FliPASED
WWW.FLIPASED.EU

D1.6 Data Analytics for Model Validation

Thiemo Kier, Yasser Meddaikar, Matthias Wuestenhagen (DLR), Charles Poussot-Vassal (ONERA), Sebastian Koeberle, Julius Bartasevicius, Fanglin Yu (TUM), Balint Vanek, Daniel Balogh, Tamas Luspay, Bela Takarics (SZTAKI)

GA number: 815058
Project acronym: FLIPASED
Project title: FLIGHT PHASE ADAPTIVE AEROSERVO-ELASTIC AIRCRAFT DESIGN METHODS
Funding Scheme: H2020 **ID:** MG-3-1-2018
Latest version of Annex I: 1.1 released on 12/04/2019
Start date of project: 01/09/2019 **Duration:** 40 Months

Lead Beneficiary for this deliverable: SZTAKI
Last modified: 17/11/2022 **Status:** Delivered
Due date: 30/09/2021

Project co-ordinator name and organisation: Bálint Vanek, SZTAKI
Tel. and email: **+36 1 279 6113** vanek@sztaki.hu
Project website: www.flipased.eu

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

“This document is part of a project that has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 815058.”

Glossary

ASE	Aeroservoelastic
CAD	Computer-aided Design
CPACS	Common Parametric Aircraft Configuration Schema
DLM	Doublet Lattice Method
FEM	Finite Element Model
SW	Soft-ware
HW	Hard-ware
VV	Verification and Validation
GLA	Gust Load Alleviation
MLA	Manoeuvre Load Alleviation
MDO	Multidisciplinary Design Optimization
MLA	Manoeuvre Load Alleviation
PID	Proportional-Integral-Derivative
RCE	Remote Component Environment
HIL	Hardvare-in-the-loop
FCC	Flight Control Computer
TCL	Tool Command Language

Table of contents

List of Figures

1 Executive Summary

The following results are based on preliminary findings.

The deliverable ”D1.6 Data Analytics for Model validation” focuses on comparing results and findings coming from different sources. The main reason to have specific assessment of results coming from theoretical models or experimental tests is to build confidence in the developed tools and methods. The data from flight tests will serve as a baseline to validate structural dynamics, aerodynamics, controls and avionics instrumentation models. Analysis tools with standard validation routines will be provided in Nastran and Matlab environment for structural dynamics and controls respectively. These tools along with Python based data science software will be used within the project and the underlying theory along with interfaces of these tools are documented in D1.6.

The project goal, set in the proposal within Task 1.4 Data Analytics for Model Validation (SZTAKI, DLR,ONERA,TUM) aims at: ”*Significant part of the engineering effort is devoted in research projects to provide the adequate interface for tools and methods developed in prior projects. These tools and the corresponding analysis steps with their interfaces will be part of the open data initiative, to provide seamless access to the core problems with adequate analysis tools to the research community. The data from flight tests in the early part of the project, provided by re-using the demonstrator platform developed within FLEXOP, will serve as a baseline to validate structural dynamics, aerodynamics, controls and avionics instrumentation models. Analysis tools with standard validation routines will be provided in Nastran and Matlab environment for structural dynamics and controls respectively. During the second half of the project these tools will be expanded to address the attainment of MDO criteria, during the development cycle – including the functions of weight reduction, fuel efficiency and gust load alleviation. Based on the large amount of simulation and experimental data both analytical and data driven approaches will be pursued for model predictive control, function shape fitting by support vector machines and deep learning, parameter search by Monte Carlo methods, and more. The project will also use Python based data science software, including numpy, scipy, pandas, scikit-learn, Tensorflow, Keras, matplotlib and many more, in Jupyter notebooks, as the emerging de facto standard sharing and collaboration tool for data scientists.*”

2 Overall Architecture and Tools to connect MDO and Testing

This section describes the overall structure of MDO toolchain and the tools used there. A short introduction regarding each blocks is also given. The connection between the MDO, HIL test and flight test toolchains is also explained in this section.

2.1 Overall Architecture and Tools of MDO Toolchain

The MDO toolchain is the main block in this case which has its own optimization and gets back to the CPACS generation block after each iteration. The main goal of the MDO toolchain is to show the improvements of the optimization, which involves aircraft geometry, sizing, modeling and control design simultaneously, with respect to the reference aircraft. Figure 1 shows the overall architecture of MDO toolchain.

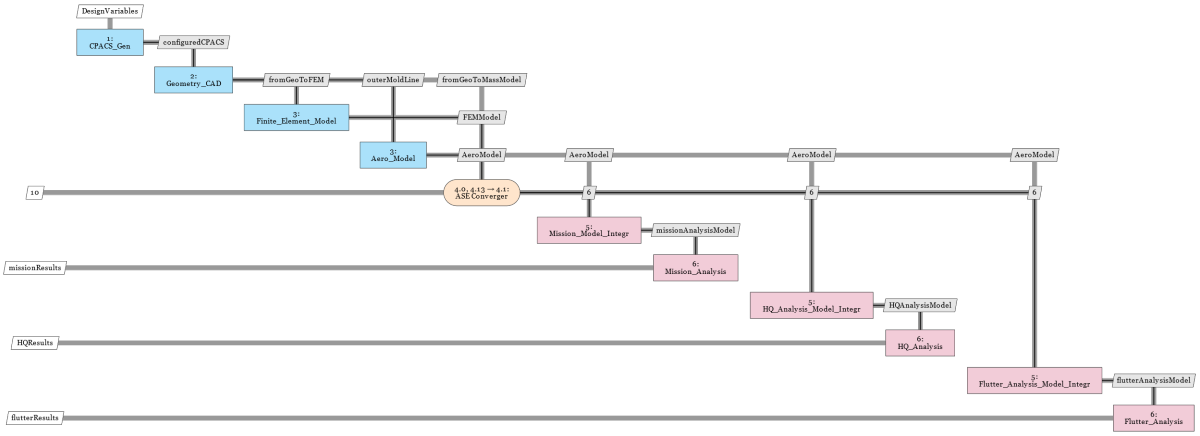


Figure 1: MDO Toolchain for demonstrator T-FLEX

The following sections will give a brief introduction of function blocks in MDO toolchain and the used standard tools. For more informations please refer to previous deliverables 1.2, 1.4, 2.2 and 4.1.

2.1.1 CPACS

The data model CPACS has been introduced and developed at the German Aerospace Center (DLR) since 2005. CPACS is implemented in XML. Making use of the hierarchical representation of data in XML the structure of CPACS mainly follows a top-down approach which decomposes a generic concept (e.g., an aircraft) into a more detailed description of its components. This originates from the conceptual and preliminary design of aircraft, where the level of detail is initially low and continues to increase as the design process progresses. The hierarchical structure furthermore promotes the simplicity of the exchange format which is required in collaborative design environments so that the various stakeholders can easily append their results. CPACS serves as the data model in this toolchain.

2.1.2 RCE

DLR's Remote Component Environment (RCE) [4] is an open-source software environment for defining and executing workflows containing distributed simulation tools by integrating them into a peer-to-peer

network. In this toolchain, RCE is used as the integration platform.

2.1.3 CPACS generation block

CPACS generation block, which is the first block in the MDO toolchain, aims to generate the first version of CPACS file with a Python script.

2.1.4 Geometry block

Geometry block aims to update the Catia model based on the incoming CPACS file from the upstream CPACS generation block.

2.1.5 FE-model block

The function of the FE-model block is meshing the geometry model and assigning structural properties. A Splining model, which couples the structural and aerodynamic model, is also generated in this block.

2.1.6 Aero-model block

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

2.1.7 Aeroelastic Model Generation and Simulation

Based on the aerodynamic, structural and spline grid information as well as mass and stiffness matrices, simulink model will be generated in this block and will be used for control synthesis design.

2.1.8 Baseline and Flutter Suppression Control Design Blocks

This block takes the nonlinear Simulink model with the configuring struct file from DLR-SR as the input files and generates two models in this case, one for the flutter control synthesis block and one for the baseline controller. Based on the model the flutter suppression controller and the baseline controller are generated. Once the flutter suppression and baseline control design blocks finish the synthesis, a frequency domain analysis will be ran to assesses the performance of the two controllers acting together simultaneously, to check the robustness margins and flutter margins of the resulting controllers.

In this MDO toolchain one of the main focus for the model generation, model reduction blocks and control design blocks are the robustness aspects of the underlying algorithm. These need to run automatically, without human interaction in the presence of changes in the aircraft. This comes at an expense that the individual controllers do not achieve the highest possible performance. This is also where testing comes into play to validate the developed methodologies.

2.2 Connection between MDO Toolchain and Testing

The HIL test and flight test blocks serve as auxiliary tools to validate the developed methodologies, as shown in figure 2.

The HIL tests evaluate the implementation aspects of the controllers and serve as a final step before flight testing the controllers.

The flight test goals are twofold in case of the MDO toolchain. The main goal is to validate the control design technology maturity. This is especially valid for the MLA, GLA and flutter suppression controllers of the project since they have not been flight tested yet (using the model based design methodology within FLiPASED). The other goal of the flight tests, as opposed to the MDO toolchain, is that the resulting controllers can be fine tuned by ”hand” to achieve optimal performance and provide lessons-learnt to the designers and to the aviation community in general. In this case the robustness of the

synthesis algorithms to be able to be run in an automatic manner is not of a paramount criterion. In addition, the fine tuning of the controllers is to be done based on the aircraft model that has been updated via flight test data.

At the end of the cycle, the lessons learnt from the HIL tests and flight tests need to be fed back to the MDO toolchain. This is done via engineering considerations. If the HIL tests indicate that some controller has implementation difficulties, the corresponding control design algorithm needs to be updated. Similarly, if the flight tests show that a controller has lack of performance or robustness during flight tests, the algorithms need to be adjusted as well.

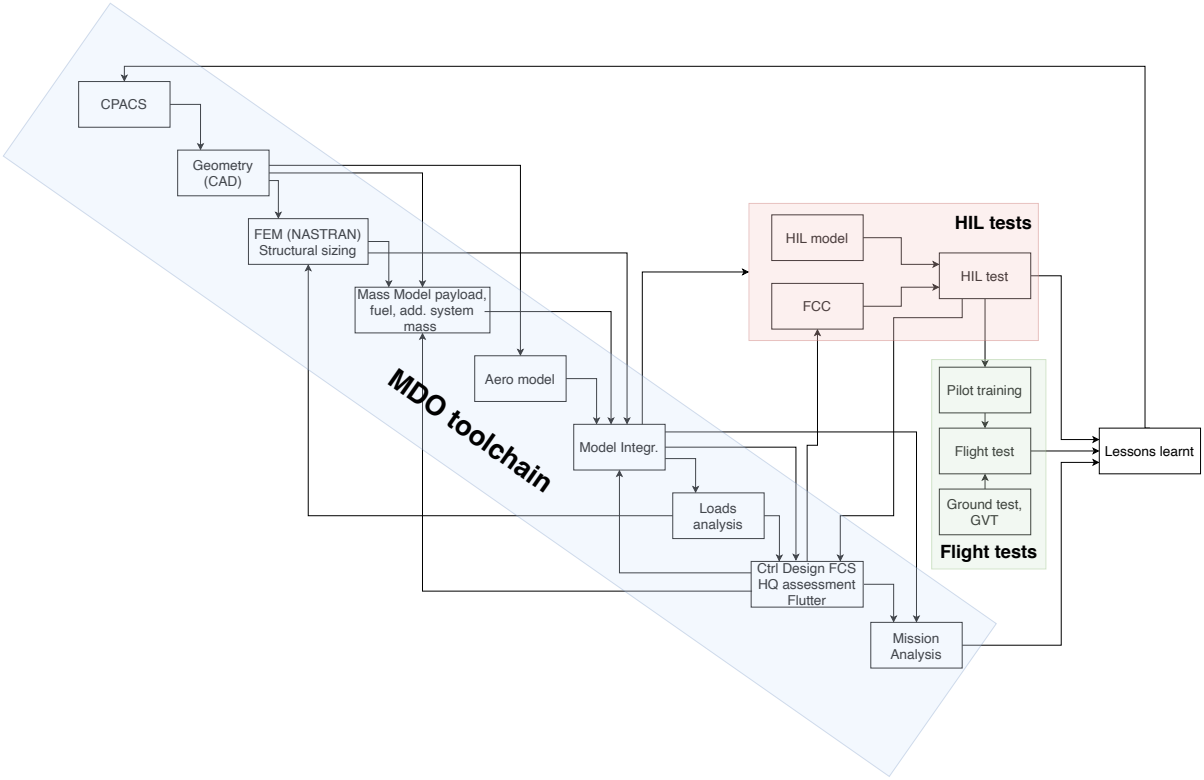


Figure 2: Toolchains developed in FLiPASED

More details regarding toolchain validation will be given in the following sections.

3 Structural Dynamics Model Validation

The tasks related to structural dynamics of the aircraft models are led by DLR-AE, but contributions are made by ONERA, TUM, SZTAKI and DLR-SR as well.

The main steps regarding the task are:

- structural model development and GVT based update
- Model comparison and fine tuning for RCE toolchain based and GVT based model matching
- Operational modal analysis based model update during flight tests and its connection how this feeds back to NASTRAN models
- Description of used tools and how they can we standardized

In this chapter, a summary of the structural dynamics model and the model-updating activities pertaining to its update are described.

3.1 NASTRAN structural dynamic model

The structural dynamic models of the T-Flex aircraft are developed using a modelling toolchain established during FLEXOP and FLIPASED. In total, three pairs of wings are designed, manufactured and tested on the UAV test-bench:

(i) wings -0 - a pair of wings optimized using balanced-symmetric type of laminates serving as the reference wing

(ii) wings -1 - a pair of flutter wings designed to trigger flutter within the test-regime, whose flight envelope will then be extended using active flutter control

(iii) wings -2 - a pair of wings optimized using unbalanced composite laminates, to demonstrate passive load alleviation through aeroelastic tailoring

The structural FE models for the wing pairs -0 and -2 are generated using an in-house model generator ModGen at DLR-AE [17], while those of the -1 wing are obtained from a CAD-FEM toolset at TUM. The wing models are integrated to the fuselage and empennage models generated during FLEXOP at DLR-AE. The fuselage and empennage models are also generated using ModGen [17].

3.2 Model-updating of the -0 wings

A ground-test campaign [35] involving structural tests and ground vibration tests (GVT) has been performed on the T-Flex aircraft. An update of the FE model of the -0 wing has been performed based only on experimental data from static tests, while an update using data from the GVT is being studied at present.

The static test was performed with the main objective being the assessment of the stiffness properties of the wing and validation of the pertinent structural models developed. Figure 3 shows the deflection of the wing-tip as a function of the applied tip-load. Shown in Figure 4 is the span-wise displacement

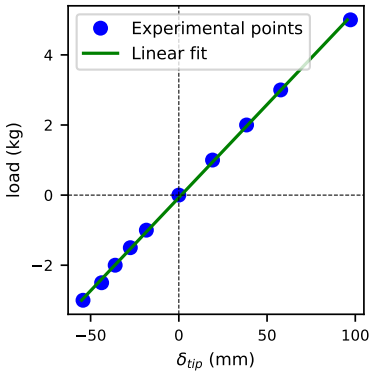


Figure 3: Displacement vs load at tip of the wing from static tests (-0 wing)

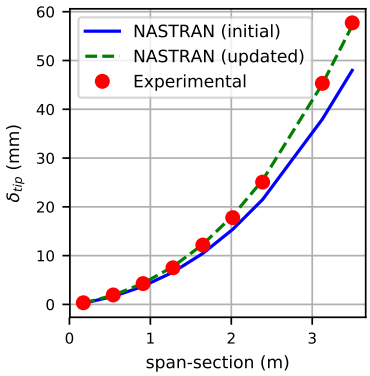


Figure 4: Span-wise displacement of wing under tip load for the updated model (-0 wing)

of a wing-half subjected to 3kg load at the tip, comparing the static tests and the initial FE model. The observed difference in the displacement could be attributed to several factors including modelling assumptions and simplifications, manufacturing deviations, material scatter, etc.

The present version of the model-updating is performed by introducing a knock-down on the engineering stiffness (E_1, E_2, G_{12}) in the FE model of the wing and in the clamp used for the wing attachment. An alternative approach through the inclusion of tuning-beams and optimizing its properties has also been attempted. A comparison of the frequencies between this stiffness-updated FE model and the GVT results is shown in Table 1. Also shown is the modal assurance criterion (MAC) which is an indicator of similarity between mode shapes from two sets in Figure 5. It is seen that the FE model captures the out-of-plane bending behaviour of the wing well. On the other hand, the in-plane behaviour of the wing and the stiffness and mass modelling of the fuselage and empennage need to be investigated in more detail.

The stiffness-updated structural model serves as the basis for generating a next iteration of ASE models for controller synthesis. In the next steps, a more refined approach at model-updating will need to be performed considering other possible sources of deviation such as an improved modelling of wing-fuselage joint, localized stiffness-updates and updated mass-modelling while utilizing also the frequencies and mode-shapes obtained from the GVT.

3.3 Comparison of -1 aircraft structural dynamic model with static test

Static test of -1 wing was conducted in the FLEXOP project at the same time as -0 and -2 wing to verify the stiffness properties of manufactured wing and validate the FE-Model developed during design stage. Figure 6 shows wing tip deflection at different load cases and their linear fit. Linear stiffness property can be clearly seen in the figure 6. Because of measurement error, there is zero drift when the load was increased from zero and decreased to zero again.

FE-model is elaborated to replicate the static test. Figure 7 shows the comparison of span-wise displacement of wing under 5 kg tip load between simulation and test. The manufactured wing is more flexible than it modelled. It shows same trend as the -0 and -2 wing. The deviation between simulation and test is around 12%, without consideration of zero drift in the test.

Same investigation was made for the torsional load cases. Figure 8 shows the linearity of the model

Mode	GVT (Hz)	FE (Hz)	Δf (%)
2n_wing_bending-s	3.37	3.27	-2.9
3n_wing_bending-a	8.28	8.35	0.9
1n_wing_inplane-a	8.88	18.45	-
4n_wing_bending-s	12.12	11.86	-2.1
tail_rock-a	17.32	-	-
1n_wing_inplane-s	19.26	18.09	-6.1

Table 1: Comparison of eigen frequencies of the flexible modes: GVT vs stiffness-updated FE model of the -0 aircraft (*in* - *i* nodes in the mode, s - symmetric, a - antisymmetric)

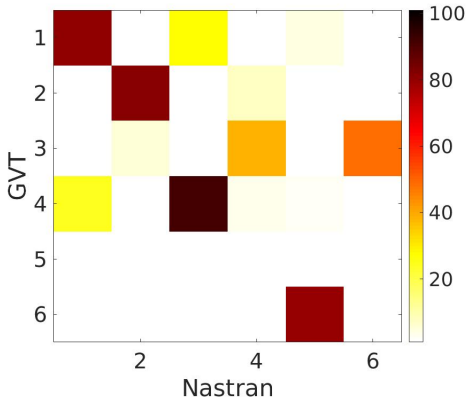


Figure 5: MAC matrix: GVT vs stiffness-updated FE model of the -0 aircraft

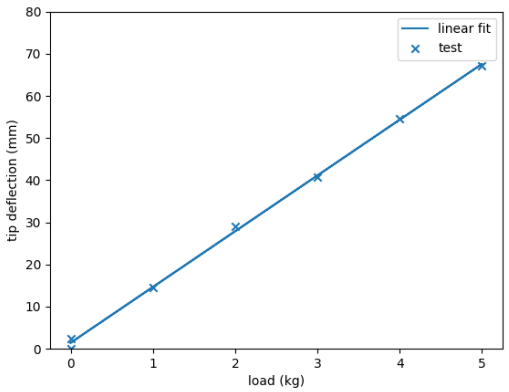


Figure 6: Displacement vs load at tip of the wing from static tests (-1 wing)

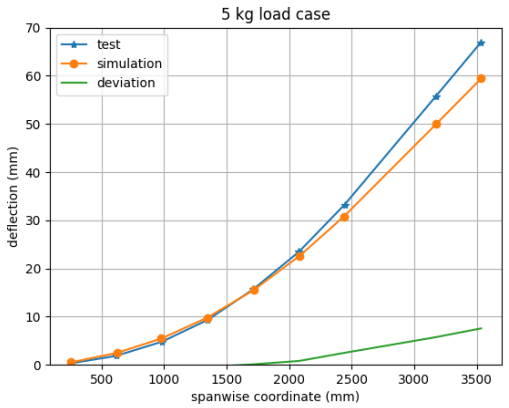


Figure 7: Span-wise displacement of wing under tip load (-1 wing)

under the torque loads. Figure 9 shows the comparison of span-wise torsion of wing under 2 kg torque load between simulation and test. There are only 0.1 deg differences. Taking the measurement error into account, the results match quite well.

3.4 Comparison of -1 aircraft structural dynamic model with GVTs

The -1 wing FE model is generated using a CATIA - Hypermesh toolset at TUM. The model is of a very high fidelity comprising of detailed elements for the structural as well as non-structural entities such as on-board systems.

A comparison of the eigen frequencies of the -1 aircraft model (without update) and the GVT is shown in Table 2. It is seen that a generally good agreement between the FE model and the GVT results exists. Two observations can be made with respect to this comparison.

The third flexible mode (3n_wing_bending-a) has the largest different in the experimental results with respect to the GVT. Given that this wing bending mode participates in the critical flutter mechanisms

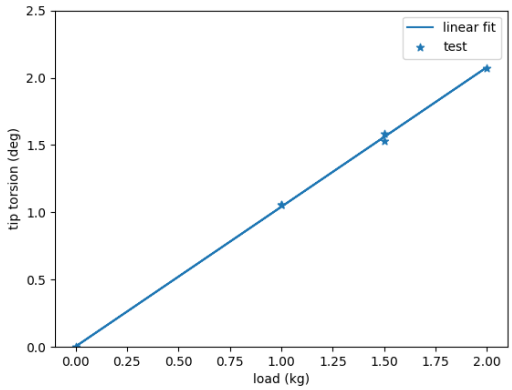


Figure 8: Torsion vs load at tip of the wing from static tests (-1 wing)

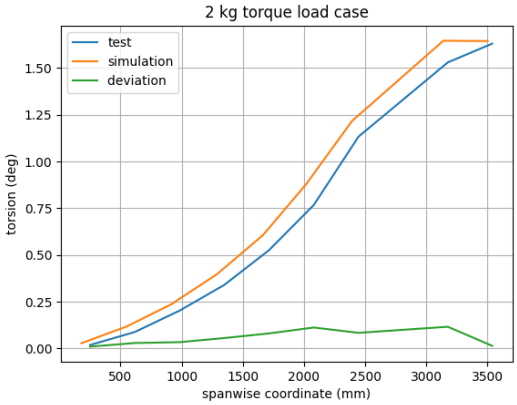


Figure 9: Span-wise torsion of wing under tip load (-1 wing)

Mode	GVT (hz)	FE (hz)	Δf (%)
2n_wing_bending-s	2.94	2.91	-1.02
1n_wing_in-plane-a	7.01	–	–
3n_wing_bending-a	7.57	8.15	7.66
wing_torsion-s	10.27	10.50	2.24
wing_torsion-a	10.73	10.61	-1.12
4n_wing_bending-s	12.13	12.11	-0.16
2n_wing_in-plane-s	15.07	15.06	-0.07

Table 2: Comparison of eigen frequencies of the flexible modes: GVT vs FE model of the -1 aircraft (*i*n - *i* nodes in the mode, s - symmetric, a - antisymmetric)

of the -1 aircraft, it is important to update the wing FE model with respect to the frequency of this concerned mode. Secondly, the second flexible mode (named 1n_wing_in-plane-a) which is observed during the GVT but not in the FE simulations appears to involve a relative motion between the fuselage and wing as shown in Figure 10. Such a mode is expected due to some free-play or softness in the attachment between the fuselage and wings, which is not tuned for in the FE models where an idealized attachment is assumed. In order to be able to simulate this mode, one approach would be to tune soft springs at the wing-fuselage interface such that the mode is present in the simulation. Both the FE model update mentioned in the former and a study on how to introduce the concerned mode discussed in the latter are being studied at present. An approach using so-called tuning beams is planned for this task.

