. American Institute of Aeronautics and Astronautics (AIAA), 2023. DOI: 10.2514/6.2023-1529.

[3] Thiemo Kier and Gertjan Looye. Unifying manoeuvre and gust loads analysis. In *IFASD 2009*, Juni 2009. `https://elib.dlr.de/97798/`.

[4] Christian Weiser, Simon Schulz, Arne Voß, and Daniel Ossmann. Attitude control for HALE aircraft considering structural load limits. In *AIAA SciTech 2023 Forum*, 2023. DOI: 10.2514/6.2023-0106.

[5] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, nov 2010. DOI: 10.2514/1.49978.

[6] Daniel Milz, Gertjan Looye, and Marc May. Dynamic inversion: An incrementally evolving methodology for flight control design. *to be published*, 2024.

[7] Thiemo M. Kier, Reiko Müller, and Gertjan Looye. Analysis of automatic control function effects on vertical tail plane critical load conditions. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, 2020. DOI: 10.2514/6.2020-1621.

[8] Ewoud J. J. Smeur, Qiping Chu, and Guido C. H. E. de Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016. DOI: 10.2514/1.G001490.

[9] Ralph D. Kimberlin. *Flight Testing of Fixed-Wing Aircraft*. American Institute of Aeronautics and Astronautics, Inc., 2003. DOI: 10.2514/4.861840.

[10] United States Air Force. Flight test engineering handbook aftr no. 6273. Technical report, Edwards Air Force Base, CA, 1966.

[11] Gertjan Looye. Flight testing of autopilot control laws: Fly the helix! In *Deutscher Luft- und Raumfahrtkongress 2009*, 2009. `https://elib.dlr.de/62017/`.

[12] Wim van Ekeren, Gertjan Looye, Richard O. Kuchar, Q Ping Chu, and Erik-Jan Van Kampen. Design, implementation and flight-tests of incremental nonlinear flight control methods. In *2018 AIAA Guidance, Navigation, and Control Conference*, 2018. DOI: 10.2514/6.2018-0384.

[13] Christian Weiser, Daniel Ossmann, and Matthias Heller. In-flight validation of a robust flight controller featuring anti-windup compensation. In *2018 Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, 2018. DOI: 10.2514/6.2018-2982.

[14] Christian Weiser, Daniel Ossmann, Richard O. Kuchar, Reiko Müller, Daniel M. Milz, and Gertjan Looye. Flight testing a linear parameter varying control law on a passenger aircraft. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, jan 2020. DOI: 10.2514/6.2020-1618.

[15] Twan Keijzer, Gertjan Looye, Q Ping Chu, and Erik-Jan Van Kampen. Design and flight testing of incremental backstepping based control laws with angular accelerometer feedback. In *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, jan 2019. DOI: 10.2514/6.2019-0129.

[16] Tijmen Pollack, Gertjan Looye, and Frans Van der Linden. Design and flight testing of flight control laws integrating incremental nonlinear dynamic inversion and servo current control. In *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, jan 2019. DOI: 10.2514/6.2019-0130.

[17] Daniel Milz, Marc May, and Gertjan Looye. Flight testing air data sensor failure handling with hybrid nonlinear dynamic inversion. In *Proceedings of the 2024 CEAS EuroGNC conference*, 2024.

[18] Ramesh Konatala, Erik-Jan Van Kampen, and Gertjan Looye. Reinforcement learning based online adaptive flight control for the cessna citation II(PH-LAB) aircraft. In *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, 01 2021. DOI: 10.2514/6.2021-0883.

[19] Ramesh Konatala, Erik-Jan Van Kampen, Gertjan Looye, Daniel Milz, and Christian Weiser. Flight testing reinforcement learning based online adaptive flight control laws on cs-25 class aircraft. In *AIAA Scitech 2024 Forum*. American Institute of Aeronautics and Astronautics, 01 2024. DOI: 10.2514/6.2024-2402.