3.5 Model-updating of the -1 wing

The model updating of -1 wing is first conducted with static test data. Knock-down factor is applied on the engineering stiffness (E_1, E_2, G_{12}) of the wing skin and spar. The model updating is based on the 3 kg bending load case. As you can see from Figure 11, the simulation result matches quite well with test data. The deviation with the test result is reduced to 2mm within the range of test error. Figure 12 shows the simulation result of 2kg torsional load case with updated model. There are no noticeable differences as expected, because the parameter is updated according to the bending load case.

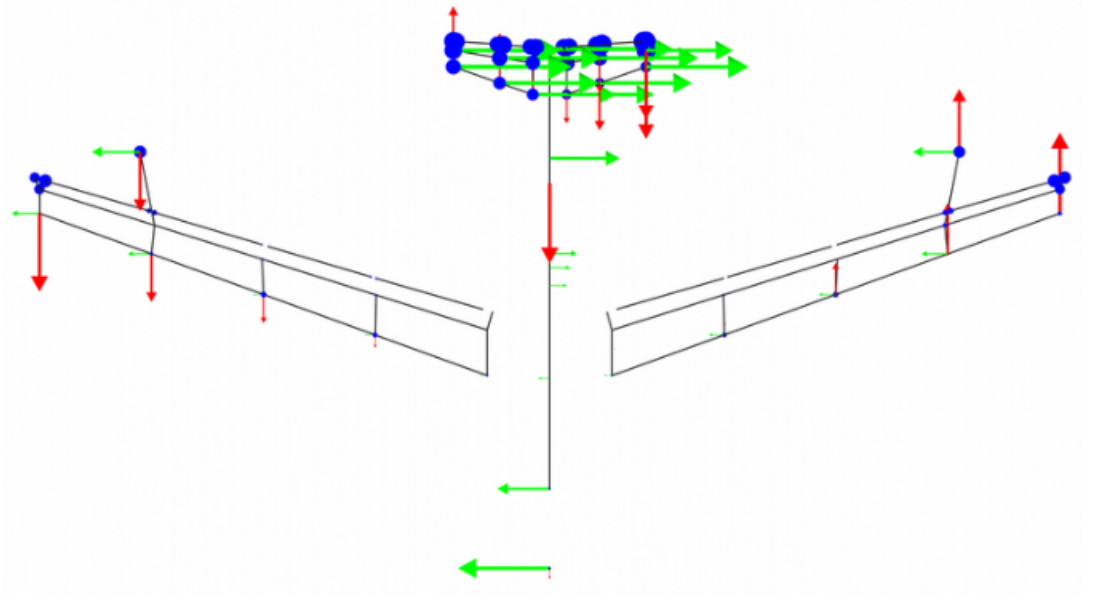


Figure 10: -1 aircraft 1n_wing_in-plane mode

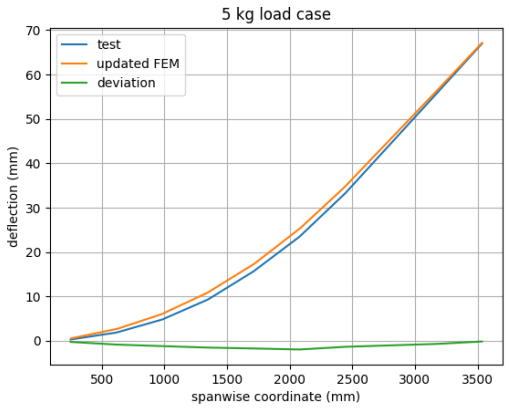


Figure 11: Span-wise displacement of wing under tip load (-1 wing)

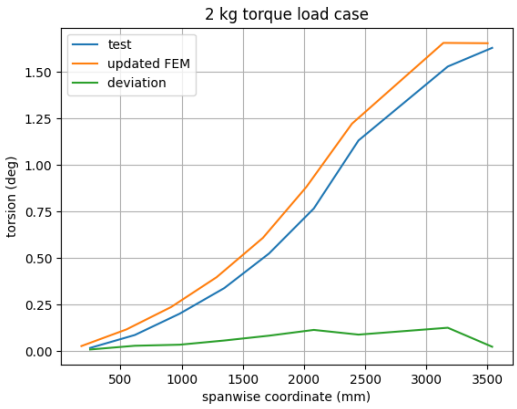


Figure 12: Span-wise torsion of wing under tip load (-1 wing)

After model updating, a modal analysis is conducted with updated model. Figure 13 shows comparison of eigenfrequencies between GVT, FEM and updated FEM. Only the 3n asymmetric wing bending is improved, all other modes become worse. This is due to the fact that updating with the static test, the engineering stiffness (E_1) is tuned down. This results in tuned down eigenfrequencies. Next step is to

use tuning beams locally to improve the 3n bending while not destroying the other mode shapes.

Nr	Mode	GVT	FEM	deviation %	updated FEM	deviation updated %
0	2n wing bending-s	2.94	2.91	-1.19	2.74	-6.91
1	3n wing bending-a	7.57	8.15	7.7	7.66	1.13
2	wing torsion-s	10.27	10.5	2.2	10.43	1.6
3	wing torsion-a	10.73	10.61	-1.15	10.53	-1.86
4	4n wing bending-s	12.13	12.11	-0.19	11.42	-5.83
5	2n wing inplane-s	15.07	15.06	-0.05	14.32	-4.96

Figure 13: Comparison of eigen frequencies of the flexible modes: GVT vs FE model vs updated FE model of the -1 aircraft (*i*n - *i* nodes in the mode, s - symmetric, a - antisymmetric)

3.6 Comparison of RCE aircraft model with static test and GVT

The initial model generated with MDO toolchain was prepared to replicate the static test set up. The results can be seen from figure 14 that the RCE model is way stiffer than the manufactured. Using the same approach as for -1 wing updating, a knock-down factor was applied to the engineering stiffness of wing spar and skin. The figure 15 shows the wing bending results with updated RCE model. The wing bending is much closer to the static test results.

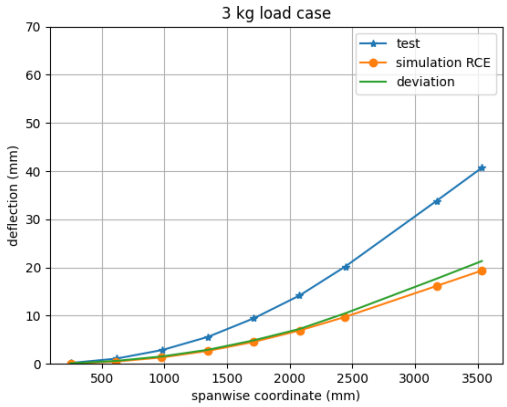


Figure 14: Span-wise bending of wing under tip load (-1 wing RCE model)

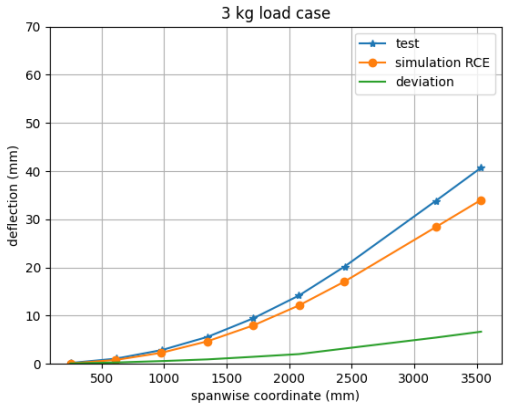


Figure 15: Span-wise bending of wing under tip load (-1 wing RCE updated model)

A modal analysis is also conducted for the initial RCE model and updated RCE model. The results can be seen in the figure 16. After the update, all the modes listed are improved.

Nr	Mode	GVT	RCE FEM	deviation %	updated RCE FEM	deviation updated %
0	2n wing bending-s	2.94	3.81	29.68	2.89	-1.54
1	3n wing bending-a	7.57	10.05	32.82	7.8	2.99
2	wing torsion-s	10.27	12.25	19.3	10.65	3.74
3	wing torsion-a	10.73	12.7	18.33	10.86	1.24
4	4n wing bending-s	12.13	17.38	43.31	13.25	9.25
5	2n wing inplane-s	15.07	17.86	18.54	14.43	-4.27

Figure 16: Comparison of eigen frequencies of the flexible modes: GVT vs RCE FE model vs updated RCE FE model of the -1 aircraft (*i*n - *i* nodes in the mode, s - symmetric, a - antisymmetric)

3.7 Validation of the low order aeroservoelastic (ASE) model

The last step is to validate the accuracy of the low order ASE model that is constructed in the modeling block of the RCE framework. This model serves as the basis for the automatic baseline and flutter suppression control design algorithms. The model is a set of linear time invariant (LTI) models that are obtained form the nonlinear model by Jacobian linearization at airspeed values between 38 and 64 m/s. The model needs to capture the low frequency dynamics of the aircraft for the baseline control design and the flutter modes for the flutter suppression design.

The base model is the low order model of the Flexop aircraft that is described in [23]. The pole map trajectories (as function of the airspeed) of the base model and the RCE generated models are shown in Figure 17.

The plots show good match between the legacy Flexop and the RCE generated model. The pole trajectories show similar trends and the interdependency between them is also very similar between the two modeling frameworks.

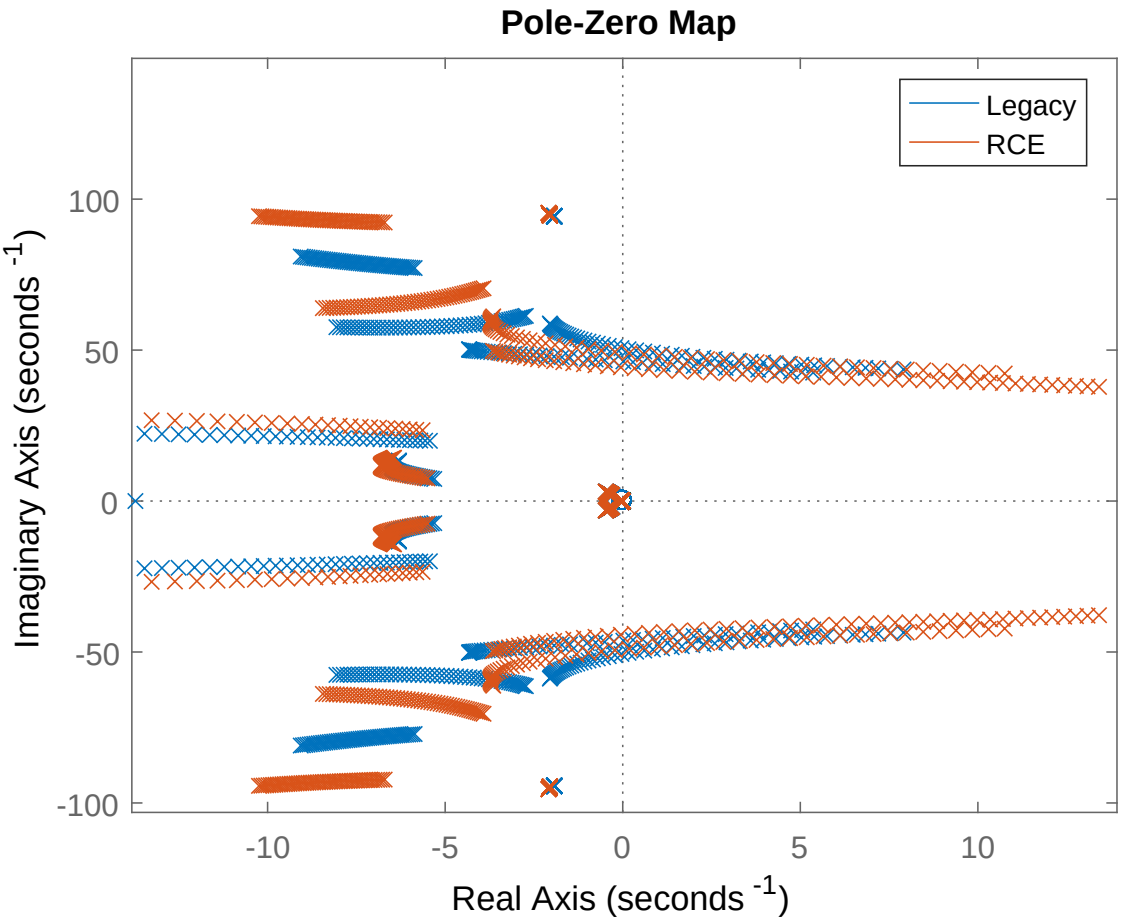


Figure 17: Comparison of pole trajectories of the ASE models: Legacy Flexop model vs RCE generated model of the -1 aircraft

4 Aerodynamics Model Validation

The Deliverable D3.2 – Flight Test Report Phase 1 described the taxi tests, flight tests and aerodynamic analysis performed within years 2020 and 2021. An issue is reported in the deliverable about the actual aircraft producing significantly less lift than was initially modelled. An almost constant lift coefficient offset of around 0.2 can be observed, which results in 35-45 percent lift loss in the 2-4 deg angle of attack region. FT5 and FT7 data do align in the same trend. Accordingly, an investigation was launched not only in the available flight test data, but also in aerodynamic modelling tools. The further findings from the flight test data will be presented in the Deliverable 3.6 – Flight Test Report Phase 2.

In order to select an aerodynamic tool which models the aerodynamic characteristics of T-FLEX correctly, a comparison study of different aerodynamic tools was conducted.

In the mean time, MDO toolchain poses additional requirements on the tools regarding the simulation time and automatic execution.

Investigated tools includes the low order aerodynamic tools, Athena Vortex Lattice (AVL), XFLR5, PyTornado, Tornado, VSPAERO, PAWAT, FlightStream and high fidelity tool STAR-CCM+.

4.1 Aerodynamics modelling tools

This section will give introduction to the investigated tools. For more details please refer to the paper [44].

4.1.1 AVL

AVL is a program for performing aerodynamic analysis of rigid aircraft of arbitrary configurations [8]. It uses the VLM method to model the lifting surfaces. Because of an intrinsic limitation of VLM, AVL is only suitable for inviscid calculation at small angles of attack and sideslip.

4.1.2 Tornado

Tornado is a Vortex Lattice Method for linear aerodynamic wing design applications in conceptual aircraft design or in aeronautical education [25]. The method is built in MATLAB [1] and is based on the description as provided by Moran [27].

The VLM implementation in Tornado ignores the thickness effects of the airfoil, but includes the camber. In Tornado, modelling of control surfaces is possible. Experimental functions that generate a Trefftz-plane analysis can be used. Different options for mesh creation (linear panel distribution, cosine panel distribution, etc.) are available. A graphical interface is available which can plot coefficients of interest, display geometries and mesh.

Initially, Tornado was designed only to include linear aerodynamics. However, the code has been updated to include viscous effects as well [5].

If required, stability derivatives can be calculated using central-difference approximation around the trim condition. It is also possible to calculate trimmed polars.

4.1.3 PyTornado

PyTornado is an aerodynamic tool for conceptual aircraft design. Short computation times make it possible to easily obtain estimates of aerodynamic loads and to benchmark different concepts [11]. Although a similar name as Tornado, PyTornado has been implemented from scratch within the European research project AGILE. A Vortex Lattice Method is implemented in this code. It has a user interface,

pre- and post-processing in Python and a calculation core routine in C++ [26], which guarantees a user friendly interface and computational efficiency. It can be used as a standalone aerodynamic solver or can be integrated into a MDO toolchain. The deformation feature, which is under development, could be potentially used for aeroelastic analysis.

4.1.4 XFLR5

XFLR5 is a software tool designed specifically with model sailplanes in mind [6][7]. Therefore, it focuses on wings operating at low Reynolds numbers. The tool uses XFoil [9] (XFoil v6.99 since XFLR5 v6.55) to calculate the 2D aerodynamics of an airfoil. Non-linear Lifting Line Theory (based on the NACA technical note 1269 [34]), Vortex Lattice Method with quadrilateral rings (as recommended by Katz and Plotkin [14]) or 3D Panel Method (based on Maskew [22]) can be used for 3D wing and tail analysis. Body analysis is not recommended by the author [7].

Unlike the usual VLM solvers, the VLM method implemented in XFLR5 provides a viscous drag correction. In such case, lift-related characteristics (lift distribution, induced drag) are kept inviscid and after local lift distribution is calculated, viscous drag correction using 2D airfoil polars is applied. The lift distribution is not changed. This method is also used during this study. However, the author of the software raises awareness that such correction is not scientifically sound, as using 2D polars ignores any spanwise effects [7].

4.1.5 VSPAERO

VSPAERO [15] is the aerodynamic analysis tool integrated within the conceptual aircraft design package OpenVSP [29]. The tool has two methods available - the Vortex Lattice Method with a simple stall prediction methodology (not used in this study) and a 3D Panel method [16]. Propellers can be included in the simulation. The tool also incorporates the possibility to calculate the parasite drag using the component build-up method. In the current study, only the VLM method is used.

4.1.6 PAWAT

The Preliminary Design Tool for Propeller-Wing Aerodynamics (PAWAT) is an aerodynamic tool for the conceptual design of aircraft [36]. The calculation of the steady state lifting surface aerodynamics in PAWAT is based on a modified three-dimensional nonlinear lifting line theory with a fixed wake model employing nonlinear airfoil data to model nonlinear and viscous effects to a certain extent [36]. PAWAT is also capable of modelling propellers and it allows investigations of the interaction effects between wing and propeller.

The method is built in MATLAB [1]. The description of the lifting line method used is described by Phillips and Snyder [30].

4.1.7 FlightStream

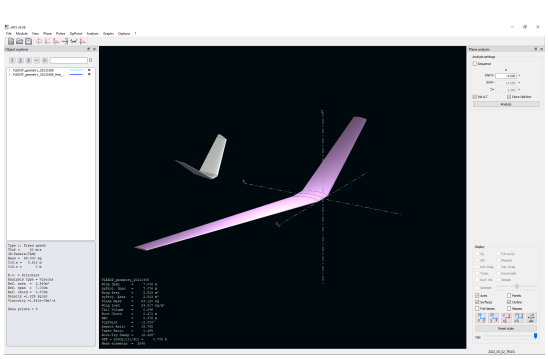
FlightStream is a novel surface vorticity solver capable of using structured or unstructured surface meshes. As a vorticity-based solver, the code can be expected to be substantially more robust and stable compared to pressure-based potential-flow solvers and less sensitive to surface perturbations, and it also allows the use of coarser meshes with an acceptable level of fidelity [28].

To account for viscous effect, integral boundary layer was implemented in FlightStream and was coupled with inviscid solver via displacement of the inviscid boundary equal to the displacement thicknesses of the local boundary layers. More features like prediction of flow separation and stall characteristics are also enabled by this implementation.

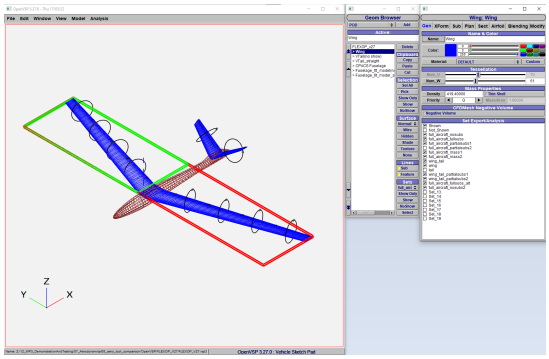
4.1.8 STAR-CCM+

Simcenter STAR-CCM+ is a multiphysics computational fluid dynamics (CFD) software. In this study, it is used to provide the reference data for comparison.

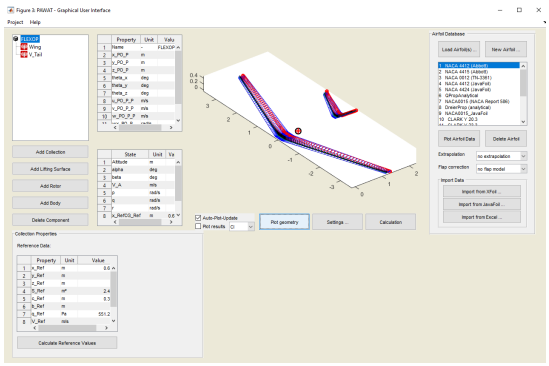
One has to emphasize that most of the tools simulated the wing and tail (Figure 18). Fuselage was included only in simulations of STAR-CCM+ and FlightStream. A study was done with STAR-CCM+ to investigate the influence of the fuselage on the spanwise lift distribution. A small influence was noted at low angles of attack. However, at high angles of attack the fuselage does change the flow at the wing root.



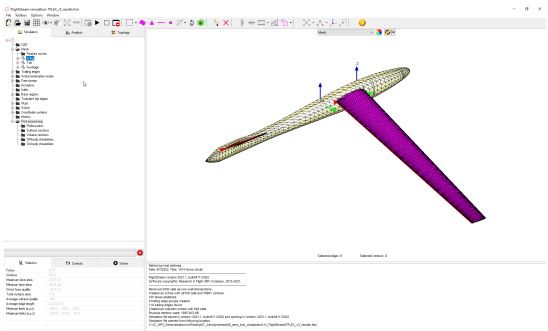
((a)) XFLR5.



((b)) OpenVSP.



((c)) PAWAT.



((d)) FlightStream.

Figure 18: T-FLEX demonstrator modelled in different tools.

4.2 Global aerodynamic coefficients

This section will compare the aerodynamic tools regarding the global aerodynamic coefficients. For more details please refer to the paper [44].

4.2.1 Lift

The lift coefficient data is plotted with respect to the angle of attack in Figure 19 as well as in Figure 20 for the linear part of the slope. The lift curve slope coefficients C_{L_α} and zero angle lift coefficients C_{L_0} are shown in Table 3.

Significant reduction in lift is apparent when comparing the turbulent simulations to Euler simulations. This is expected, as the viscous boundary layer on the top surface of the wing reduces the effective camber line, therefore reducing the aerodynamic angle of attack. Interestingly, most of the tools show better alignment with the turbulent simulations than with the inviscid ones, even though only PAWAT and FlightStream take viscosity into account when calculating lift.

When only the linear part of the lift curve is concerned, the calculated curve slope agreed with each

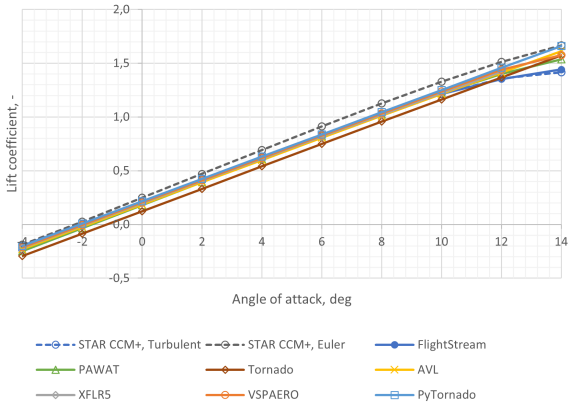


Figure 19: Lift coefficient C_L with respect to the angle of attack α.

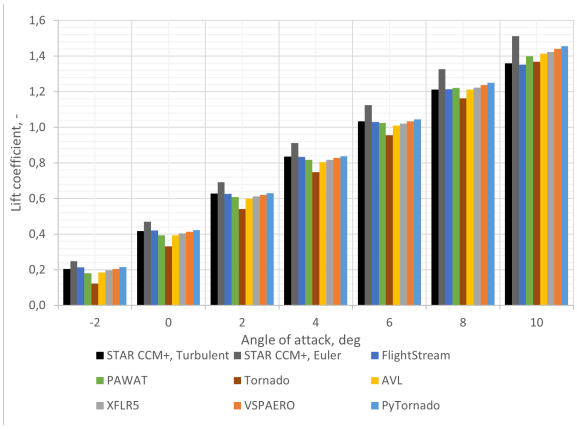


Figure 20: Lift coefficient C_L with respect to the angle of attack α.

Table 3: Comparison of lift curve slope C_{L_α}, zero angle lift coefficient C_{L_0}, minimum drag coefficient $C_{D_{min}}$, pitching moment curve slope C_{m_α} and zero angle pitching moment coefficient C_{m_0} for different aerodynamic modelling tools.

	Turbulent	Euler	FlightStream	PAWAT	Tornado	AVL	XFLR5	VSPAERO	PyTornado
C_{L_α}	0.106	0.111	0.103	0.107	0.105	0.104	0.104	0.104	0.104
C_{L_0}	0.206	0.248	0.214	0.180	0.122	0.185	0.198	0.205	0.215
$C_{D_{min}}$	0.020	0.005	0.015	0.016	0.001	0.002	0.015	0.012	0.002
C_{m_α}	-0.027	-0.028	-0.030	-0.047	-0.050	-0.032	-0.032	-0.026	-0.030
C_{m_0}	0.132	0.141	0.117	0.103	0.193	0.214	0.147	0.128	0.159

other. The zero angle of attack lift shows deviations among tools. Tornado differs most from the other tools. Taking into account that all the tools are meant for preliminary design phase, the differences between them could be categorised as being insignificant.

The nonlinear part of the curve is predicted by both PAWAT and FlightStream. Even at high angle of attack, the lift curve from FlightStream matches quite good with CFD turbulent result. However, as no CFD simulations above 14 degrees were done, the $C_{L_{max}}$ could not be estimated.

The spanwise normalized lift distribution for $\alpha = 2\,\text{deg}$ is plotted in Figure 21. As only the shape of the distribution is of importance here, the local lift coefficients are normalized with respect to the maximum local lift coefficient for the same tool.

The normalized lift distributions between the turbulent and Euler simulations are almost identical. The estimated maximum local lift location is similar for all the tools. The overall shape is very similar with some discrepancies at the root and tip areas. The differences between the STAR-CCM+ results and the other tool results at the wingtip might be due to the poor discretization when extracting the lift distribution from STAR-CCM+.

4.2.2 Drag

Figure 22 shows the inviscid drag polar. While all the VLM tools and panel-based method FlightStream agree mainly, the differences compared with the STAR-CCM+ Euler simulation are noticeable even at

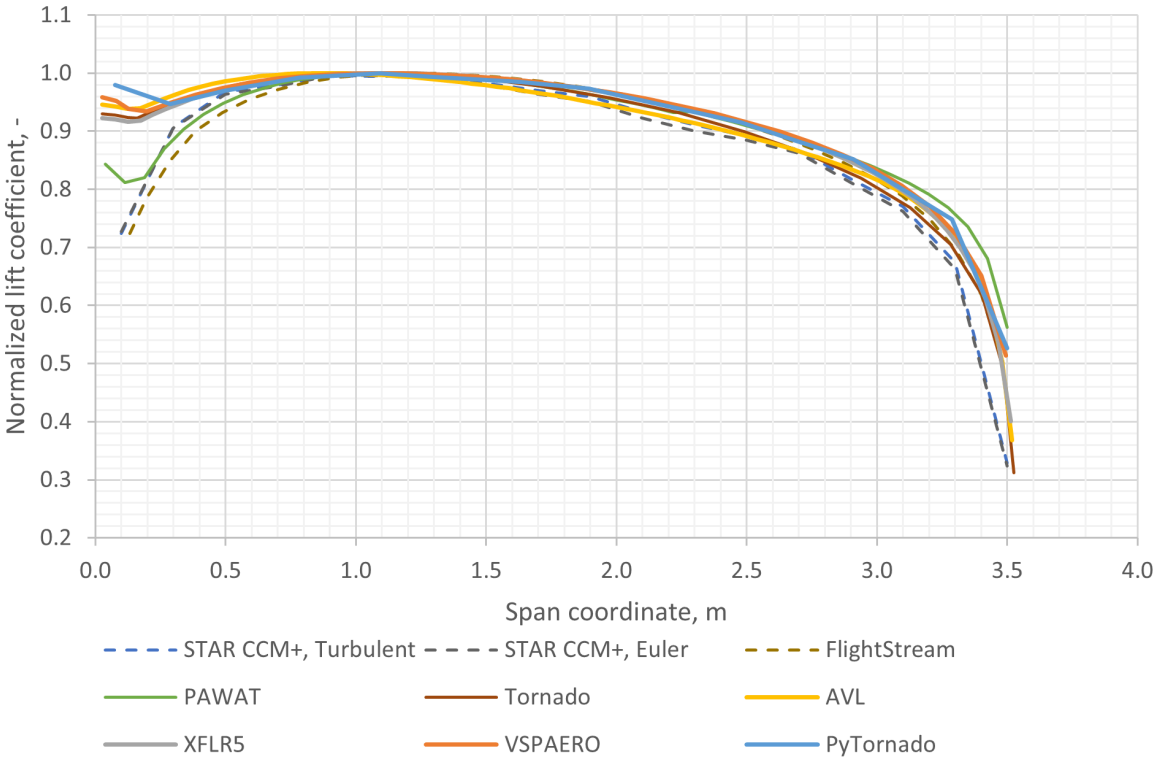


Figure 21: Spanwise normalized lift distribution for $\alpha = 2\,\text{deg}$. The local lift coefficients are normalized with respect to the maximum local lift coefficient of the same tool.

low lift coefficient.

One has to note that the inviscid drag extracted from STAR-CCM+ here is the pressure drag component acting on the aircraft. Strictly speaking, this is not equal to the induced drag by definition. The separation of induced and profile drag from CFD is not straight-forward, and if Euler simulations are used the induced drag due to viscous effects are then ignored. Nowadays there exist some methods to extract these two drag components from CFD [19], but they were not implemented at the time of writing this report.

The total drag coefficient shown in Figure 23 includes both viscous and inviscid drag. Significant differences can be seen in between the tools that correct for viscous drag (STAR-CCM+ (turbulent), FlightStream, PAWAT, XFLR5, VSPAERO) and the ones that do not (Tornado, AVL, PyTornado).

Different methods were used to correct the viscous drag in different software tools. Variation of viscous drag is clearly visible in the Figure 23.

Both PAWAT and XFLR5 correct viscous drag based on 2D airfoil polar data. For XFLR5, 2d viscous drag is interpolated from local wing lift coefficient. The interactive boundary layer, which is a coupling method between potential flow and viscous flow on surfaces, is not implemented in the VLM available in XFLR5 [6]. The consequence of underestimation of viscous drag is confirmed in the Figure 23.

In PAWAT, equations are established for wing segments based on the aerodynamic force derived from three-dimensional vortex lifting law and the aerodynamic force derived from nonlinear airfoil characteristics of the segment and the segment area [36]. Iterative procedure is needed to solve the equations.

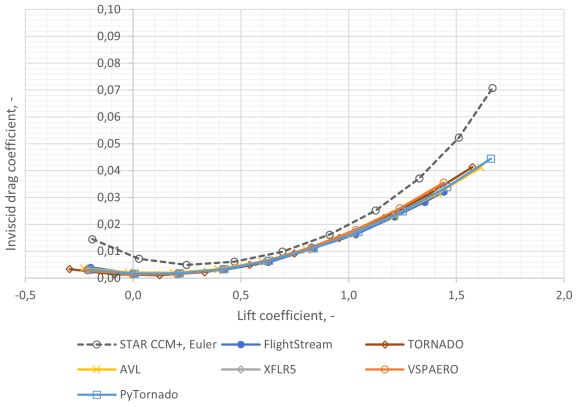


Figure 22: Inviscid drag coefficient C_{D_i} with respect to the angle of attack α.

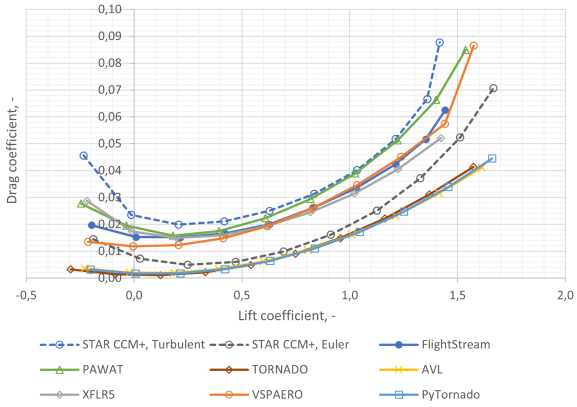


Figure 23: Total drag coefficient C_D with respect to the angle of attack α.

Total drag coefficient from PAWAT matches quite well with CFD data.

In FlightStream, integral boundary layer is coupled with inviscid surface solver to account for viscous drag. Even though, the total drag seems to be underestimated.

4.2.3 Pitching moment

The pitching moment coefficient with regard to angle of attack is shown in Figure 24. The results predicted by STAR-CCM+ Euler simulation and turbulent simulation are almost identical, except that at the high angle of attack, turbulent simulation shows a pitch up trend. FlightStream shows a similar pitch up trend as turbulent simulation, even though an offset of the curve is visible.

The pitching moment coefficients predicted by VSPAERO, XFLR5 and PyTornado match quite good with the CFD results. The results from PAWAT, AVL and Tornado show noticeable deviation to the reference.

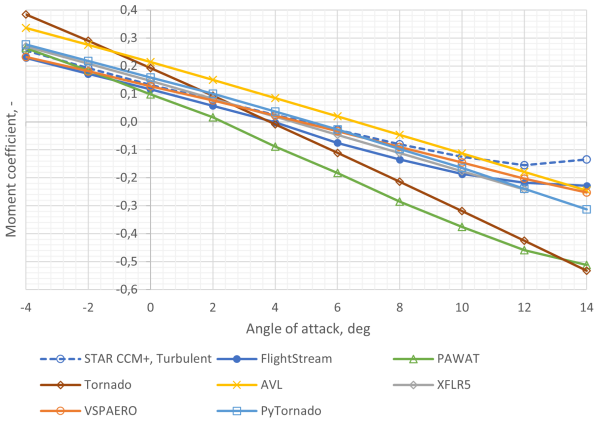


Figure 24: Pitching coefficient C_m with respect to the angle of attack α .

5 Flight Dynamics Model Validation

The flight dynamics of a flexible air vehicle is characterised by an aero(servo)elastic (ASE) model that considers the interaction of aerodynamics, structural dynamics, rigid body dynamics, and control laws which comprise interdisciplinary aircraft modelling. These subsystems can be independently modelled using a theoretical approach, and experimental results from various ground and flight tests can be incorporated into the models. Before flight tests can be conducted safely and effectively, a number of ground tests must be performed. The static and ground vibration tests (GVT) are essential for evaluating and improving the accuracy of the numerical models used during the aircraft design phase. The results of ground testing, for instance, are used to update and validate the Finite Element (FE) model that represents the structural dynamic part of the aircraft model. Similarly, an aerodynamic model can be generated using computational fluid dynamics (CFD) simulations or panel methods derived from potential theory, typically representing quasi-steady and unsteady aerodynamics, which can be partially updated using limited data from wind tunnel-tests, which are often used in the aircraft design and development phase for configuration optimization. Compared to completely flight mechanical models representing rigid body aircraft dynamics, which are adequately represented by low-order dynamics, and purely structural models, which are adequately represented by higher-order dynamics, ASE models encompass both the low- and high-frequency range.

For the validation of the flight dynamics model, an updating methodology for correcting of the numerical system matrices A, B, C and D of the linear discrete-time state-space models of the flexible aircraft has been developed combining the phenomenological and behavioral model structures. The proposed approach updates the system matrices using the measured input-output data from flight test and the initial state-space model of the flexible aircraft which is derived from the linearization of the nonlinear first-order differential equations describing the aircraft motion. The key feature of the proposed approach is that updating can be executed in a single step with multiple data bases from different flight tests with nearly-identical initial conditions, resulting in a more physically realistic correction of the system matrices. The updating method addresses linear estimation problems which allows an manageable implementation with fast execution avoiding optimization problems for approximation of solutions of nonlinear differential equations resulting from aircraft equations of motion.
The primary purpose of numerical model improvement is to identify and correct (update) the discrepancies between experimental and predicted numerical outputs. In the case of significant disparities between model predictions and experimental data from flight tests, the numerical aircraft model must be updated until there is a satisfactory correlation between model predictions and experimental results.

The proposed updating algorithm and its advanced application is based on the study described in [38]. The formulation of the proposed updating approach enables correction of the system matrices A, B, C, and D of the initial linearized discrete-time (DT) state-space system derived from a flexible aircraft model. The updating method addresses linear estimation problems combining the phenomenological and behavioral model assumptions.

Thus, two formulations for the error minimization, i.e. minimization of the output residual between flight measured data and model predictions have been defined. The first utilises the state-space system's output equations, whereas the second requires both the state and output equations. The methodology for updating that will be described here is based on a linear least-squares approximation resp. a minimum norm solution. The updating algorithm has three steps. In the first step, the calculated states from the initial model corresponding to the rigid body aircraft dynamics will be corrected using output equations. Here, the considered states are measured and comprise a subset of the outputs. In the second step, we use the same approach as in the first. Here, the principal difference is that we consider the states corresponding to rigid body and flexible aircraft dynamics. In the third step of the algorithm, the system matrices A, B, C, and D are directly updated using the updated states from the previous two steps. Note that the algorithm described in [38] consists of four steps. Here, we omit the fourth

step in which the system matrices C and D are reestimated to ensure a better output match without regard to the actual system's internal behaviour. Within the context of the method and model validation analysis used for this study, we evaluate the quality of the updated model by residual analysis using Theil's inequality for assessment of the fit. Theil's inequality and the breakdown of fit error in terms of bias, variance and covariance proportions, offer insight on the validity of the predicted responses from the updated model. The following diagram (Fig.25) illustrates a summary of the study presented in this work, including the flight test domain, model structure, updating algorithm, and model validation process.

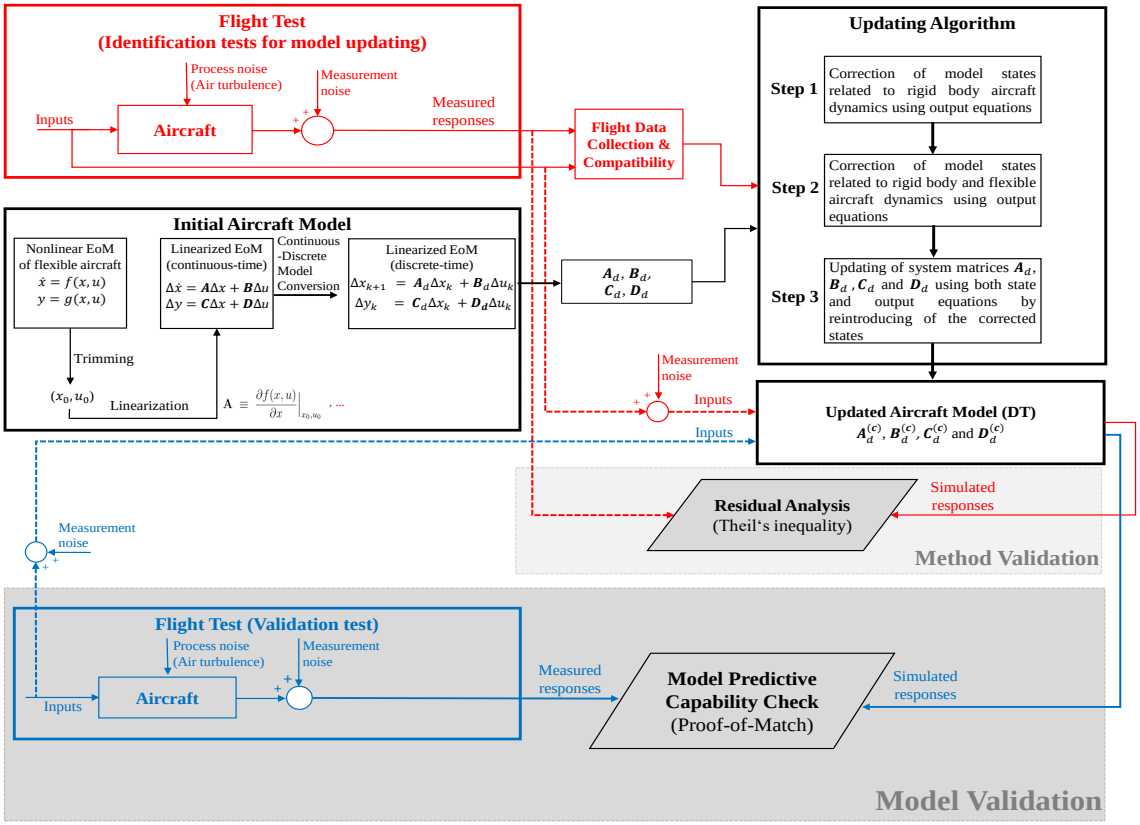


Figure 25: Overview of the model structure, updating algorithm and validation process [38]

5.1 Updating algorithm

This section outlines the formulation of the proposed updating approach for the numerical system matrices $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$, and $\mathbf{D}_d$ of the discrete-time linear state-space system representing the flexible aircraft model. The main objective of numerical model improvement is to detect and correct the discrepancies between experimental and estimated system outputs. The updating methodology is based on error minimization of the output residuals using both state and output equations. Its algorithm is a three-step procedure and it is based on a linear least-squares approximation resp. a minimum norm solution. We begin by defining the error minimization formulations used in the updating approach, and then we outline the related updating steps.

The first error minimization problem which represents an output residual formulation between flight test data and model predictions by using only the output equations, is defined as:

$$\min_{\Delta x_{k,i}} \sum_{i=1}^{N_{test}} \sum_{k=1}^{N_t-1} \|\overline{y}_{k,i} - \mathbf{S}_{sens} \cdot (\mathbf{C}_{\mathrm{d}} \Delta x_{k,i} + \mathbf{D}_{\mathrm{d}} (\overline{u}_{k,i} - u_{0,i}) + y_{0,i})\|^2 \,. \tag{1}$$

We shall denote the value of perturbed state vector $\Delta x_{k,i}$ that minimizes Eq. (1) by $\Delta x_{k,i}^{(c)}$, for fixed $\mathbf{C}_d$ and $\mathbf{D}_d$:

$$\Delta x_{k,i}^{(c)} \;=\; \arg\min_{\Delta x_{k,i}} \sum_{i=1}^{N_{\text{test}}} \sum_{k=1}^{N_t-1} \|\overline{y}_{k,i} - \mathbf{S}_{sens} \cdot (\mathbf{C}_d \Delta x_{k,i} + \mathbf{D}_d (\overline{u}_{k,i} - u_{0,i}) + y_{0,i})\|^2 \,, \tag{2}$$

where $\overline{u}_{k,i} \in \mathbb{R}^m$ and $\overline{y}_{k,i} \in \mathbb{R}^{\overline{l}}$ are measured inputs and outputs from ith flight test. N_{test} denotes the number of flight test sets used for model updating. For clarification: $\overline{l}$ is the number of outputs from test and l is the number of outputs from numerical model. $\mathbf{S}_{sens} \in \mathbb{R}^{\overline{l} \times l}$ is the sensor matrix allocating the measured outputs with the estimated outputs from model. It is defined as an identity matrix $\mathbf{S}_{sens} = \mathbf{I} \in \mathbb{R}^{l \times l}$ if all outputs from the numerical model are measured. Otherwise it becomes a rectangular matrix $\overline{l} < l$. For most cases, a sensor matrix is needed when you are interested in outputs that are essential but cannot be measured during the test, or when only a subset of measured quantities is intended for model updating. The error minimization formulation given in 1 is used for the first and second step of the updating algorithm, where for fixed $\mathbf{C}_d$ and $\mathbf{D}_d$ the perturbed states Δx_k can be corrected.

The second error minimization formulation requires both state and output equations where system matrices $\mathbf{A}_d$, $\mathbf{B}_{\mathrm{d}}$, $\mathbf{C}_{\mathrm{d}}$ and $\mathbf{D}_{\mathrm{d}}$ can be corrected by using the updated states $\Delta x_{k,i}^{(c)}$ obtained from the Eq. (2):

$$\begin{aligned} \mathbf{A}_d^{(c)}, \mathbf{B}_{\mathrm{d}}^{(c)}, \mathbf{C}_d^{(c)}, \mathbf{D}_d^{(c)} \;=\; \arg \min_{\left(\begin{smallmatrix} A_d & B_d \\ C_d & D_d \end{smallmatrix}\right)} \sum_{i=1}^{N_{test}} \sum_{k=0}^{N_t-2} \left\| \begin{pmatrix} \Delta x_{k+1,i}^{(c)} \\ \overline{y}_{k,i} \end{pmatrix} - \right. \\ \left. \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{S}_{sens} \end{bmatrix} \left(\begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d \\ \mathbf{C}_d & \mathbf{D}_d \end{bmatrix} \begin{pmatrix} \Delta x_{k,i}^{(c)} \\ \overline{u}_{k,i} - u_{0,i} \end{pmatrix} + \begin{pmatrix} 0 \\ y_{0,i} \end{pmatrix} \right) \right\|^2 . \end{aligned} \tag{3}$$

Here, the error minimization formulation in Eq. (3) is used in the third step of the updating algorithm. The three-step procedure of the proposed updating method may be summarised as follows:

1. Correction of measured states corresponding to rigid body aircraft dynamics using the first error minimization formulation given in Eq. (2)

2. Correction of measured and unmeasured states relating to rigid body and flexible aircraft dynamics using the first formulation for error minimization given in Eq. (2)

3. Correction of system matrices $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$ and $\mathbf{D}_d$ using the updated states from the first and second step using the second formulation for error minimization given in Eq. (3)

Detailed mathematical derivation of the updating methodology can be found in [46].

5.2 Model Validation

Facts are distinct from estimates. Model validation is essential for gaining confidence in or rejecting a certain model. Comparing measured and simulated outputs is required to validate the updated model. There are numerous aspects of model validation that can be broadly categorised into three subcategories [12]:

- Statistical properties of the estimates
- Residual analysis
- Model predictive quality

These three methods offer insight into the effectiveness or ineffectiveness of model parameters. They provide the essential means to evaluate the suitability of identified (updated) models and their parameters in duplicating the system closely enough.

Within the framework of the method and model validation analysis employed in this study, we evaluate the quality of the updated model by residual analysis using Theil's inequality for assessment of the fit. The prediction capability of the updated model, which will be described later in this section, is then evaluated.