[20] Ramesh Konatala. Verification & validation of reinforcement learning based online adaptive flight control laws on cs-25 class aircraft. In *Proceedings of the 2024 CEAS EuroGNC conference*, jun 2024.

[21] Cornelis Mattheus Vlaar. Incremental nonlinear dynamic inversion flight control implementation and flight test on a fixed wing uav. Master's thesis, Delft University of Technology, 2014.

[22] Fabian Grondman, Gertjan Looye, Richard O. Kuchar, Q Ping Chu, and Erik-Jan Van Kampen. Design and flight testing of incremental nonlinear dynamic inversion-based control laws for a passenger aircraft. In *2018 AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, jan 2018. DOI: 10.2514/6.2018-0385.

[23] Peter Zaal, Daan Pool, Alexander in 't Veld, Ferdinand Postema, Max Mulder, Marinus van Paassen, and Jan Mulder. Design and certification of a fly-by-wire system with minimal impact on the original flight controls. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, jun 2009. DOI: 10.2514/6.2009-5985.

[24] European Aviation Safety Agency. *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes CS-25*, 2018.

[25] A. Muis, J. Oliveira, and J. A. Mulder. Development of a flexible flight test instrumentation system. In *17th SFTE (EC) Symposium*, Amsterdam, Netherlands, 2006.

[26] Gertjan Looye Christian Weiser, Daniel Ossmann. Flight testing total energy control autopilot functionalities for high altitude aircraft (accepted). In *AIAA Scitech 2024 Forum*. American Institute of Aeronautics and Astronautics, 2024. DOI: 10.2514/6.2024-2204.

[27] Gertjan Looye and Hans-Dieter Joos. Design of autoland controller functions with multiobjective optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 29(2):475–484, March–April 2006. DOI: 10.2514/1.8797.

[28] T. J. J. Lombaerts, H. O. Huisman, Q. P. Chu, J. A. Mulder, and D. A. Joosten. Nonlinear reconfiguring flight control based on online physical model identification. *Journal of Guidance, Control, and Dynamics*, 32(3):727–748, may 2009. DOI: 10.2514/1.40788.

[29] Gertjan Looye and Hans-Dieter Joos. Design of robust dynamic inversion control laws using multi-objective optimization. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2001. DOI: 10.2514/6.2001-4285.

[30] P. Smith. A simplified approach to nonlinear dynamic inversion based flight control. In *23rd Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, aug 1998. DOI: 10.2514/6.1998-4461.

[31] Xuerui Wang, Erik-Jan van Kampen, Qiping Chu, and Peng Lu. Stability analysis for incremental nonlinear dynamic inversion control. *Journal of Guidance, Control, and Dynamics*, 42(5):1116–1129, may 2019. DOI: 10.2514/1.g003791.

[32] Thomas Lombaerts and Gertjan Looye. Design and flight testing of nonlinear autoflight control laws. In *AIAA Guidance, Navigation, and Control Conference*, 2012. DOI: 10.2514/6.2012-4982.

[33] Nir Kastner and Gertjan Looye. Generic TECS based autopilot for an electric high altitude solar powered aircraft. In *Proceedings of the 2013 CEAS EuroGNC conference*, pages 1324–1343, 2013. `https://elib.dlr.de/83888/`.

[34] Antonius A. Lambregts. Vertical flight path and speed control autopilot design using total energy principles. In *Guidance and Control Conference*. American Institute of Aeronautics and Astronautics, 1983. DOI: 10.2514/6.1983-2239.

[35] K. R. Bruce, J. R. Kelly, and L. Jr. Person. NASA B737 flight test results of the Total Energy Control System. In *AIAA Astrodynamics Conference*, 1986. DOI: 10.2514/6.1986-2143.