5.3 Theil's inequality analysis and decomposition of fit error

Theil's inequality provides an usefull type of output statistics for the overall fit. Theil's inequality coefficient TIC is defined as

$$\mathrm{TIC_o} = \frac{\sqrt{\frac{1}{N_t}\sum_{k=0}^{N_t-1}\left(\overline{y}_{\mathrm{k,o}} - y_{\mathrm{k,o}}\right)^2}}{\sqrt{\frac{1}{N_t}\sum_{k=0}^{N_t-1}\left(\overline{y}_{\mathrm{k,o}}\right)^2} + \sqrt{\frac{1}{N_t}\sum_{k=0}^{N_t-1}\left(y_{\mathrm{k,o}}\right)^2}}, \quad o = 1, 2, ..., \bar{l} \tag{4}$$

where $y_{\mathrm{k,o}}$ is the o'th output from model at time $t = kT_s$ and $\overline{y}_{\mathrm{k,o}}$ is its counterpart from flight test. Theil's inequality coefficient measures the conformity between two time series. In statistical terms, it is the ratio of the root-mean-square fit error to the sum of the root-mean-square values of the measured and estimated signals. TIC $= 0$ (case of equality) implies a perfect match, whereas TIC $= 1$ indicates the case of maximal inequality.
Additionally, Theil analysed the fit error between the two time series in terms of bias, variance, and covariance proportions given by [21], [41]:

$$\mathrm{TIC_{M,o}} = \frac{\left(\overline{y}^M_{\mathrm{k,o}} - y^M_{\mathrm{k,o}}\right)^2}{\frac{1}{N_t}\sum_{k=0}^{N_t-1}\left(\overline{y}_{\mathrm{k,o}} - y_{\mathrm{k,o}}\right)^2} \tag{5}$$

$$\mathrm{TIC_{S,o}} = \frac{\left(\sigma_{\overline{y},o} - \sigma_{y,o}\right)^2}{\frac{1}{N_t}\sum_{k=0}^{N_t-1}\left(\overline{y}_{\mathrm{k,o}} - y_{\mathrm{k,o}}\right)^2} \tag{6}$$

$$\mathrm{TIC}_{\mathrm{C,o}} = \frac{2\left(1-\rho_{\mathrm{o}}\right)\sigma_{\overline{y},o}\sigma_{y,o}}{\frac{1}{N_{\mathrm{t}}}\sum_{k=0}^{N_{\mathrm{t}}-1}\left(\overline{y}_{\mathrm{k,o}} - y_{\mathrm{k,o}}\right)^2} \tag{7}$$

wehre $\overline{y}^M_{\mathrm{k,o}}$ and $y^M_{\mathrm{k,o}}$ refer to the mean values of the oth measured and simulated output. σ and ρ are the standard deviations and correlation coefficient respectively of the two output signals $\overline{y}$ and y. They are defined as

$$\sigma_{\overline{y},o} = \sqrt{\frac{1}{N_{\mathrm{t}}}\sum_{k=0}^{N_{\mathrm{t}}-1}\left(\overline{y}_{\mathrm{k,o}} - \overline{y}^M_{\mathrm{k,o}}\right)^2} \quad , \quad \sigma_{y,o} = \sqrt{\frac{1}{N_{\mathrm{t}}}\sum_{k=0}^{N_{\mathrm{t}}-1}\left(y_{\mathrm{k,o}} - y^M_{\mathrm{k,o}}\right)^2} \tag{8}$$

$$\rho_{\mathrm{o}} = \frac{1}{\sigma_{\overline{y},o}\sigma_{y,o}}\frac{1}{N_{\mathrm{t}}}\sum_{k=0}^{N_{\mathrm{t}}-1}\left(\overline{y}_{\mathrm{k,o}} - \overline{y}^M_{\mathrm{k,o}}\right)\left(y_{\mathrm{k,o}} - y^M_{\mathrm{k,o}}\right) . \tag{9}$$

Again, we separate the three proportions defined in Eqs. 5-7 for each output.

The bias proportion $\mathrm{TIC}_{\mathrm{M,o}}$ is a measure of the systematic error in the updated model, while variance proportion $\mathrm{TIC}_{\mathrm{S,o}}$ indicates the model's capacity to duplicate the variability of the true system. Nonsystematic error is quantified using the covariance proportion $\mathrm{TIC}_{\mathrm{C,o}}$. The above breakdown offers insight into the sources of fit error. For an ideal case, the bias and variance proportions should be close to zero, and the covariance proportion should be close 1. The sum of these three proportions equals 1. For both $\mathrm{TIC}_{\mathrm{M,o}}$ and $\mathrm{TIC}_{\mathrm{S,o}}$, a large value, often greater than 0.1, would be cause for concern and the updated (identified) model should be scrutinised and analysed in detail. In conjunction with a visual evaluation of the fit between the output signals, these criteria give slightly more insight into the characteristics of residuals [46].

5.3.1 Model predictive capability

The predictive capability check of the updated model is determined by comparing the flight measured responses with those estimated by the updated model for the "same" control inputs. This requires a flight test data set with the nearly identical control inputs and trim conditions as chosen in identification tests for model updating. In the terminology of aircraft applications, this procedure is frequently referred to proof-of-match (POM), which is not an easy task. Both the control input and output measurements are susceptible to measurement noise. In addition, even when the proof-of-match manoeuvres can be performed in apparently calm atmospheric conditions, the aircraft is excited by a small amount of non-measurable turbulence-induced excitation. In general, complementary flight data, i.e. flight manoeuvres not used in the identification tests for model updating, are used to evaluate the model predictive capability. Validation on complementary data is sometimes referred to informally as a "acid test" [13].

5.4 Case Study

The algoritm of the proposed updating method is coded in MATLAB. With both the discrete state-space model and access to flight test data of the FLiPASED demonstrator aircraft, the application of the proposed updating methodology is demonstrated, followed by the presentation of convincing findings.

5.5 Flight Test

The FLiPASED aircraft is a jet-powered UAV with a wing span of 7.1 m and a takeoff mass of 65 kg. It was manually operated within visual line of sight. Ref.[43] offers more insight into the flight test campaign. The FLiPASED aircraft is equipped with an integrated measurement system that is considered in both the nonlinear aircraft model and the linearized state-space representation, respectively. The usual air data, position and inertial parameters are being logged on the aircraft. Attached to the front and rear spars are 12 inertial measurement units (IMU) that records the structural deflections of the wings. The wing-mounted IMUs measure translational accelerations in z direction and the angular rates ω_x and ω_y. As outputs, 36 time histories are therefore provided. Figure 26 shows the configuration of the IMUs' placement on the wings [37]. Further, a fuselage-mounted IMU provides flight measured translational accelerations a_{Fuse} and the rotational rates Ω_{Fuse}. More information on flight test instrumentation (FTI) is given in Ref. [43].

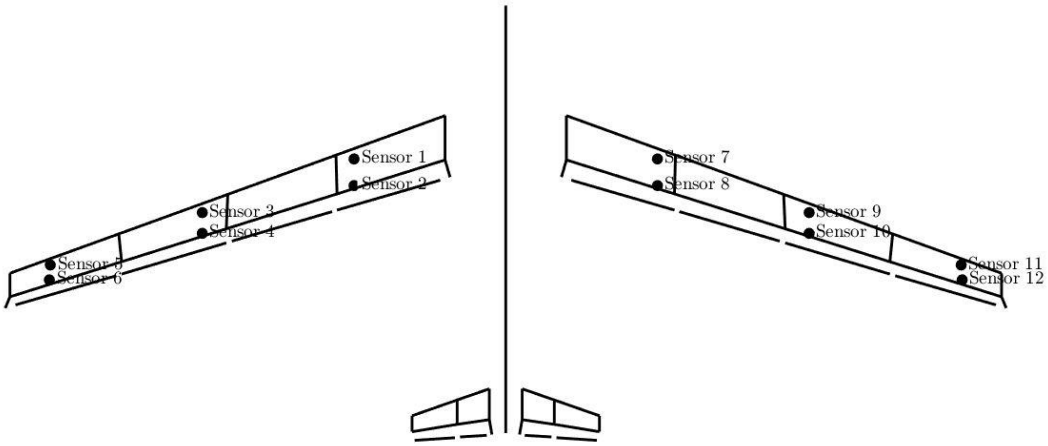


Figure 26: Location of accelerometers (IMUs) on FLiPASED aircraft [38]

The flight test data used in this study are provided by a pushover-pull-up manoeuvers. The primary objective of pushover–pull-ups, commonly known as roller-coaster, is to identify lift and drag character-

istics, longitudinal stability, and elevator trim requirements. The maneuver starts from a trimmed level flight condition with a constant thrust [12]. With a sampling rate of 200 Hz and a 20-second time window, experimental data from three test sets have been used for model updarting method. The recorded input/output time series are then upsampled to 1 kHz to obtain data consistency with the discrete state-space model of the aircraft. The flight measured outputs with their physical quantities are listed in the table 4. In addition, the trim conditions obtained from flight test measurements are stated in table below (Tab.5).

Table 4: Physical quantities of the states and inputs-outputs of the linear discrete-time state-space system of the aircraft model*

States				Inputs	Outputs
kinematic	dynamic	elastic	aerodyn. lag		
$\Delta\phi$	Δu	$\Delta\eta_f$ (30)	Δx_L (288)	$\Delta\xi$ aileron defl. (8)	ϕ
$\Delta\theta$	Δv	$\Delta\dot{\eta}_f$ (30)		$\Delta\eta$ elevator defl.(4)	θ
$\Delta\psi$	Δw			$\Delta\delta_F$ Thrust setting	ψ
Δx_{0E}	Δp				$a_{x,\ \text{IMU-Fuse}}$
Δy_{0E}	Δq				$a_{y,\ \text{IMU-Fuse}}$
Δz_{0E}	Δr				$a_{z,\ \text{IMU-Fuse}}$
					$p_{\ \text{IMU-Fuse}}$
					$q_{\ \text{IMU-Fuse}}$
					$r_{\ \text{IMU-Fuse}}$
					z_E
					V_{IAS}
					α
					β
					h_{baro}
					p_{stat}
					p_{total}
					IMUs-Wing: a_z, ω_x, ω_y (36)

* The numbers in parentheses denote the number of corresponding physical quantity.

Table 5: Trim values measured from identification flight tests

	$\eta_{elev,0}$ [deg]	θ_0 [deg]	$V_{ias,0}$	$h_{baro,0}$ [m]	α_0 [deg]	β_0 [deg]
Identification Test #1	-3.68	0.99	38.5	814	2.78	0.403
Identification Test #2	-3.52	8.12	35.7	807	3.73	-0.818
Identification Test #3	-3.52	1.79	39.7	738	1.67	1.424

As is clear from the table above (5), the initial conditions assessed by three sets of flight data are not "equal" as expected under real conditions. Here, we limit ourselves to the barometric altitude $h_{baro,0}$ and $V_{ias,0}$ parameters from which, inter alia, the generated state-space models are dependent.

For pushover-pull-up manoeuvres the aircraft is excited by elevator deflections, as depicted in Fig.27.

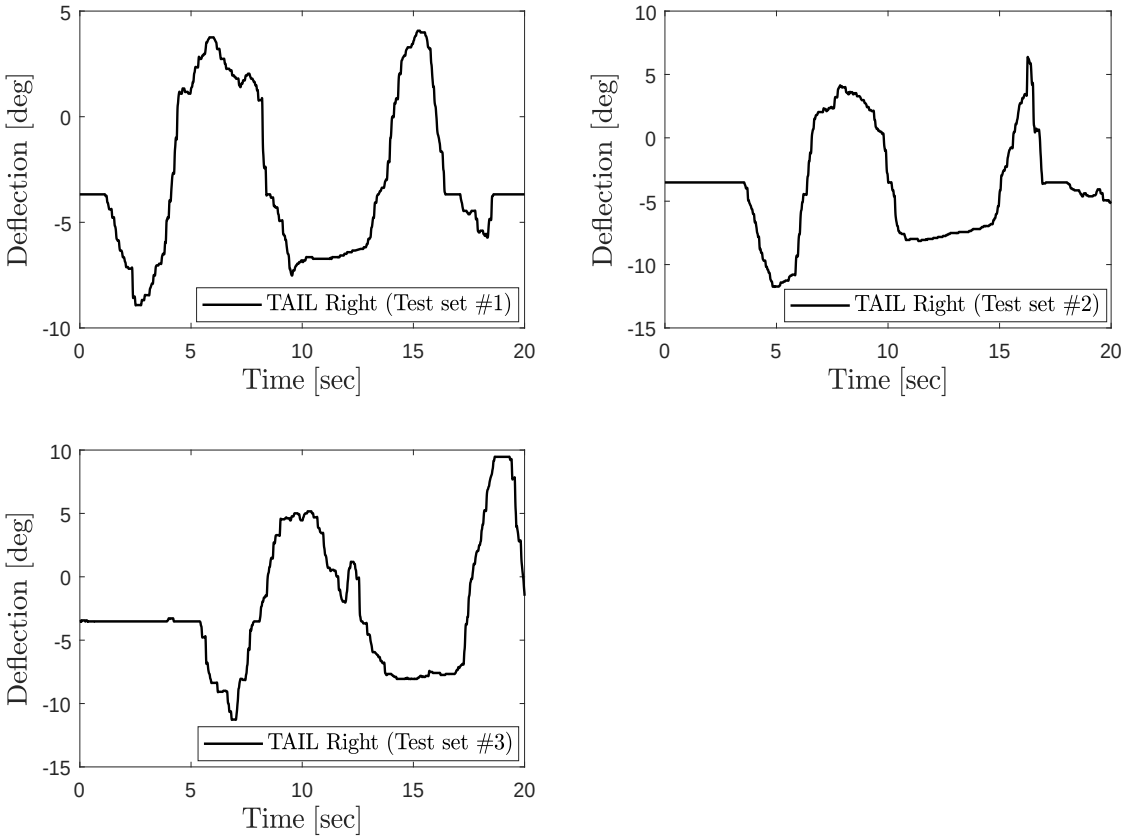


Figure 27: Elevator deflections used for pushover–pull-ups

5.6 Results

The presented updating method has been successfully applied on the linearized FLiPASED aircraft model including the use of the flight test data. The results are based on the research described in [46].

Using residual analysis by means of Theil's inequality formulation given in 4, we assess the quality of the updated model for each of the output variables $o = 1, 2, ..., \bar{l}$. Although the acceptable value for TIC_o varies on the application, as a general guideline, a value < 0.25 indicates a satisfactory agreement. It is also feasible to establish a single measure for the overall fit. Thus, we define the mean Theil's inequality coefficient TIC_{mean} given by

$$\text{TIC}_{\text{mean}} = \frac{1}{\bar{l}} \cdot \sum_{o=1}^{\bar{l}} \text{TIC}_o \quad \% \tag{10}$$

Figure 28 illustrates the correlation results calculated from Theil's inequality formulation between the considered subset of output signals from three flight test sets and corresponding outputs from the initial and updated model.

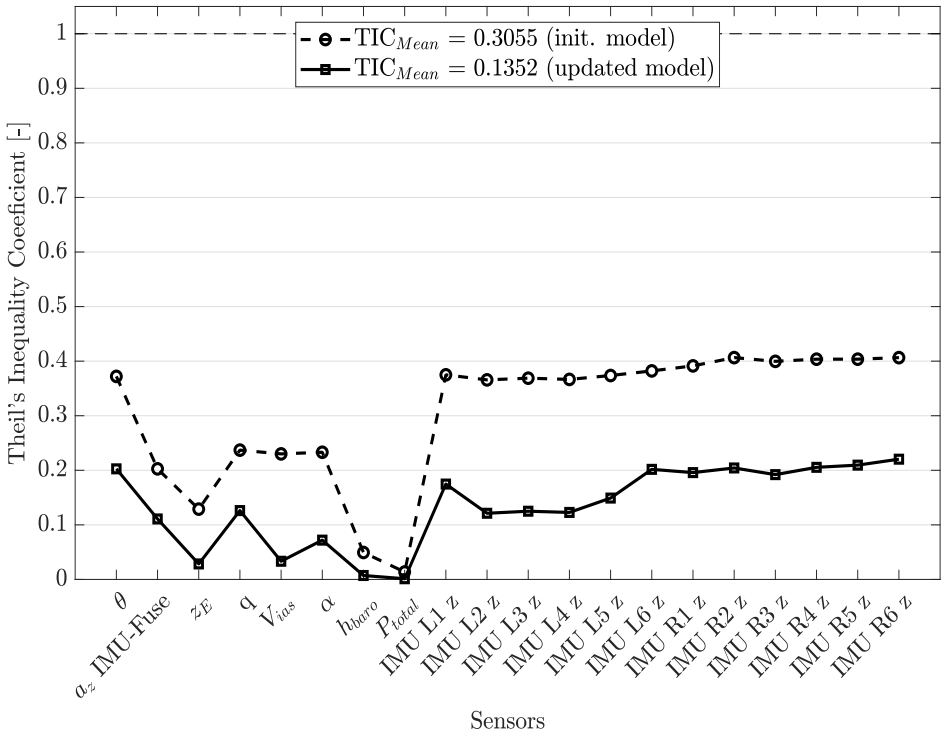


Figure 28: Fit error distribution between flight test and updated model data for each output (Number of test sets = 3)

As is evident from the plot in Fig. 28, a high degree of fit between flight test data and results from updated linearized aircraft model has been achieved. The mean Theil's inequality coefficient TIC_{mean} (Eq.10) between the outputs from updated model and the recorded data from flight test is approximately 0.14 and has decreased by 0.17. Again, $\text{TIC} = 0$ implies a perfect match (best fit), whereas $\text{TIC} = 1$ indicates the case of minimum correlation. In addition, we partitioned the fit error TIC_o for each output

between the flight measured and reconstructed responses from the updated model into proportions of bias, variance, and covariance given in the equations 5-7 as shown in Fig. 29.

As already mentioned in Section 5.2, the bias and variance proportions TIC_M and TIC_S should be very small, typically $(TIC_M + TIC_S) \leq 0.2$. Again, TIC_M is an indicator of the systematic error in the updated model, while variance proportion TIC_S implies the model's capability to duplicate the variability of the physical system [12]. The diagram in Fig. 29 clearly demonstrates that nonsystematic error, represented by the covariance proportion TIC_C, predominates the source of the fit error with $TIC_{C,o} > 0.90$, $\forall o = 1, 2, ..., \bar{l}$, which suggests a high quality of the updated model, that is, capacity to duplicate the true system response.

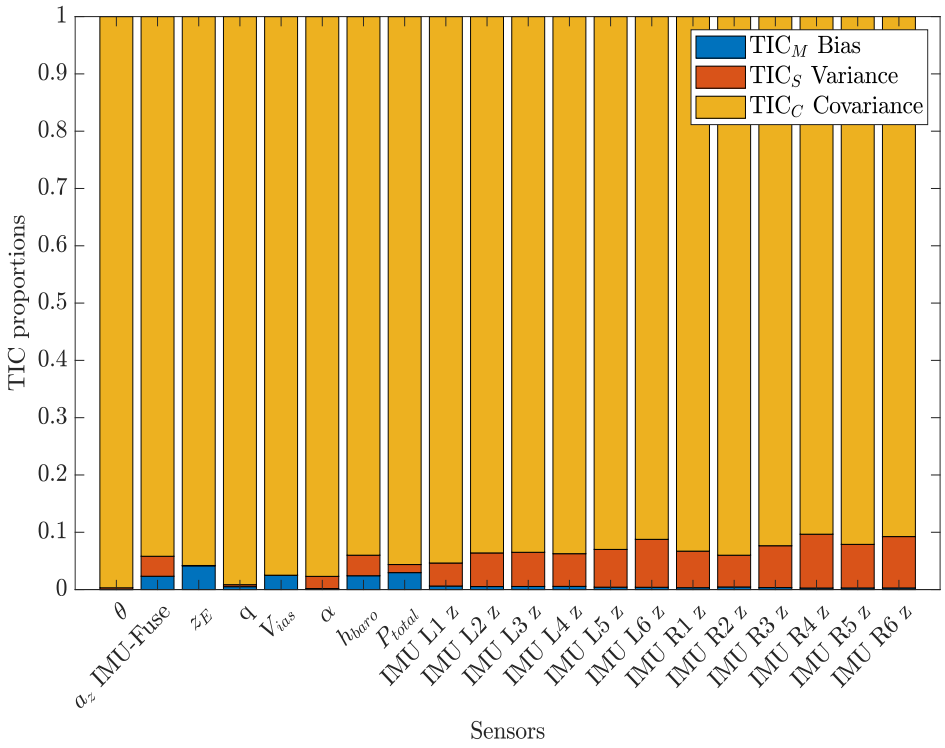


Figure 29: Breakdown of the fit error into proportions of bias, variance, and covariance (Number of test sets = 3)

Table (6) provides a detailed summary of the results obtained from Theil's inequality formulation including the decomposition of fit error between flight measured and reconstructed outputs from the updated model.

Table 6: Correlation results including the decompositon of fit error between flight measured outputs from identification tests and reconstructed outputs from updated model

	TIC [-]	TIC_M [-]	TIC_S [-]	TIC_C [-]
θ	0.20267	4e-05	0.003	0.99695
$az_{\text{IMU}-\text{Fuse}}$	0.11085	0.02322	0.03488	0.9419
z_{E}	0.02841	0.04142	0.00059	0.95799
q	0.12621	0.00506	0.00346	0.99149
V_{ias}	0.03331	0.0249	0.00016	0.97494
α	0.07202	0.00177	0.02124	0.97699
h_{baro}	0.00721	0.02412	0.03584	0.94003
P_{total}	0.00128	0.02959	0.01411	0.9563
IMU L1 z	0.17478	0.00614	0.04021	0.95365
IMU L2 z	0.12126	0.00526	0.05868	0.93606
IMU L3 z	0.12501	0.00539	0.05964	0.93497
IMU L4 z	0.1228	0.00553	0.05709	0.93738
IMU L5 z	0.1491	0.00417	0.06588	0.92995
IMU L6 z	0.20174	0.00398	0.08365	0.91237
IMU R1 z	0.19578	0.00325	0.06382	0.93293
IMU R2 z	0.20428	0.00452	0.05543	0.94005
IMU R3 z	0.19206	0.00347	0.07302	0.92351
IMU R4 z	0.20545	0.0027	0.09392	0.90338
IMU R5 z	0.20948	0.0028	0.07607	0.92113
IMU R6 z	0.22037	0.00282	0.08966	0.90752

In the following, a few selected outputs from the flight test are plotted together with the outputs from initial and updated linearized model of the aircraft (Fig. 30 - 36). It is clearly evident that the presented updating method enables to reconstruct all the measured responses obtained from the flight test with high accuracy even in case of highly noise-contaminated experimental data.

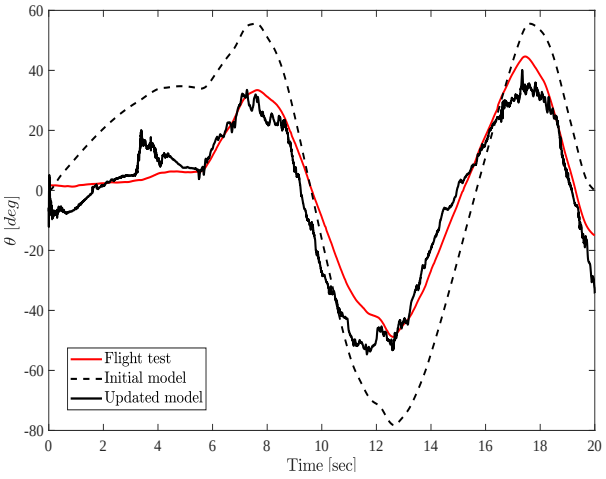


Figure 30: Pitch angle θ

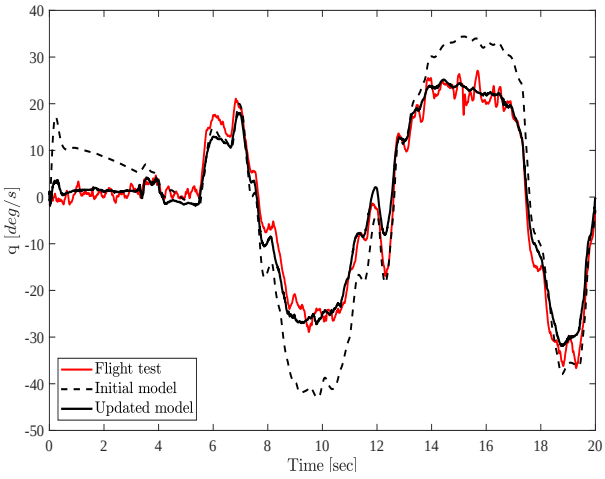


Figure 31: Pitch rate ($q_{\text{IMU-Fuse}}$)

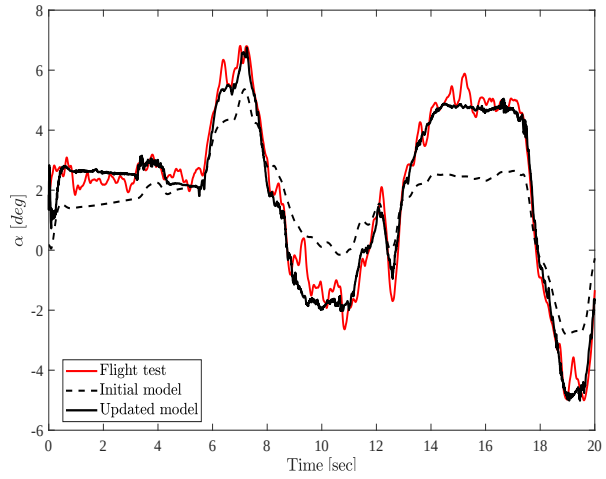


Figure 32: Angle of attack α

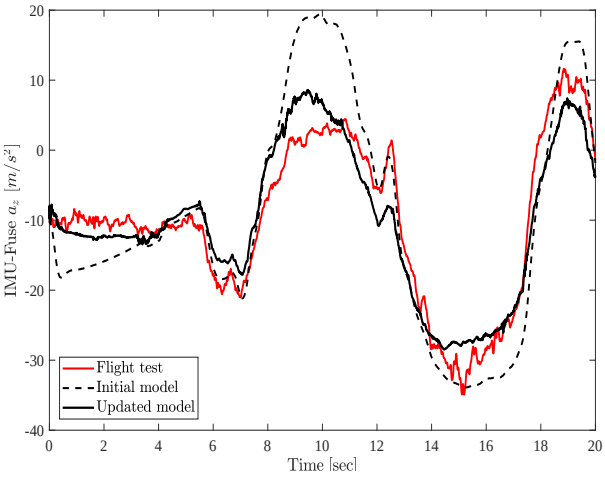


Figure 33: Vertical acc. $a_{z,\ \text{IMU-Fuse}}$

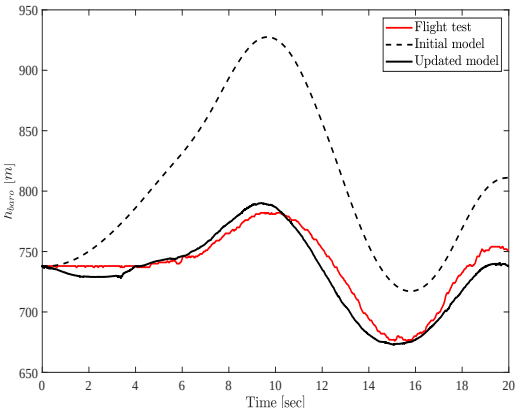


Figure 34: Barometric altitude h_{baro}

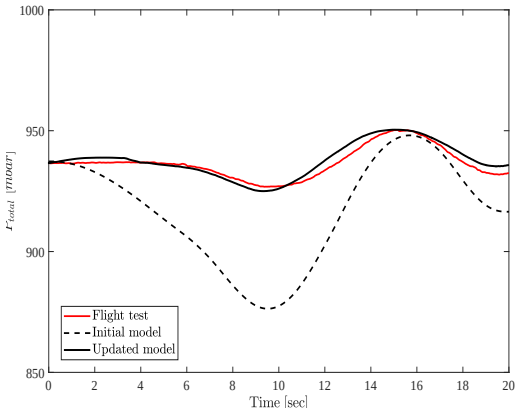


Figure 35: Total pressure P_{total}

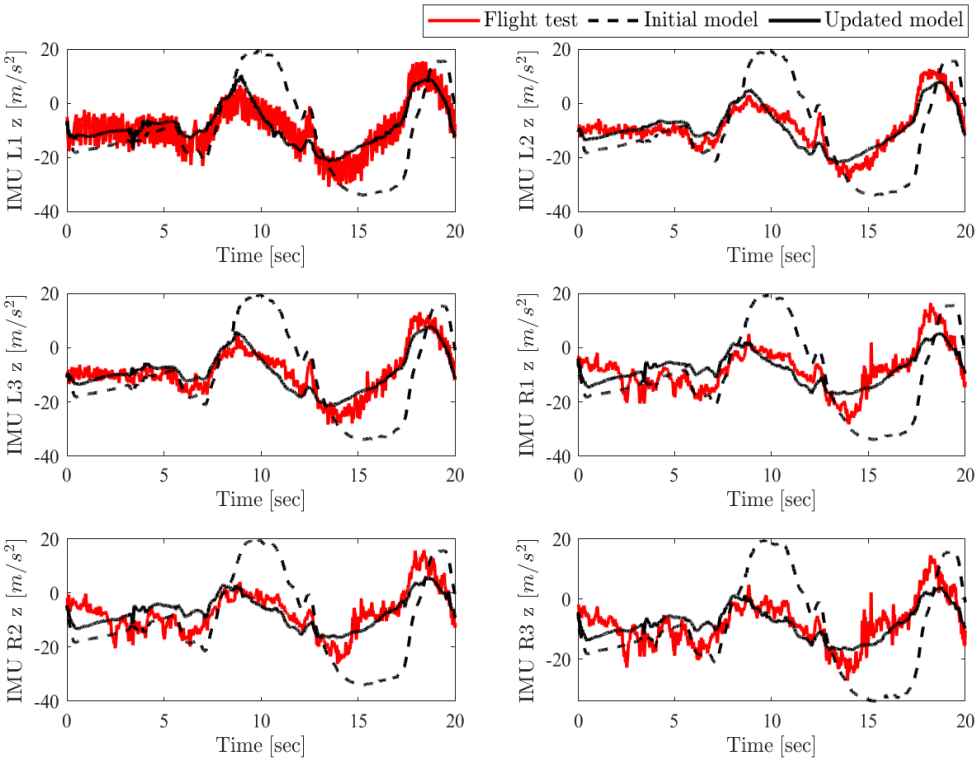


Figure 36: A subset of vertical accelerations $a_{z,\text{IMU}}$ recorded by six IMUs on the wings

5.7 Proof-of-match

For model validation within the framework of proof-of-match procedure, a flight test data set with nearly identical control inputs and trim conditions is needed as chosen in identification tests for model updating. Hence, a suitable set of test data is chosen for model validation, where the aircraft is excited for a pushover-pull-up manoeuvre by an elevator deflection shown in Fig. (37) below.

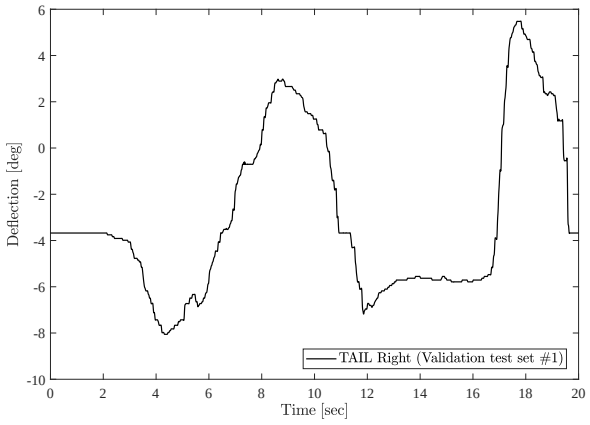


Figure 37: Measured elevator deflections and trim values from flight test for model validation

Trim values	
$\eta_{\text{elev},0}$ [deg]	-3.68
θ_0 [deg]	2.89
$V_{\text{ias},0}$	37.8
$h_{\text{baro},0}$ [m]	809
α_0 [deg]	2.64
β_0 [deg]	0.873

Figure (38) demonstrates that a high degree of match has been achieved between outputs from validation flight test and outputs estimated from the updated aircraft model. The mean Theil's inequality coefficient TIC_{mean} is approximately 0.18 and has decreased by 0.15.

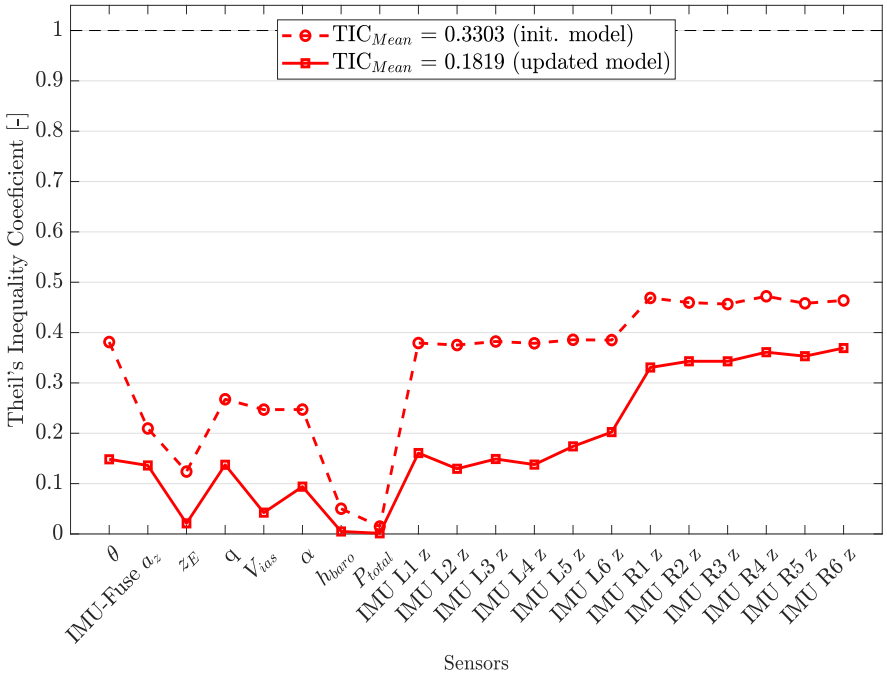


Figure 38: Fit error distribution between outputs from validation test and outputs provided from updated model for proof-of-match procedure

5.8 Conclusion

In this study, it was proved that the proposed updating method can be successfully applied to discrete-time LTI state-space models describing rigid body and flexible aircraft dynamics by employing multiple flight test data. The suggested method permits adjustment of the predicted system matrices $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$ and $\mathbf{D}_d$, beginning with the initial model, by minimizing the output residuals using both state and output equations. The methodology enables the reconstruction of all flight-measured responses with a high degree of overall correlation. The updated model has both phenomenological and behavioural characteristics, with the structure, sparsity, and density degree of the updated matrices remaining quasi-constant. The algorithm of the updating method described here is a three-step procedure and is based on linear least-squares approximation resp. a minimum norm solution which enables an appropriate implementation with fast execution avoiding optimization problems for approximation of solutions of nonlinear differential equations derived from aircraft equations of motion. Another benefit of the presented approach is that updating algorithm can be performed with different data bases derived from flight tests with the nearly identical initial conditions, which would lead to a more physically realistic correction of the system matrices. For the method and model validation, Theil's inequality formulation was utilised. This enables insight into the sources of fit error and hence assesses the quality of the updated model with physical insight.

6 Control System Design and Performance Validation

The lead responsible partner for control system design is ONERA, but both DLR-SR and SZTAKI provided significant contribution to this task.

The present chapter discusses the following topics:

- Short summary of the flight control design tools for baseline, load and flutter control design
- Description of hardware-in-the-loop test results with control laws
- Description of flight test results with baseline control laws
- Description of used tools and how the model based design tools are validated to standardize them in the MDO toolchain

6.1 Baseline control structure

The function of the baseline controller is to control the rigid body motion of the T-Flex aircraft. For this purpose a structured controller configuration was selected, which allows the sequential testing and tuning of the different control laws. The structure can be depicted in Figure 39.

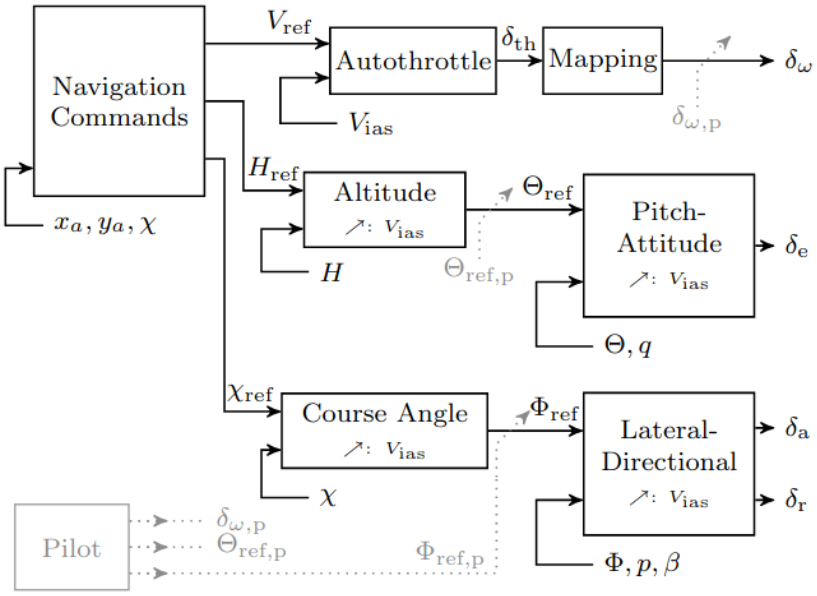


Figure 39: Structure of the baseline controller

The baseline controller has three operational mode:

(i) **Direct Mode**: The direct mode allows the pilot on the ground to bypass the flight control system. The only part active in the flight control computer is the mapping from the received remote-control

signals to the commanded surface deflections. The pilot controls the pitch, roll and yaw axis directly via the aircraft's control surface deflections and its velocity via the thrust setting.

(ii) **Stability Augmentation Mode**: The augmented mode switches on basic augmentation for the pilot. Instead of directly controlling the surfaces the pilot inputs pitch- and roll-attitude commands. The side-slip angle is automatically regulated to zero, reducing the pilots need to control the yaw axis separately. Velocity control remains in direct control, i.e., the pilot controls the velocity via the thrust setting.

(iii) **Autopilot Mode**: In this mode the pilot fully delegates the aircraft control to the flight control system. Altitude, course angle, velocity and side-slip angle are automatically controlled. To fly along the defined test pattern, reference commands based on the aircraft position are generated in a navigation module.

The inner loops of the control system in roll, pitch and yaw provide the basis for the operational model (ii) and (iii). Mode (iii) is the core element of the autopilot adding the outer loops for course angle, altitude and speed control (autothrottle) as illustrated in Figure 39. Thus, a series of cascaded control loops is used to facilitate the control design task. As the cross-coupling between longitudinal and lateral axis is negligible, longitudinal and lateral control design is separated. Thrust commands δ_{th} which are transferred to an engine revolution command δ_ω via a nonlinear mapping and the elevator δ_e are the available actuators for longitudinal control.

6.2 Baseline control design

Lateral-directional control generates aileron (δ_a) and rudder commands (δ_r), which is a multivariable problem and requires the coordinated use of aileron command δ_a and rudder command δ_r. The most inner loop features roll-attitude (Φ) tracking, roll-damping augmentation via the roll rate (p), and coordinated turn capabilities, i.e. turns without side-slip, via feedback of the side-slip angle (β). The outer loop establishes control of the course angle (χ). All controllers are scheduled with velocity to increase performance over the velocity range. Within the fully automated flight mode (iii) the reference signals for the velocity (V_{ref}), altitude (H_{ref}), and course angle (χ_{ref}) are provided by a dedicated navigation algorithm. It uses the GPS longitudinal and lateral position of the aircraft (x_a and y_a) as well as the current course angle (χ) to provide the commands.

Structure wise, the control loops use scheduled elements of proportional-integral-derivative (PID) controller structures with additional roll-offs in the inner loops to ensure that no aeroelastic mode is excited by the baseline controller. A scheduling in dependence of the indicated airspeed V_{ias} is used to ensure an adequate performance over the velocity range from 32 m/s to 70 m/s. For the scheduling a first or second order polynomial in V_{ias} is applied. As an example the proportional gain $k_p = z_0 + z_1 V_{\text{ias}} + z_2 V_{\text{ias}}^2$ is depending quadratically on V_{ias} with the free parameters z_0, z_1, and z_2. A comprehensive summary of the used controller structures for each cascaded loop is provided in Table 7, including the channel description in the controller architecture and the implemented scheduling.

Note that the controller outputs δ_e, δ_a, and δ_r defer from the actual surface inputs to ease the actual control design task. Thus, they need to be transformed to physical actuator commands via an adequate control allocation. The T-Flex aircraft has multiple control surfaces and features combined rudder and elevator surfaces (ruddervators). The commands to the actuators of the two aileron pairs are determined by

$$\begin{array}{rcrcr} \delta_{a,l2} & = & \delta_{a,l3} & = & 0.5\delta_a \\ \delta_{a,r2} & = & \delta_{a,r3} & = & -0.5\delta_a \end{array} \tag{11}$$

Table 7: Summary of the control loops of the FLEXOP baseline flight control system with the inner loop functions (first part) and autopilot functions (second part).

Control Loop	Channel	Structure	Scheduling
Pitch Attitude Control	$(\Theta_{\text{ref}} - \Theta) \rightarrow \delta_e$	PI	2nd-order polyn. in V_{ias}
Pitch Damping	$q \rightarrow \delta_e$	P	1st-order polyn in V_{ias}
Roll Attitude Control	$(\Phi_{\text{ref}} - \Phi) \rightarrow \delta_a$	P	1st-order polyn in V_{ias}
Roll Damping	$p \rightarrow \delta_a$	P	1st-order polyn. in V_{ias}
Yaw Control	$\beta \rightarrow \delta_r$	PID	2nd-order polyn. in V_{ias}
Autothrottle	$(V_{\text{ref}} - V_{\text{ias}}) \rightarrow \delta_{\text{th}}$	2 DOF-PID	none
Altitude	$(H_{\text{ref}} - H) \rightarrow \Theta_{\text{ref}}$	PI	2nd-order polyn. in V_{ias}
Course Angle	$(\chi_{\text{ref}} - \chi) \rightarrow \Phi_{\text{ref}}$	PID	2nd-order polyn. in V_{ias}

to generate the required differential aileron deflections for roll motion control. For the ruddervators superposition of the elevator command δ_e and the rudder command δ_r is applied by

$$\begin{array}{rcccl} \delta_{elev,l1} & = & \delta_{elev,l2} & = & \delta_e + 0.5\delta_r \\ \delta_{elev,r1} & = & \delta_{elev,r2} & = & \delta_e - 0.5\delta_r. \end{array} \tag{12}$$

Thus, symmetric deflections on the left and right of the ruddervators correspond to elevator commands while differential deflections establish rudder commands.

6.2.1 Parameter Tuning

With the baseline controller structure available, the next step is to tune the free parameters of the individual control loops. During this process, an individual optimization problem is set up for the tuning of each control loop. This results in six optimization problems to be solved, as summarized in Table 8. Note that the proportional damping augmentations in roll and pitch are not tuned separately but included in the optimization problems of the corresponding tracking loops. For the inner loops a phase margin of at least $45°$ is demanded. As short period damping is relevant, a minimum of 0.6 is set as optimization constraint. For the roll motion a fast response time of 1 s with good tracking capabilities (steady state error of 0.1) is defined. For the coordinated turn capabilities via the side slip angle feedback a single constraint on the disturbance rejection gain is applied. For the outer loops an adequate frequency separation commonly used in a cascade controller design is applied. The outer loops for controlling attitude and course angle are designed to be five times slower than the inner loops, leading to a corresponding bandwidth or response time constraint. Finally, the auto-throttle is a little more involved due to the complex engine dynamics. Therefore, a model matching problem using the non-linear simulator is used which aims to minimize the recorded error between the desired and achieved response in the simulation.

6.3 Baseline control flight test results

The baseline controller has been tested in an intensive flight test campaign, where the separate loops have been sequentially engaged in the different flights. It is important to emphasize, that the controllers have been designed and tuned based on the available mathematical models of the aircraft, which is only an approximation of the real dynamics. Therefore, several adjustments (fine-tuning) of the control gains might be necessary in order to improve the performance of the baseline controller. This can be performed smoothly due to the hierarchical structure of the controller. Accordingly, different tuning were tested and compared to each other for the best achievable design.

Table 8: Overview of the six defined optimization problems with the number of free parameters and optimization criteria within the model based design procedure of the baseline controller.

Channel	Structure	Free Parameters	Criteria
Pitch Attitude Control incl. Pitch Damping	PI P	8	Damping ration of 0.6 Phase margin of $45°$
Roll Attitude Control incl. Roll Damping	P P	4	Response time of 1s, steady state Error of 0.1, phase margin of $45°$
Yaw Control	PID	9	Disturbance rejection gain
Auto-Throttle	2 DOF-PID	5	Model matching error
Altitude	PI	6	Bandwidth criterion
Course Angle	PID	9	Response time of 5 s

6.3.1 Augmented Mode Flight Tests

The first step of the testing is the so called stability augmented flights, where three inner-loops are engaged (i.e. lateral, longitudinal and yaw) and the pilot directly controls the pitch and roll behaviour of the aircraft, instead of the control surfaces.

In Flight Test 11 (FT11) three different lateral inner loop controllers were tested:

- The gains of AP1.1. were tuned down, for a slower response, in order to safely check the functionalities.
- AP1.2. gains are higher than AP1.1. and provides a more aggressive (hence less robust) tracking performance.
- AP1.3. uses a different Look-Up-Table mapping of the baseline control signals onto the surface deflections.

The three controller is compared in Figure 40, where the roll angle tracking and the corresponding aileron deflections are shown for each configuration. Based on the flights the pilot and the flight test crew have agreed that the behaviour of AP1.3 was acceptable, therefore this version was used in the later flights.

The post-flight numerical analysis supported the findings of the flight test crew, as AP1.2. and AP1.3. provided almost the same bank angle performance with Root Mean Square (RMS) error of 4.15 and 5.6 degrees, respectively, while AP1.1.'s error was approximately 7.3 degrees.

Due to the negligible cross-coupling between the lateral and longitudinal axes, the inner and outer-loop control law for the longitudinal motion could be tested independently. That being said, AP2 of FT11 consisted the pitch attitude inner loop and the engagement of the altitude hold outer loop. The altitude hold feature was tested for the first time, where the aim of the control was to hold the GPS altitude registered at the moment of the baseline autopilot's engagement. The corresponding flight data is given in Figure 41.

As it can be depicted the altitude hold function was working properly, holding the constant altitude with an RMS value of 6.6 meters, due to the various gust disturbances. The performance of the pitch attitude control was very promising as well, with mean-error value of -0.016 degrees. Note that the blue reference signal of the θ tracking plot in Figure 41 is provided by the outer-loop controller in order to maintain the desired altitude. The lower subfigure also shows the computed elevator deflections of the aircraft, where no saturation was observed.

However, the flight test also pointed out that the outer loop's bumpless transfer mechanism was not properly implemented. When the pilot engaged AP2, the aircraft nosed down slightly, resulting in a sharp transient as seen in Figure 41. This feature has been corrected.

The third function of the baseline controller, which was tested in Flight Test 11 was the sideslip loop. This loop was engaged during the testing of AP1.1, AP1.2, AP1.3 and AP2 as well: Figure 42 shows the recorded data. The goal of the controller is to maintain 0 sideslip angle, which is clearly achieved: the mean error of β was approximately 0.07 degrees when the baseline controller was turned on, compared to the 1.8 degree of uncontrolled value.

6.3.2 Altitude Tracking and Autothrottle Tests

The goal of Flight Test 12 was to check further functionalities of the baseline control structure. First, the reference tracking properties of the altitude hold control loop was tested. During this test, the reference altitude was changed by ± 25 meters, instead of the constant value applied in FT11. Figure 43 shows the tracking performance, where an approximate 10 seconds of settling time was observed with a permanent error of ≈ 3 meters. This latter result implies a further fine tuning of the integral part of the longitudinal outer loop. Besides this phenomena, the altitude loop can be considered functional.

Testing of the autothrottle loop involved three different control laws:

- A 2-degrees-of-freedom PID controller with low gains was tested first in order to first check the basic response of the control structure. This version is considered as a robust solution.
- A 2-degrees-of-freedom PID controller with higher gains is labelled as performance solution, where the response is more aggressive.
- Lastly a Total Energy Control solution was also implemented, where the altitude hold and the speed control are coupled together.