[36] Christian Weiser and Daniel Ossmann. Baseline flight control system for high altitude long endurance aircraft. In *AIAA SciTech 2022 Forum*. American Institute of Aeronautics and Astronautics, 2022. DOI: 10.2514/6.2022-1390.

[37] Reiko Müller. The Gnomonic Projection for B-Spline parameterized 4-D Trajectory Optimization Problems. In *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, 2020. DOI: 10.2514/6.2020-3205.

[38] Reiko Müller. *Four-dimensional multi-objective trajectory optimization using high-fidelity aircraft models and a projected parameterization*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2021. `https://publications.rwth-aachen.de/record/841766`.

[39] Hans-Dieter Joos. Worst-case parameter search based clearance using parallel nonlinear programming methods. In Andreas Varga, Anders Hansson, and Guilhem" Puyou, editors, *Optimization Based Clearance of Flight Control Laws: A Civil Aircraft Application*, volume 416, pages 149–159. Springer Berlin Heidelberg, Berlin, Heidelberg, 01 2012. ISBN: 978-3-642-22626-7. DOI: 10.1007/978-3-642-22627-4_8.

[40] Andras Varga. Optimisation-based clearance. In Christopher Fielding, Andras Varga, Samir Bennani, and Michiel Selier, editors, *Advanced Techniques for Clearance of Flight Control Laws*, pages 107–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN: 978-3-540-45864-7. DOI: 10.1007/3-540-45864-6_7.

[41] Andras Varga. Optimisation-based clearance: The linear analysis. In Christopher Fielding, Andras Varga, Samir Bennani, and Michiel Selier, editors, *Advanced Techniques for Clearance of Flight Control Laws*, pages 385–413. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN: 978-3-540-45864-7. DOI: 10.1007/3-540-45864-6_21.

[42] Tobias Bellmann, Johann Heindl, Matthias Hellerer, Richard Kuchar, Karan Sharma, and Gerd Hirzinger. The DLR robot motion simulator part I: Design and setup. In *IEEE International Conference on Robotics and Automation*, may 2011. DOI: 10.1109/icra.2011.5979913.

[43] Andreas Seefried, Alexander Pollok, Richard Kuchar, Matthias Hellerer, Martin Leitner, Daniel Milz, Christian Schallert, Thiemo Kier, Gertjan Looye, and Tobias Bellmann. Multi-domain flight simulation with the DLR Robotic Motion Simulator. In *2019 Spring Simulation Conference, SpringSim 2019*, Mai 2019. `https://elib.dlr.de/127688/`.

[44] M. van Paassen, Olaf Stroosma, and J. Delatour. DUECA - data-driven activation in distributed real-time computation. In *Modeling and Simulation Technologies Conference*. American Institute of Aeronautics and Astronautics, aug 2000. DOI: 10.2514/6.2000-4503.

[45] Alexander in't Veld, Hans Mulder, and Gertjan Looye. Flight tests on fault-tolerant autopilot control laws in laboratory aircraft Citation II. In *Society of Flight Test Engineers – European Chapter Symposium*, 05 2018.

[46] Sebastian Kümper, Matthias Hellerer, and Tobias Bellmann. DLR Visualization 2 library - real-time graphical environments for virtual commissioning. In Martin Sjölund, Lena Buffoni, Adrian Pop, and Lennart Ochel, editors, *Proceedings of 14th Modelica Conference 2021*, pages 197–204. Modelica Association and Linköping University Electronic Press, 2021. DOI: 10.3384/ecp21181197.

[47] Daniel Milz, Christian Weiser, Franciscus van der Linden, Matthias Hellerer, Andreas Seefried, and Tobias Bellmann. Advances in flight dynamics modeling and flight control design by using the DLR Flight Visualization and Flight Instruments libraries. In *Proceedings of the 13th International Modelica Conference*, volume 157 of *Linköping Electronic Conference Proceedings*, pages 481–488. Linköping University Electronic Press, Linköpings universitet, 2019. DOI: 10.3384/ecp19157481.