Figure 44 compares the control performance of the three controllers. The results are matching the model based expectations clearly: the 'robust' solution provided a slow tracking response for the commanded $4\frac{m}{s}$ step change in the airspeed, compared to the 'performance' version of the same control structure. Based on the post-flight numerical analysis, the TECS solution performed the best: the mean speed error was $-0.55\frac{m}{s}$ only, smaller than the robust ($-1.5\frac{m}{s}$) and the performance ($-1\frac{m}{s}$) errors.

However, the flight tests revealed a few shortcomings of the autothrottle loop:

- The saturation limits of the 2-DOF-PID solutions were implemented wrongly, hence these controllers could not use the entire RPM range of the engine. The problem has been fixed and further flight tests will be performed in order to evaluate these controllers.
- The engine was spooling down when the autothrottle loop was engaged (see the drops of ECU RPM values in Figure 44. The cause of this phenomena is again the bumpless transfer mechanism. In the new version of the autothrottle controller, an integrator tracking solution is implemented in order to avoid sudden drops in the RPM.
- The control action of the autothrottle (for each configurations) showed large variation in the RPM values. Figure 45 illustrates the problem. It can be clearly seen that the requested and the commanded RPM values are moving with a different phase, this indicates that the estimated delay for the engine dynamics differs from the actual value. To overcome this problem an updated engine model is needed and accordingly the re-design of the autothrottle loop.

Based on the experiences of the pilot and the flight test crew, the 'performance' controller has been selected for future future flight tests on the site.

6.3.3 Course Angle Flight Test

Flight Test 14 was dedicated to test the course angle hold and tracking capabilities of the baseline controller. This has been performed in two consecutive steps.

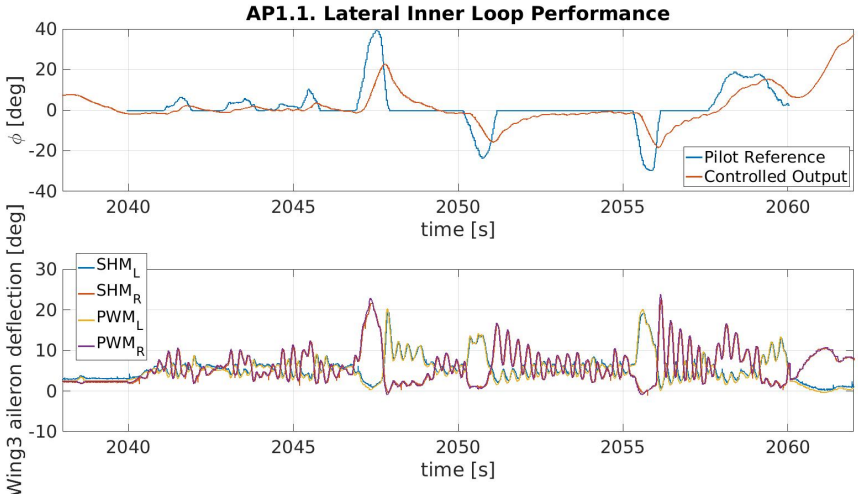
First, the course angle loop was tested through a reference step change, and a coordinated turn maneuver. Figure 46 shows the inner and outer loop performance of the lateral loops. As it can be seen, first the heading reference angle χ was changed in a step-like fashion, which was tracked by the controller properly. After this successful test, a coordinated turn maneuver was performed, where the course angle was changed incrementally to fly a complete circle with the T-Flex aircraft. It can be depicted in Figure 46 that the inner loop's tracking performance is excellent, while the outer loop follows the reference signal with a slight delay. Figure 47 shows the trajectory of the aircraft during the maneuver.

Upon the successful testing of the course angle loop, the navigation logic was tested. The goal was to fly the complete horserace pattern in a fully automated manner, i.e. all baseline loops engaged. Figure 48 shows the flight trajectory in North-East coordinate system, where one can clearly observe that the entire functionality of the baseline works smoothly and is able to fly the desired pattern fully autonomously.

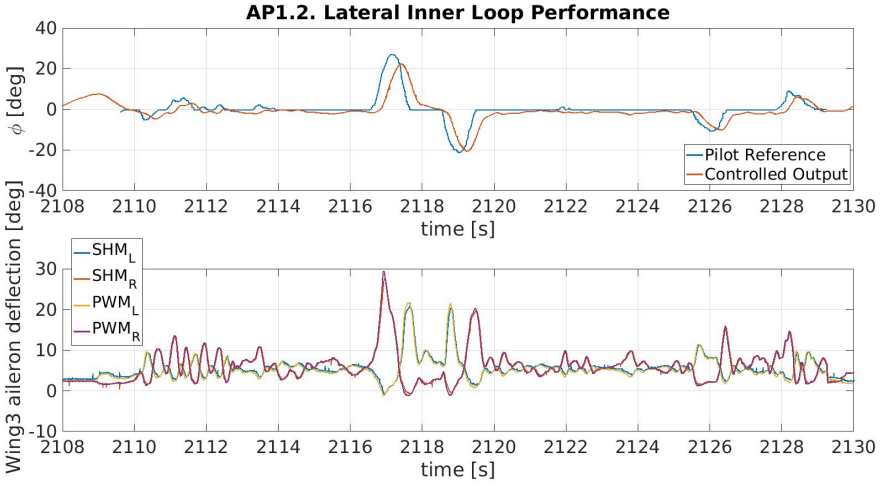
6.3.4 Preparation for Flutter tests

As the baseline functionalities have been tested with acceptable control performances, further a flight test was performed in order to facilitate future flight tests for flutter control. The objective of the test was to gradually increase the speed of the aircraft and see the behaviour of the baseline controller. This test is useful in the preparation of the flutter flight tests, where the speed will be increased in the straight legs of the horserace pattern. In addition, these tests are also validating the range of the baseline controller. As mentioned previously, the gains of the controller are scheduled with the indicated airspeed, hence expanding the speed range expands also the domain of the baseline controller.

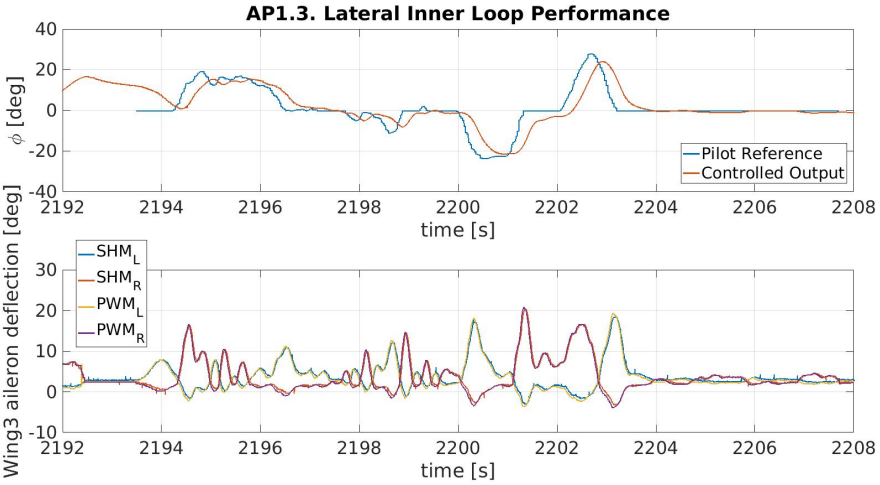
In order to keep the aircraft within the view sight of the pilot, full circles have been flown with increasing speed. Figures 49-51. Figure 49 shows the lateral inner-outer loop performances during the flight, where similar performance has been observed as in the previous flight tests. The corresponding trajectory of the aircarft is shown in Figure 50 with the multiple full circles. Lastly, Figure 51 shows the speed profile during the flight. It can be seen that the autothrottle was able to track the increasing speed reference until $50\frac{m}{s}$, but the control performance was diminished above this speed. This is due to the previously noticed incorrect saturation limit on the RPM values. Nevertheless, it is also worth to mention that the tracking performance below $50\frac{m}{s}$ was better than in earlier flight tests, this is due to the design process of the autothrottle, which was tuned through non-linear optimization and with all other baseline loops engaged.



((a)) Lateral inner loop for AP1.1. - Flight Test 11



((b)) Lateral inner loop for AP1.2. - Flight Test 11



((c)) Lateral inner loop for AP1.3. - Flight Test 11

Figure 40: Comparison of different lateral inner loop controllers during Flight Test 11

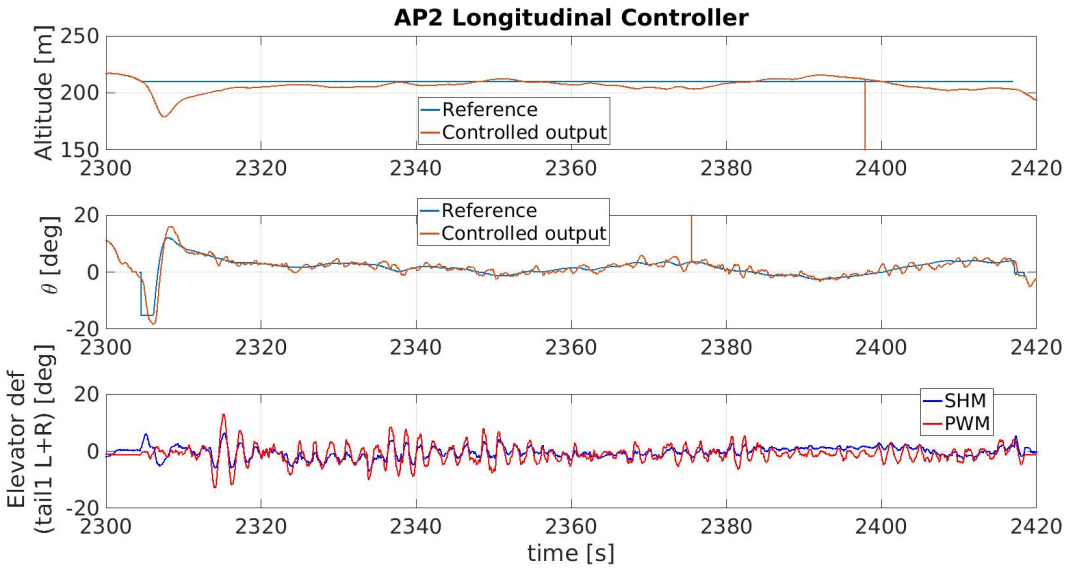


Figure 41: Flight test evaluation of the longitudinal control laws

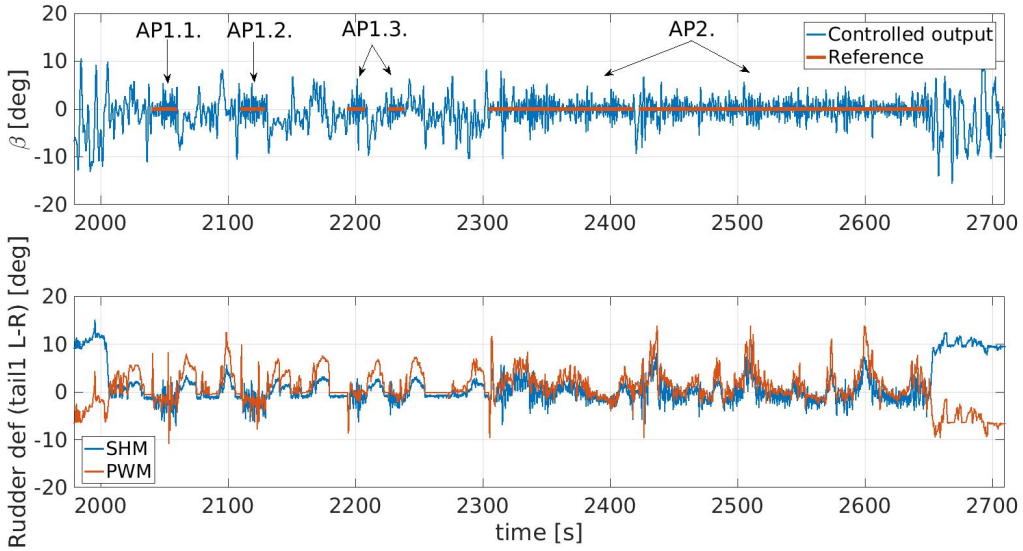


Figure 42: Sideslip loop performance during Flight Test 11

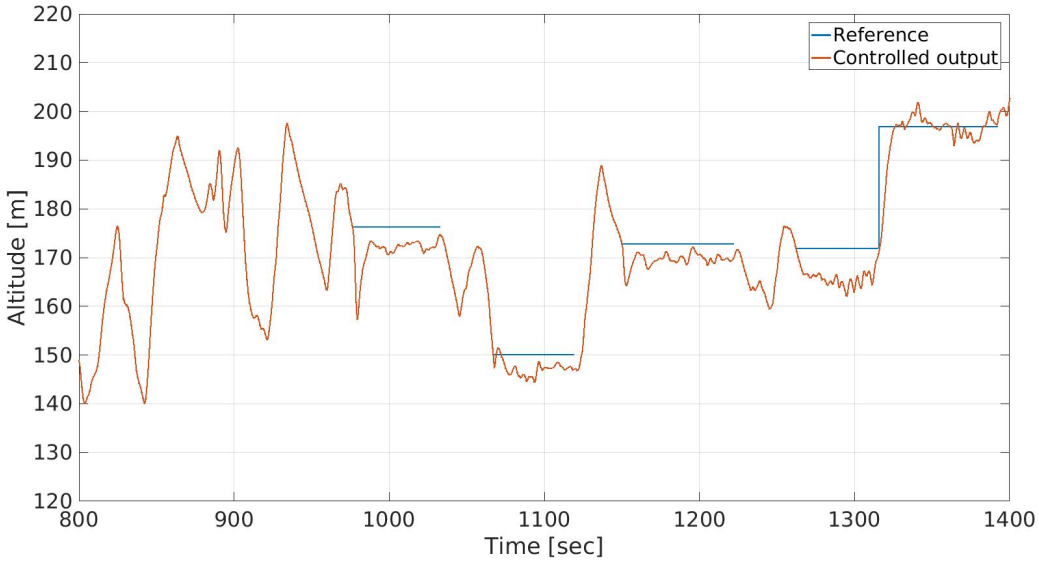


Figure 43: Altitude reference tracking during FT12

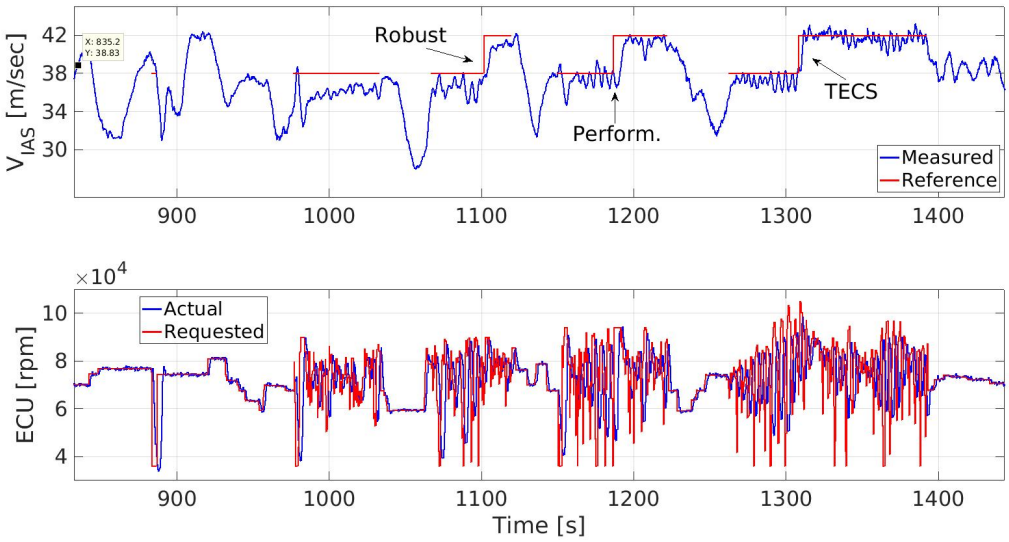


Figure 44: Comparison of different autothrottle controllers during FT12

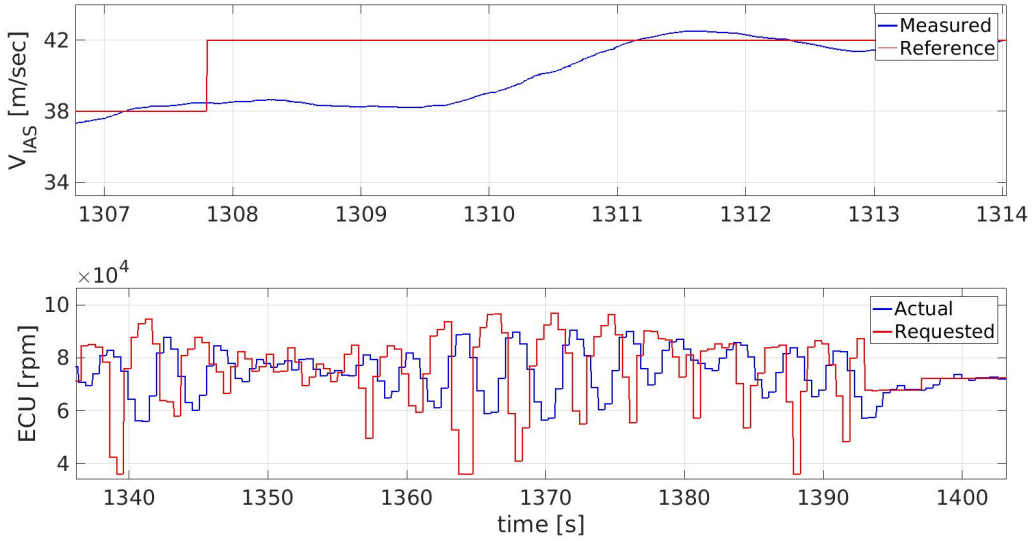


Figure 45: Speed tracking and the corresponding RPM signal

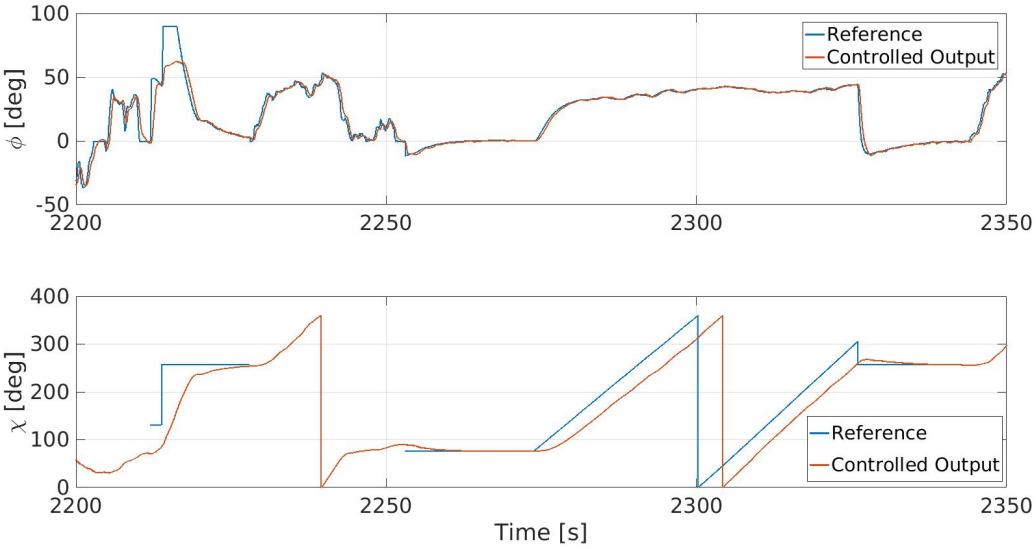


Figure 46: Course angle tracking performance during reference step change and coordinated turn

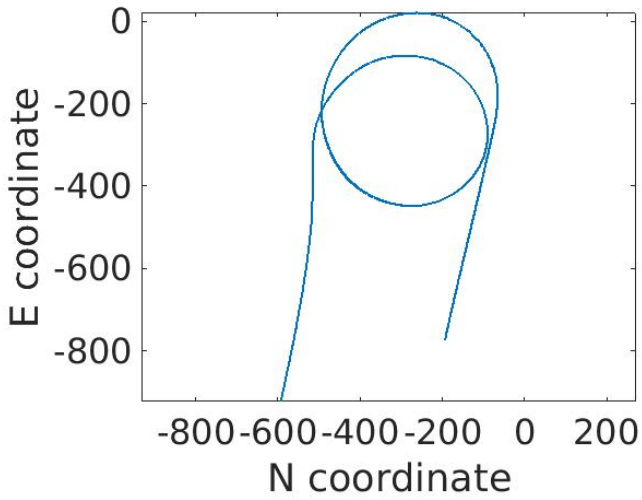


Figure 47: Coordinated turn

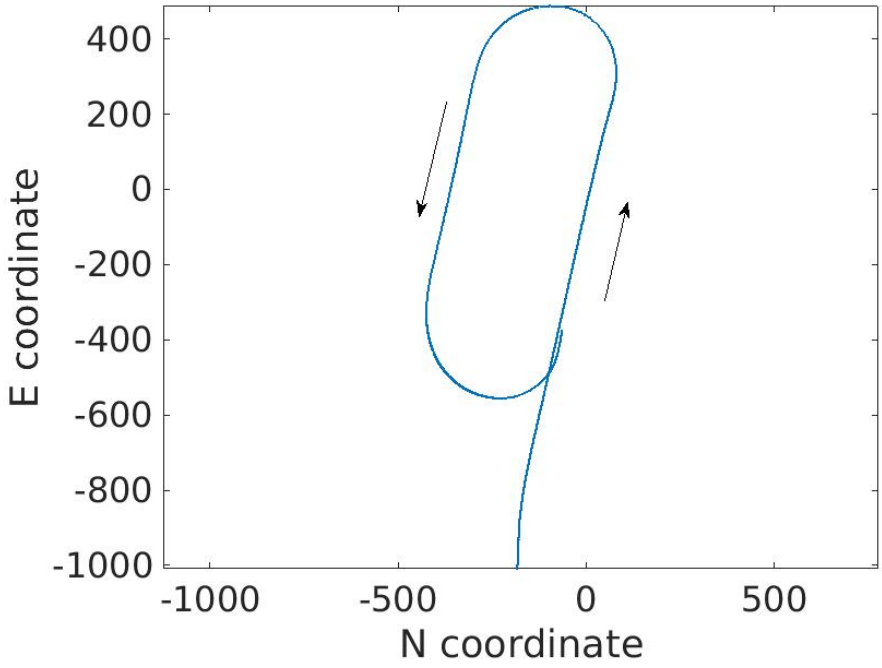


Figure 48: Horserace flight pattern

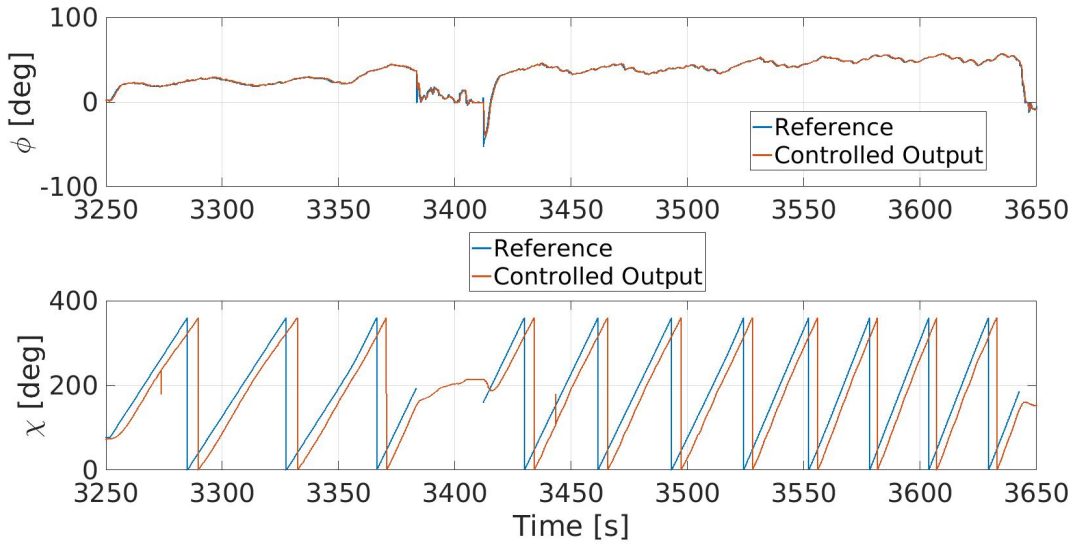


Figure 49: Course angle for full circle tests with increasing speed during FT16

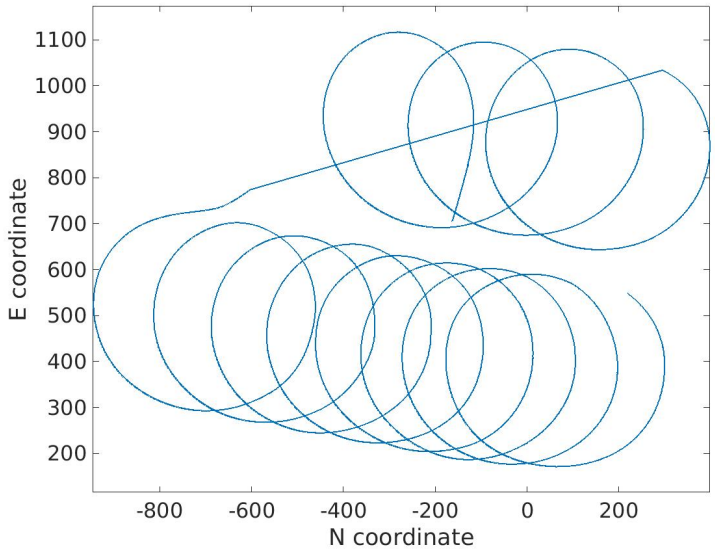


Figure 50: Full circle trajectories with increasing speed during FT16

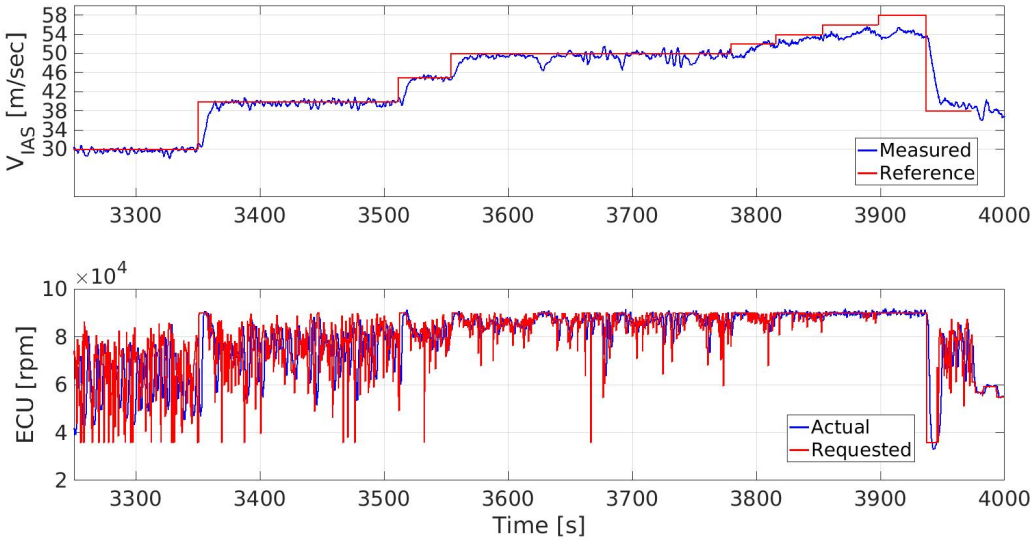


Figure 51: Increasing speed during FT16

7 Validation of data driven wingshape estimation by analytic models

One of the main goals of the FLiPASED project is to develop drag reduction control laws for aircraft with highly flexible wings. The main motivation of this chapter is to present two different approaches, one relying on purely theoretical models, the other using experimental data utilizing machine learning tools, to estimate the flexible dynamics of the T-Flex demonstrator which can be then used for the design of a wing shape controller for drag reduction purposes.

The most straightforward solution for designing a state predictor is the Kalman filter ([18]) for linear systems and the extended Kalman filter (EKF) in the case of nonlinear systems, which is widely used for inertial estimation of wing shape ([20]). However, it has two main drawbacks. First, it requires the exact mathematical state-space description of the nonlinear system, which might not be available or simply its use is computationally too expensive. The second drawback is that the EKF requires knowledge of noises and disturbances related to observations and states. To solve the first issue, the approximation of the full, nonlinear system with a Linear Parameter Varying (LPV) model ([39]) can be considered, as it can significantly ease the computational burden, while being suitable for use with an EKF ([32]). However, noise information is still required.

Data-driven approaches have the great advantage that they can be used for inertial odometry ([10]) and inertial aided navigation ([45]) problems without needing any specific information about model or observation uncertainties. To utilize this advantage the new KalmanNet architecture was created by [31], which is based on Kalman filtering, but it uses a Recurrent Neural Network (RNN) to estimate the Kalman gain. As a result, it does not need any information about the noises and model uncertainties present. The standard KalmanNet architecture uses linear layers and a Gated Recurrent Unit (GRU) to be able to establish correlations between data samples in time. The main novelty of this research are the followings. First, we apply the KalmanNet architecture to a complex LPV model of a real-life system with high dimensionality. This required a new loss calculation to ensure the stability of the whole state predictor, the slight modification of the layer dimensions in order to decrease computational burden and the implementation of a hyperparameter optimizing algorithm as well. Second, we propose a different neural network architecture for the KalmanNet which uses 1D convoultional layers alongside the GRU layer, since 1D convolution can be effectively used for processing timeseries data ([33]).

This chapter presents the LPV-based EKF and the KalmanNet for predicting the modal coordinates and aerodynamic lag states of the nonlinear model of an Unmanned Aerial Vehicle (UAV) T-Flex, which was created within the FLEXOP project for demonstrator purposes. ([42]) The training of the neural network was done in Python with PyTorch, while testing was carried out with MATLAB, Simulink simulations. The chapter is organized as follows. In Section 7.1, the dynamic model of the FLEXOP demonstrator is presented. Section 7.2. introduces the reduced, LPV model of the original nonlinear system, and explains the idea of the LPV-based EKF. In Section 7.3. the KalmanNet's basic structure is summarized with the two different neural network architecture: a linear one and a convolutional one both utilizing a GRU cell. Section 7.4 presents the results of the modal coordinate and lag state estimations. The accuracy of the LPV-based EKF and the KalmanNet is compared. Conclusions are drawn in Section 7.5.

7.1 T-FLEX demonstrator dynamic model

The chosen system for our research is the nonlinear, state space representation of the FLEXOP demonstrator aircraft. The model consists of the following main parts: states that are responsible for the description of the rigid body dynamics; states related to flexible dynamics and aerodynamics, and finally,

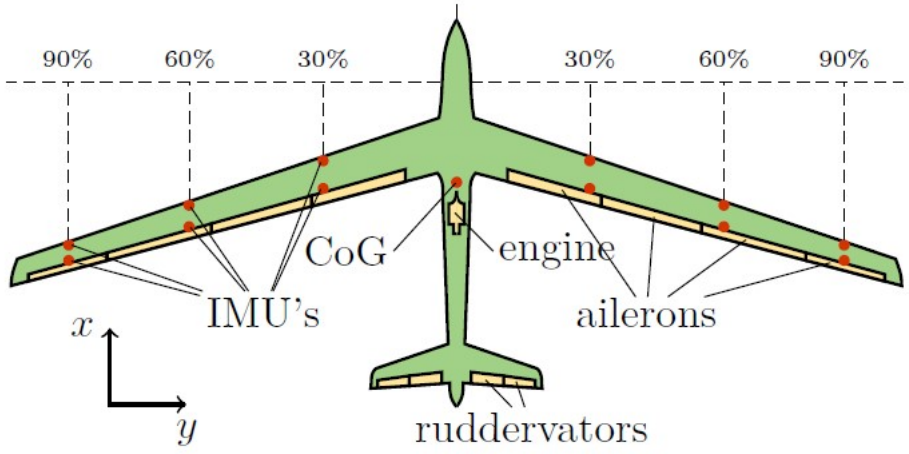


Figure 52: Demonstrator control surfaces and IMU locations

states that represent the control surface inputs and their first derivatives. The state vector is denoted as $\underline{x} \in \Re^{48}$. The rigid body motion is represented with a 6-DOF model with 12 states: states of translational and angular velocities, position, and orientation.

The states which describe the flexible dynamics are the modal coordinates and their first derivatives. Due to the reduced order modelling only the six most significant modal coordinates and two aerodynamic lag states were considered. This is denoted as $\underline{x}_{\text{flex}} \subset \underline{x}$ and $\underline{x}_{\text{flex}} \in \Re^{14}$. The main objective of this research is the estimation of $\underline{x}_{\text{flex}} = [U_{f1}, U_{f2}, U_{f3}, U_{f4}, U_{f5}, U_{f6}, \dot{U}_{f1}, \dot{U}_{f2}, \dot{U}_{f3}, \dot{U}_{f4}, \dot{U}_{f5}, \dot{U}_{f6}, \text{lag}_1, \text{lag}_2]^T$. Further details of the modeling and model order reduction are given in [42], [40] and [24].

The elements of the input vector, $\underline{u} \in \Re^{19}$, of the T-Flex aircraft model are (Figure 52): two landing gears (GearR/L), two landing gear wheelbrakes (WheelbrakeR/L), two airbrakes located on the aircraft's fuselage (AirbrakeR/L) and one turbofan engine (Throttle). The demonstrator has 12 control surfaces: four-four ailerons (AileronR/L) on each wing and on the V-tail two-two 'ruddervators' (TailR/L). From the inputs, the landing gear-related ones are insignificant in our research since the estimation of the structural dynamics is only conducted during airborne operations.

The output vector, $\underline{y} \in \Re^{64}$, of the demonstrator model is made up from the following elements. It has 23 rigid body related outputs, which provide information about the aircraft's position (x_E, y_E, z_E), orientation (ϕ, θ, ψ), translational (v_N, v_E, v_D) and angular velocity (p, q, r), and acceleration (a_{xB}, a_{yB}, a_{zB}). Furthermore, the course angle (χ), angle of attack (α), sideslip angle (β), air (p_a) and total pressure (p_T), barometric altitude (h_{baro}), indicated (v_{IAS}) and the true airspeeds (v_{TAS}) are measured as well. Each wing of the demonstrator has six-six inertial measurement units (IMUs). An IMU provides acceleration and angular velocity data around the x-, y- and z-axis of its coordinate system. We opted for such an IMU configuration, where the IMUs on the leading-edge measure accelerations in the x, y, and z directions, while the IMUs on the trailing-edge provide angular velocity data around the x- and y-axis, and acceleration data in the z direction. The exact location of the IMUs can be seen in Figure 52 as well. In addition, the wingtip coordinates can be measured with a mono camera for preventing acceleration-based estimation errors from diverging in time ([20]). On each wing, the coordinates of four wingtip points are measured in each direction.

7.2 Model based estimation of flexible dynamics

7.2.1 LPV model

The linear parameter varying (LPV) model is an approximation to describe the behaviour of a nonlinear system ([39]). It is essentially a pointwise linearization of a state-space system: the nonlinear system is linearized at different trim points that are defined by - the so called – scheduling parameters. The scheduling parameters create a multidimensional grid, and a linear, state-space model is assigned to every grid point. The state-space description of a discrete time LPV system is described by

$$\begin{aligned} \underline{x}[k] &= A(\underline{\rho}[k])\underline{x}[k-1] + B(\underline{\rho}[k])\underline{u}[k], \\ \underline{y}[k] &= C(\underline{\rho}[k])\underline{x}[k] + D(\underline{\rho}[k])\underline{u}[k], \end{aligned} \tag{13}$$

where $\underline{\rho}[k]$ is the time varying vector of the scheduling parameters in time step k. $A \in \Re^{48\times48}$, $B \in \Re^{48\times19}$, $C \in \Re^{64\times64}$ and $D \in \Re^{64\times19}$ are the state space matrices of the LPV system that are dependent of the current $\underline{\rho}[k]$ vector. $\underline{x}[k]$ denotes the state vector, $\underline{u}[k]$ the system's input vector, while $\underline{y}[k]$ is the system's output vector in time step k.

In our work, we created an LPV approximation of the nonlinear bottom-up model of the T-Flex demonstrator aircraft with two scheduling parameters, $\underline{\rho} \in \Re^2$: the true airspeed ($v_{\mathrm{TAS}}$) and the roll angle ($\phi$) sensor outputs. The grid for the LPV model consisted of airspeed values from $30\mathrm{m/s}$ to $50\mathrm{m/s}$ with a $1\mathrm{m/s}$ resolution while the roll angles from $0°$ to $40°$ with $10°$ resolution. Then the nonlinear model was trimmed at each grid point. The resulting LPV model structure was then further refined to $0.1\mathrm{m/s}$ and $1°$ resolution with the spline interpolation method of the LPVTools MATLAB toolbox ([2]).

7.2.2 LPV-based Kalman filtering

For the model-based wing-shape estimation, an extended Kalman filter (EKF) was used. The EKF pipeline requires the full, nonlinear state-space description of the system as well as information about the model noise and observation noise in the form of noise covariance matrices. The nonlinear system's state-space representation with time discretization in time step k:

$$\begin{aligned} \underline{x}[k] &= \mathbf{f}(\underline{x}[k-1], \underline{u}[k]) + \underline{w}[k] \\ \underline{y}[k] &= \mathbf{h}(\underline{x}[k], \underline{u}[k]) + \underline{v}[k]. \end{aligned} \tag{14}$$

The nonlinear function $\mathbf{f}(.)$ is called state-transition function, while $\mathbf{h}(.)$ is called state-observation function. The $\underline{w}[k] \in \Re^{48}$ and $\underline{v}[k] \in \Re^{64}$ vectors are the model noise and observation noise vectors respectively. However, the explicit mathematical description – the nonlinear state-transition and state-observation functions – of the T-Flex demonstrator was not available to us, therefore a unique approach was necessary for the design of the EKF.

The general framework of the EKF consists of two main steps: *prediction* and *update*. In these steps, pointwise linearization is used to approximate the behaviour of the nonlinear system. More precisely the Jacobians of the nonlinear state-transition and state-observation functions are calculated to get the linear, state-space matrices $A[k]$, $B[k]$, $C[k]$ and $D[k]$ at each time step. In the *prediction* step the prior state estimation is calculated using the inputs of the current time step and the estimations from the previous time step with

$$\underline{\hat{x}}[k|k-1] = \mathbf{f}(\underline{\hat{x}}[k-1|k-1], \underline{u}[k]). \tag{15}$$

The prior state estimation covariance $P \in \Re^{48\times48}$ is

$$P[k|k-1] = A[k]P[k-1|k-1]A[k]^T + Q. \tag{16}$$

In the *update* step, first, the innovation

$$\underline{\tilde{y}}[k] = \underline{y}[k] - \mathbf{h}(\underline{\hat{x}}[k|k-1], \underline{u}[k]) \tag{17}$$

is calculated. then the near-optimal Kalman gain, $K_G \in \Re^{64\times 48}$

$$K_G[k] = P[k|k-1]C[k]^T(C[k]P[k|k-1]C[k]^T + R)^{-1}. \tag{18}$$

With the help of the Kalman gain, the posterior state vector

$$\underline{\hat{x}}[k|k] = \underline{\hat{x}}[k|k-1] + K_G[k]\underline{\tilde{y}}[k], \tag{19}$$

and state prediction covariance

$$P[k|k] = (I - K_G[k]C[k])P[k|k-1] \tag{20}$$

is computed. In the equations, the $Q \in \Re^{48\times 48}$ and $R \in \Re^{64\times 64}$ matrices are the model and the observation noise covariance matrices respectively.

To obtain an appropriate pointwise linearization we used our LPV model. During simulation, the true airspeed and roll angle is measured at each time step which then can be used to select an approximating linear system from the LPV model. The selected model's state-space matrices are fed to the EKF as the current $A[k]$, $B[k]$, $C[k]$ and $D[k]$ matrices. Then the EKF conducts the prediction and update steps. For acquiring the model noise matrix both the nonlinear and the LPV models were simulated with doublet inputs on the control surfaces and then the measured outputs and states were compared, and variances of the differences calculated. For the observation covariance matrix, the T-Flex's onboard sensors' noise variances were used. These were specified based on the sensors' datasheets. Note that we used the assumption that both noises have 0 mean, normal distributions, and the noise vectors at each time step are mutually independent.

7.3 Data-driven estimation of flexible dynamics

7.3.1 KalmanNet architecture

The other approach for estimating the flexible dynamics of the demonstrator is to use artificial intelligence, more precisely a neural network. Our choice was to use the relatively new KalmanNet architecture ([31]). KalmanNet combines Kalman filtering with a neural network as it still uses the current inputs and observations for giving state estimations, however, the near optimal Kalman gain is provided by a trained recurrent neural network (RNN). The main advantage of this is that KalmanNet does not require either the model (Q) or the observation noise covariance matrices (R) and it can effectively overcome any uncertainties or errors in the model of the dynamic system while retaining engineering insight about the physical system.

The KalmanNet pipeline is the following. It still consists of a *prediction* and an *update* step just like a Kalman filter. In the prediction step however only the prior state prediction (15) is calculated, the state prediction covariance (P) is not. In the update step, first the innovation difference ($\Delta\underline{y}[k] \in \Re^{64}$) and the forward update difference ($\Delta\underline{x}[k] \in \Re^{48}$) are computed:

$$\Delta\underline{y}[k] = \underline{y}[k] - \underline{\hat{y}}[k|k-1] \tag{21}$$

$$\Delta\underline{\hat{x}}[k] = \underline{\hat{x}}[k-1|k-1] - \underline{\hat{x}}[k-1|k-2]. \tag{22}$$

These act as the input features for the recurrent neural network. Furthermore, the roll angle (ϕ) scheduling parameter was also used as an input feature in order to make the handling of turning manoeuvres easier. The RNN then provides the Kalman gain in each time step. With the Kalman gain and using the innovation, the *a posteriori* state prediction vector is calculated as in (17) and (19) respectively. As it can be seen, neither the state estimation covariance matrix (P) nor the noise covariance matrices are required: the whole pipeline works without any information about the model or observation noises.

The standard Kalman gain predicting neural network presented in [31] uses a Gated Recurrent Unit (GRU) as the recurrent layer and linear layers with Rectified Linear Units (ReLU) as activation function. The architecture is the following: it has a linear layer as the input layer with ReLU activation followed by the GRU. After the GRU layer, there is another linear layer with ReLU activation, then the linear, output layer. We slightly decreased the dimensions of each layer compared to the proposed architecture. The reason behind this modification was mainly memory consumption related. Since our aircraft model has a relatively high number of states (48) and outputs (64) using the original layer dimensions, we frequently ran out of GPU memory, while training with CPU was extremely slow.

Apart from the original *linear* architecture, we implemented a different neural network loosely based on the one presented in [45]. The network architecture still uses a GRU cell, however instead of linear layers it uses 3 convolutional blocks at the beginning. A convolutional block consists of a 1D convolutional layer followed by a ReLU activation function. After the ReLU a Batch Normalization layer is used which is followed by a Dropout layer with 0.25 dropout probability. A fully connected layer is only kept at the end of the network for providing the Kalman gain matrix. The kernel size for each 1D convolution layer is seven. As the 1D convolutional layer requires a trajectory, or time-window of input features, simply using the previously mentioned forward update difference ($\Delta\underline{y}[k]$), innovation difference ($\Delta\underline{\hat{x}}[k]$) and roll angle ($\phi$) input features of the current timestep is not adequate. Therefore, we used the input features of the current timesteps and the input features form the previous 19 timesteps in the time-window buffer.

The full KalmanNet pipeline with the convolutional neural network is presented in Figure 53. The number of features is shown below the convolutional blocks, the pool size below the max pooling layer. The number of units is indicated underneath the GRU and the linear layer. The dropout rate is shown below the dropout layer.

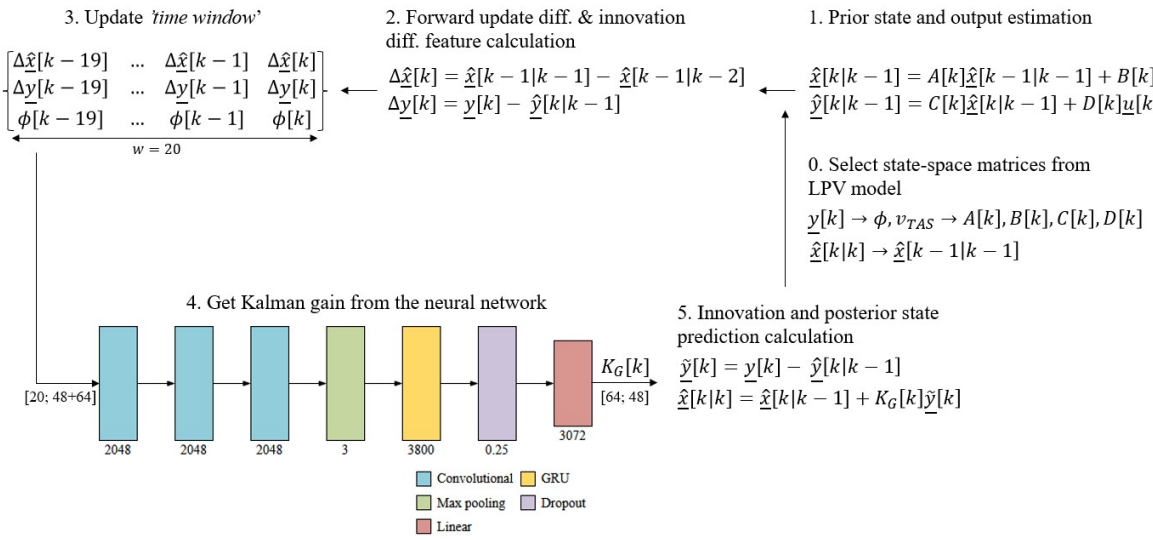


Figure 53: KalmanNet pipeline

7.3.2 Training data

Training, validation, and testing datasets were generated using the T-Flex's high-fidelity nonlinear Simulink model. In order to create a rich dataset, while having realistic flight conditions, an '8-shaped' track was generated using the baseline controller of the aircraft also implemented in Simulink.

To make each dataset different, randomized windgust and turbulence disturbances were used as well,

together with Gaussian sensor noise, based on the flight test results of the demonstrator [3]. The main purpose of applying wind loads is having disturbances that cannot be incorporated into any covariance matrix, and generating trajectories that are more realistic. Going through the generated track takes the aircraft roughly $120\mathrm{s}$ with the initial velocity of $42\mathrm{m/s}$, so the duration of the simulation was set accordingly. The sampling time was set to $5\mathrm{ms}$, which results in a 24000-sample long trajectory for each dataset. This was then split into 20, 1200-sample (6-second) long batches. For training, in each epoch 8 batches were randomly selected from the total 20. However, validation, as well as testing, was conducted on the whole 20-piece, 6-second-long trajectory in order to get meaningful information about the architecture's performance.

7.3.3 Training details

For the two neural network architectures the training parameters were set with the use of a custom made hyperparameter optimization algorithm based on *RayTune*. The hyperparameter optimization had 20 runs, each lasting for 25 epochs. Otherwise, the hyperparameter optimization used the same pipeline as normal training runs.

The optimizer algorithm was ADAM for both architectures. In order to avoid overfitting, weight decay was used. The prediction accuracy was calculated with mean squared error (MSE) function. However – although the linearized aircraft model is a stable system, the system's poles are relatively close to the unstable region. So, a stability criterion was added to the MSE loss function. It is possible to describe the complex system of the aircraft model joined with the Kalman filter with an error system:

$$\underline{e}[k+1] = (A - K_G C)\underline{e}[k], \tag{23}$$

where K_G is the Kalman gain, $\underline{e}[k] \in \Re^{48}$ is the state prediction difference at time step k. If the error system's state transition matrix $(A - K_G C)$ has any unstable poles, then the whole system is unstable. Hence, the MSE loss was extended with the distance of the error system poles from the boundary of stability if it is larger than 0, thus making the loss value larger if the computed Kalman gain results in an unstable error system. This is especially useful for the convergence of the training.

The error metrics were defined in decibels for the sake of convenience during plotting because the freshly initialized network tends to produce large errors. It is simply calculated with the following formula:

$$\mathrm{loss}_{\mathrm{MSE}}^{\mathrm{dB}} = 10\ \log_{10}(\mathrm{loss}_{\mathrm{MSE}}). \tag{24}$$

Of course, the metric was solely used for evaluation and plotting. For optimizing the network weights, the standard MSE loss value was used during backpropagation.

For initializing each layer's weights, standard normal distribution was used. However, as the whole architecture incorporates a discrete time system, it was very sensitive to the initial weight values. Therefore, standard deviation of the normal distribution for the initialization had to be chosen to be very small $(5 \cdot 10^{-6})$ for avoiding the otherwise highly diverging training process.

It is important to mention that the 1st architecture's performance proved to be more stable than the 2nd which had the tendency to get stuck in local optima. So, to overcome this issue the reduction of the learning rate during training was necessary in the case of the 2nd network. The threshold was set at $-42\mathrm{dB}$ – according to the decibel-based error metric – and the reduction factor was 0.05. The new learning rate was calculated as $\mathrm{lr}^{\mathrm{new}} = \mathrm{factor} \cdot \mathrm{lr}^{\mathrm{old}}$. The used hyperparameters for each neural network architecture are presented in Table 9 .

Table 9: Hyperparameters

Hyperparameter	Linear	Convolutional
Learning rate	$3.2 \cdot 10^{-6}$	$1.5 \cdot 10^{-7}$
Weight decay	$7.5 \cdot 10^{-5}$	$9.5 \cdot 10^{-5}$

7.4 Results

7.4.1 LPV-based EKF

The LPV-based EKF's performance was evaluated on the test dataset generated for the KalmanNet. This means that during the simulation, the aircraft followed the same 8-shaped track with wind and turbulence disturbances independent from the training data. The initial conditions were $42\mathrm{m/s}$ flight speed at $800\mathrm{m}$ altitude, with 2° course angle. The whole simulation lasted for 120 seconds which corresponds to 1 full lap around the track. The results of the EKF-based state predictions are shown in Figure 54, where the data with the 'nonlinear' label show the states of the nonlinear model, while the 'EKF' show the states estimated by the filter. Since the main purpose of the observer design is to observe the flexible dynamics of the states, only the results for these states are presented. The first 4 modal coordinates are plotted where U_{f1} is the 1st symmetric bending and U_{f2} the 1st asymmetric bending mode. U_{f3} denotes the 1st symmetric torsion mode and U_{f4} is the 1st asymmetric torsion mode. The 2 aerodynamic lag states are plotted as well.

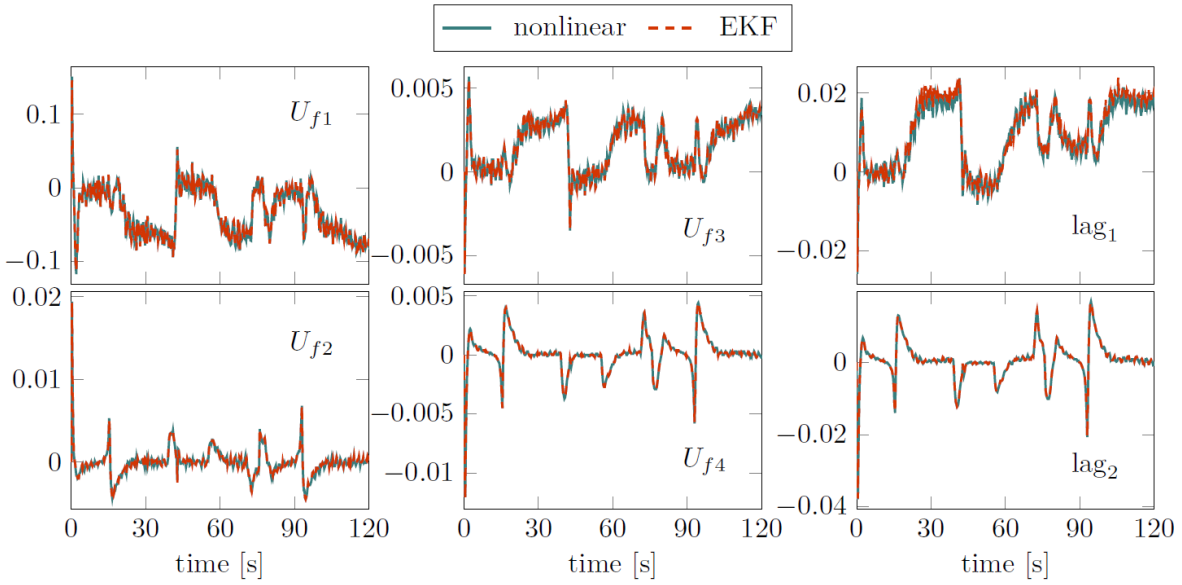


Figure 54: LPV-based EKF results

From the results, it can be concluded that the designed filter accurately predicts the modal coordinates and the aerodynamic lag states even in the presence of wind disturbances which's effects cannot be incorporated into the observation noise matrix. The predictions' root mean squared error (RMSE) for the plotted states is $7.62 \cdot 10^{-4}$. Minor errors occur only during turning manoeuvres in state lag_1. The reason behind these is that the LPV model is still just an approximation of the real, nonlinear system. However, these inaccuracies are inside the error tolerance for this problem.

7.4.2 KalmanNet

For training, the generated 20 batches of 1200 sample long trajectories were used with a sampling time of 5ms. The inputs for the KalmanNet architecture were the observations and control surface and throttle inputs of the nonlinear model. The target for the network were the nonlinear model's states. During training, validation and testing the KalmanNet used the LPV model of the nonlinear model. For evaluation the initial trim condition of $42\mathrm{m/s}$ true airspeed was used at $800\mathrm{m}$ altitude with initial course angle of 2° just like in the case of the EKF.

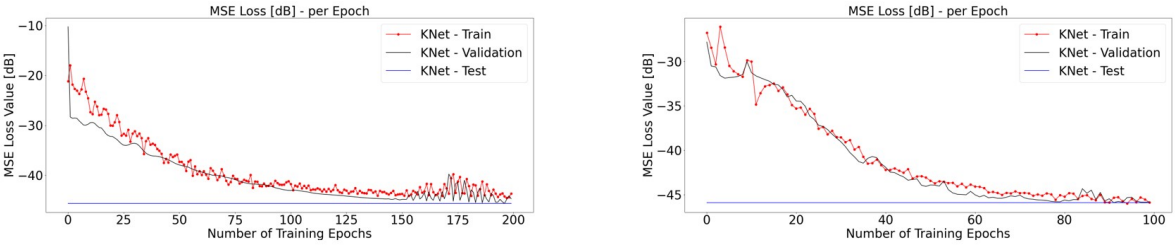


Figure 55: Linear (*left*) and convolutional (*right*) architecture training graphs

Neural network with linear layers

First, we tried the slightly modified original KalmanNet architecture which uses linear layers with the GRU. The training lasted for 200 epochs. Using an Nvidia Tesla V100 GPU with 32GBs of RAM, the whole procedure took 23 hours. The summary of the training is presented in Figure 55 (*left*). The previously discussed decibel-based metric was used for plotting.

The trained model was evaluated on the same dataset as the LPV-based EKF. The results for the first 4 modal coordinates and the 2 aerodynamic lag states can be seen in Figure 56. From the results, it

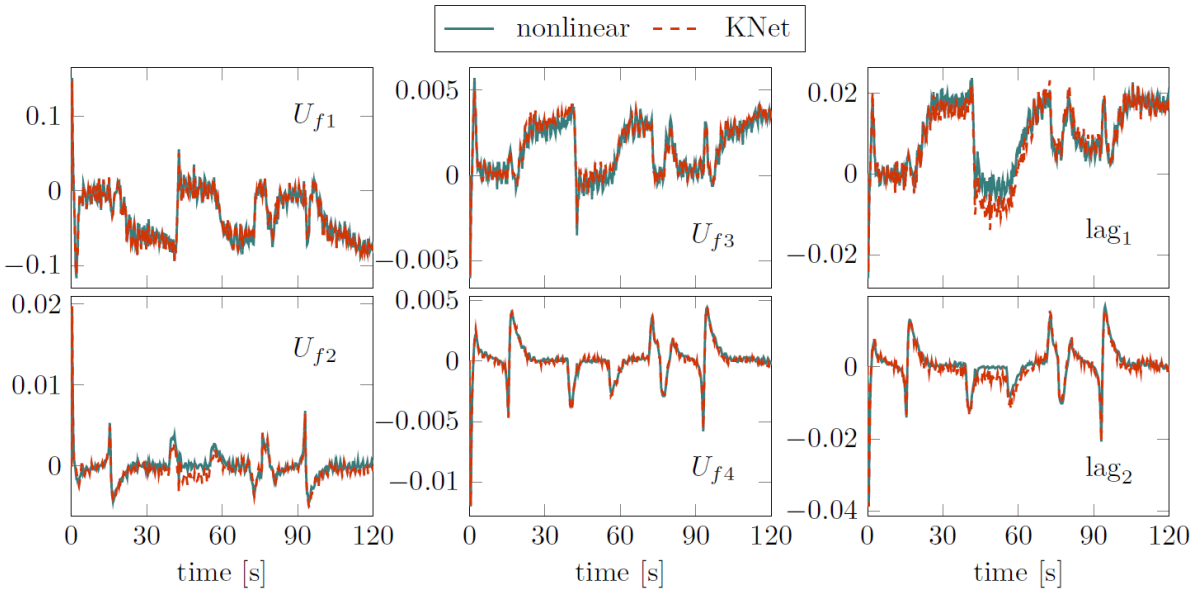


Figure 56: KalmanNet results with linear architecture

can be seen that the neural network managed to produce comparable results with the EKF in the case of U_{f1}, U_{f3}, U_{f4} and lag_2. In the prediction of U_{f2} and lag_2 however a larger error is present between

time step 8000 – 12000. As, in this interval, the aircraft conducts a heavier acceleration, it is possible that the training data is not comprehensive and balanced enough to make the neural network capable of learning such manoeuvre. The RMSE value of the predictions is $1.67 \cdot 10^{-3}$. Neural network with convolutional layers

Second, we implemented the proposed network architectures with convolutional layers. In this case, the training duration was 100 epochs. That took 15 hours to complete using the V100 GPU. The 1D convolution expects a time series as an input, a 20-sample long time window was used which equals to 0.1s trajectory. Unfortunately, we could not use a larger window size, because we ran out of GPU memory (and training with CPU was not feasible, due to its extremely slow execution speed). The training graph is shown in Figure 55 (*right*) with loss values in decibels.

Testing was done with the same dataset as in the previous approaches. The results for the first 4 modal coordinates and the 2 aerodynamic lag states are presented in Figure 57. The results indicate

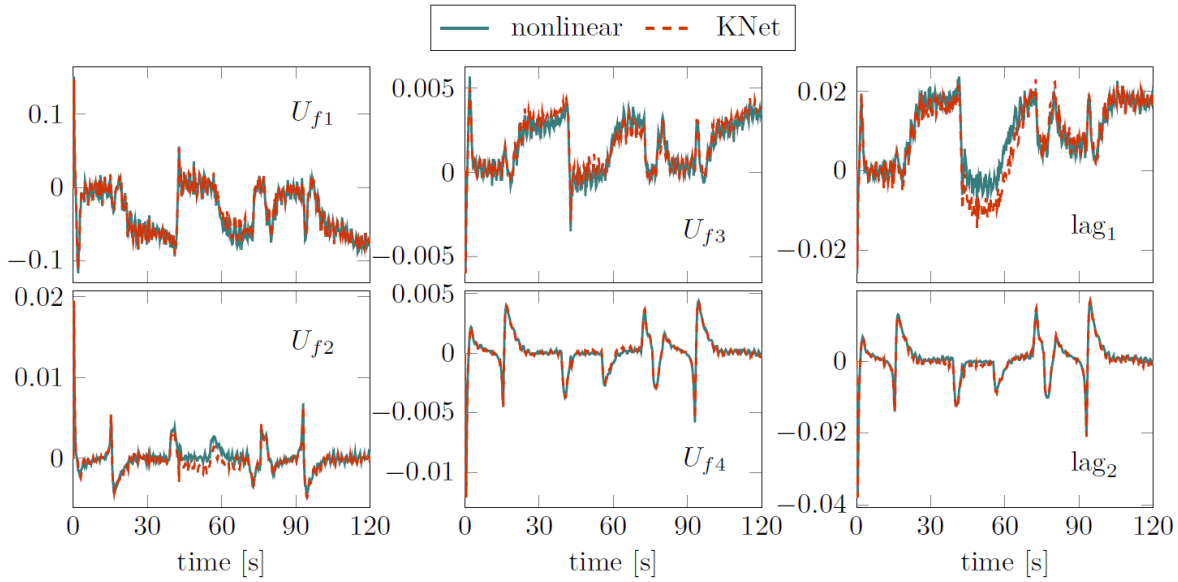


Figure 57: KalmanNet results with convolutional architecture

the following: the network manages to give similar predictions in the case of U_{f1}, U_{f3}, U_{f4} as the LPV-based filter, but performs worse on U_{f3}, lag_1 and even lag_2. However, the prediction error of lag_2 in the 8000 – 12000 interval is smaller than in the case of the linear architecture. The RMSE in this case was $1.69 \cdot 10^{-3}$ for the first four modal coordinates and the two lag states, which is somewhat bigger than the linear architecture's, but not significantly.

7.5 Data Driven vs. Model Based Estimation Conclusion

To summarize, in this work we propose a model-based and a data-driven approach to estimate the flexible dynamics of a UAV with large wingspan and highly flexible wings. The model-based approach uses an LPV-based EKF while the data-driven solution utilizes the KalmanNet architecture with 2 different neural network setups. We showed that the EKF-based estimator is able to predict the flexible and aerodynamic lag states. The neural network-based approach is also capable of estimating the above-mentioned states, however, in the case of the U_{f3}, lag_1 and lag_2 larger errors are present. Comparing the two neural network it can be concluded that both provide relatively similar accuarcy. However, al-

though the training of the *linear* network proved to be more stable, it requires almost double the training time as the *convolutional* one. It is important to mention, that the revision of the training datasets, and - generally - further research and datasets are needed to aquire better and more accurate results. Our long-term goal is to test both architectures in real-life flight data and then incorporate them in the T-Flex's FCC for real-time, airborne operations. With this, it will be possible to design a wing shape controller to minimize aerodynamic drag during flights.

8 Conclusion

The main output of the deliverable is the definition of the tools and their use within the project to analyse data generated by simulations, ground or flight tests. The models and methods are validated by comparing different methods and tools, or simulations with test data. The MDO toolchain requires FEM models and computational fluid dynamics for aerodynamics modelling what are experimentally validated for the current demonstrator configuration, and hence their validity for parametric studies are established.

Using the building blocks and flight dynamics expertise the aeroservoelastic model of the aircraft is built from analytic models what have been updated and validated using system identification methods.

The flight control laws of the demonstrator are also designed using model based tools and their HIL and flight test campaigns proved the underlying aeroservoelastic models as well as the design tools. This leads to the conclusion that parametric variation of the model within the MDO toolchain will result in valid and well performing closed-loop systems, and the results will be applicable for conceptual aircraft design.

The last step in our current investigation is to provide state estimation for wingshape control, where artificial intelligence based methods are envisioned, what can be trained with large amounts of flight test data. The performance of these machine learning based methods are validated by comparing their performance to pure LPV Extended Kalman Filtering with precise knowledge of the plant.

9 Bibliography

[1] *MATLAB - MathWorks - MATLAB & Simulink*.

[2] Gary J Balas, Andrew Packard, Peter J Seiler, and Arnar Hjartarson. *LPVTools - A toolbox for modeling, analysis and synthesis of parameter yarying control systems*. MUSYN Inc., 2015.

[3] Julius Bartasevicius, Sebastian J Koeberle, Daniel Teubl, Christian Roessler, and Mirko Hornung. Flight testing of 65kg t-flex subscale demonstrator. In *32nd Congress of the International Council of the Aeronautical Sciences*, pages 1–16. ICAS, 2021.

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

[5] A. Da Ronch, C. McFarlane, C. Beaverstock, J. Oppelstrup, M. Zhang, and A. Rizzi. Benchmarking ceasiom software to predict flight control and flying qualities of the B-747. *27th Congress of the International Council of the Aeronautical Sciences 2010, ICAS 2010*, 4(September):2906–2912, 2010.

[6] Andre Deperrois. Xflr5.

[7] André Deperrois. Guidelines for XFLR5: Analysis of foils and wings operating at low Reynolds numbers. Technical Report February, 2013.

[8] Mark Drela. Avl.

[9] Mark Drela. Xfoil.

[10] Quentin Arnaud Dugne-Hennequin, Hideaki Uchiyama, and João Paulo Silva Do Monte Lima. Understanding the behavior of data-driven inertial odometry with kinematics-mimicking deep neural network. *IEEE Access*, 9:36589–36619, 2021.

[11] Alessandro Gastaldi and Aaron Dettmann. Pytornado.

[12] R. Jategaonkar and R.V. Jategaonkar. *Flight Vehicle System Identification: A Time Domain Methodology*. Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 2006.

[13] R. Jategaonkar and R.V. Jategaonkar. *Flight Vehicle System Identification: A Time Domain Methodology*. Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 2006.

[14] Joseph Katz and Allen Plotkin. *Low-Speed Aerodynamics*. Cambridge University Press, New York, 2nd edition, 2001.

[15] David Kinney. Using VSPAERO.

[16] David Kinney. VSPAERO. . . What's New?, 2021.

[17] Thomas Klimmek. Parameterization of Topology and Geometry for the Multidisciplinary Optimization of Wing Structures. page 9, 2009.

[18] Aditya Kotikalpudi, Brian P Danowsky, David K Schmidt, Christopher D Regan, and Abhineet Gupta. Real-time shape estimation for a small flexible flying-wing aircraft. In *AIAA Scitech 2019 Forum*, page 1818, 2019.

[19] George Loubimov and Michael Kinzel. A novel approach to calculating induced drag from computational fluid dynamics. *AIP Advances*, 11(7), jul 2021.

[20] Leandro R Lustosa, Ilya Kolmanovsky, Carlos ES Cesnik, and Fabio Vetrano. Aided inertial estimation of wing shape. *Journal of Guidance, Control, and Dynamics*, 44(2):210–219, 2021.

[21] RE Maine and KW Iliff. Agard flight test techniques series. volume 3. identification of dynamic systems-applications to aircraft. part 1. the output error approach. Technical report, ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE), 1986.

[22] Brian Maskew. Program VSAERO theory document. *Nasa Cr-4023*, 1987.

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

[24] Yasser M Meddaikar, Johannes Dillinger, Thomas Klimmek, Wolf Krueger, Matthias Wuestenhagen, Thiemo M Kier, Andreas Hermanutz, Mirko Hornung, Vladyslav Rozov, Christian Breitsamter, et al. Aircraft aeroservoelastic modelling of the flexop unmanned flying demonstrator. In *AIAA scitech 2019 forum*, page 1815, 2019.

[25] Tomas Melin. A vortex lattice matlab implementation for linear aerodynamic wing applications, 12 2000.

[26] Tomas Melin. Implementation of a vortex lattice method in a heterogeneous programming language environment, 2018.

[27] J. Moran. *An Introduction to Theoretical and Computational Aerodynamics*. Dover Books on Aeronautical Engineering. Dover Publications, 2003.

[28] Erik D. Olson and Cindy W. Albertson. Aircraft high-lift aerodynamic analysis using a surface-vorticity solver. volume 0. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016.

[29] OpenVSP. Software package, version 3.22.0, 2021.

[30] W F Phillips and D O Snyder. Modern Adaptation of Prandtl's Classic Lifting-Line Theory. *JOURNAL OF AIRCRAFT*, 37(4), 2000.

[31] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[32] Muhammad K Shereen, Muhammad I Khan, Naeem Khan, and Wasi Ullah. By the design and implementation of modified kalman filter for lpv systems. *International Journal of Engineering Works*, 3(4):26–31, 2016.

[33] João Paulo Silva do Monte Lima, Hideaki Uchiyama, and Rin-ichiro Taniguchi. End-to-end learning framework for imu-based 6-dof odometry. *Sensors*, 19(17):3777, 2019.

[34] James C Sivells and Robert H Neelly. Method for calculating wing characteristics by lifting-line theory using nonlinear section lift data. *National Advisory Commitee for Aeronautics*, (865):75–93, 1947.

[35] Jurij Sodja, Roeland De Breuker, Yasser M. Meddaikar, Johannes K. Dillinger, Keith Soal, Yves Govers, Wolf Krueger, Panagiotis Georgopoulos, Christos Koimtzoglou, Christian Roessler, Sebastian J. Koeberle, Julius Bartasevicius, Daniel Teubl, Laszlo Gyulai, Szabolcs Toth, Mihaly Nagy, Daniel Balogh, Miklos Jasdi, Péter Bauer, and Balint Vanek. *Ground Testing of the FLEXOP Demonstrator Aircraft*.

[36] H.-J. Steiner. Preliminary design tool for propeller-wing aerodynamics part ii: Theory, 2010.

[37] Özge Süelözgen and Matthias Wüstenhagen. Operational modal analysis for simulated flight flutter test of an unconventional aircraft. *IFASD, The International Forum on Aeroelasticity and Structural Dynamics, 9-13 June 2019, Savannah, Georgia, USA*, 2019.

[38] Özge Süelözgen. A novel updating algorithm for linearized state-space models of an unmanned flexible aircraft using flight test data. *AIAA SCITECH 2022 Forum*, 2022.

[39] Béla Takarics and Bálint Vanek. Robust control design for the flexop demonstrator aircraft via tensor product models. *Asian Journal of Control*, 23(3):1290–1300, 2021.

[40] Béla Takarics, Bálint Vanek, Aditya Kotikalpudi, and Peter Seiler. Flight control oriented bottom-up nonlinear modeling of aeroelastic vehicles. In *2018 IEEE aerospace conference*, pages 1–10. IEEE, 2018.

[41] H. Theil. *Economic Forecasts and Policy*. Contributions to economic analysis. North-Holland Publishing Company, 1975.

[42] Matthias Wüstenhagen, Thiemo Kier, Yasser M Meddaikar, Manuel Pusch, Daniel Ossmann, and Andreas Hermanutz. Aeroservoelastic modeling and analysis of a highly flexible flutter demonstrator. In *2018 atmospheric flight mechanics conference*, page 3150, 2018.

[43] M. Wüstenhagen, Ö. Süelözgen, L. Ackermann, and J. Bartaševicius. Validation and update of an aeroservoelastic model based on flight test data. *IEEE Aerospace Conference*, 2021.

[44] Fanglin Yu, Julius Bartasevicius, and Mirko Hornung. Comparing potential flow solvers for aerodynamic characteristics estimation of the t-flex uav. 2022.

[45] Ming Zhang, Mingming Zhang, Yiming Chen, and Mingyang Li. Imu data processing for inertial aided navigation: A recurrent neural network based approach. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3992–3998. IEEE, 2021.

[46] Özge Süelözgen and Gertjan Looye. *Application and Validation of a New Updating Algorithm for Linearized State-Space Models of Flexible Aircrafts Using Flight Test Data, IFASD 2022-157*.