Cybermatrix: A Novel Approach to Computationally and Collaboration Intensive Multidisciplinary Optimization for Transport Aircraft Design

Časlav Ilić[1(✉)], Andrei Merle[1], Arno Ronzheimer[1], Mohammad Abu-Zurayk[1], Jonas Jepsen[2], Martin Leitner[3], Matthias Schulze[4], Andreas Schuster[5], Michael Petsch[6], and Sebastian Gottfried[7]

[1] Institute of Aerodynamics and Flow Technology, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany
{caslav.ilic,andrei.merle,arno.ronzheimer,mohammed.abu-zurayk}@dlr.de
[2] Institute of System Architectures in Aeronautics, German Aerospace Center (DLR), Hein-Saß-Weg 22, 21129 Hamburg, Germany
jonas.jepsen@dlr.de
[3] Institute of System Dynamics and Control, German Aerospace Center (DLR), Münchenerstr. 20, 82234 Weßling, Germany
martin.leitner@dlr.de
[4] Institute of Aeroelasticity, German Aerospace Center (DLR), Bunsenstr. 10, 37073 Göttingen, Germany
matthias.schulze@dlr.de
[5] Institute of Composite Structures and Adaptive Systems, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany
andreas.schuster@dlr.de
[6] Institute of Structures and Design, German Aerospace Center (DLR), Pfaffenwaldring 38-40, 70569 Stuttgart, Germany
michael.petsch@dlr.de
[7] Institute of Software Methods for Product Virtualization, German Aerospace Center (DLR), August-Bebel-Str. 30, 01219 Dresden, Germany
sebastian.gottfried@dlr.de

AQ1

Abstract. This paper presents an approach to multi-disciplinary optimization (MDO) of transport aircraft that attempts to strike a balance between two broad classes of MDO approaches: those arising from the formal optimization background, and those coming from the aircraft design background. It starts from the observation that any kind of numerical design process can be viewed as an approximation of a formal optimization process, where Jacobians of cost functions may be inexact and are often not explicitly computed. Based on that, a specific MDO problem representation and a highly parallel process assembly and execution protocol (the "cybermatrix" protocol) is defined, as well as one possible realization on high-performance computing (HPC) resources. The approach is applied to an optimization of a long-range transport aircraft, employing disciplinary subprocesses for high-fidelity aerodynamic design

A. Dillmann et al. (Eds.): DGLR 2018, NNFM 142, pp. 1–11, 2020.
https://doi.org/10.1007/978-3-030-25253-3_4

of wing airfoil shapes, structural sizing of lifting surfaces, and determination and evaluation of design loads.

1 Introduction

Today, research groups in the field of aircraft design are busy developing new methods which could accelerate, or even make possible at all, development of future aircraft to reach the ambitious goals stated in national and international strategies for civil air travel and transport, such as EU Flightpath 2050 [4]. It is furthermore expected that such advances will increasingly require employing physics-based simulation, and in earlier design stages than hitherto, which prompted strategies for development of numerical analysis and design capabilities on exponentially more powerful parallel computing resources, such as NASA CFD Vision 2030 [16].

The approaches that attempt to satisfy these requirements belong to two broad classes. Historically the first are the MDO approaches coming from the classic aircraft design background. They focus on process automation, rapid process assembly with many disciplines and formal data modeling [3,15], and rarely pay attention to use of HPC resources or formal optimality criteria. For example, typically there are no provisions for disciplines to exchange also first-order (Jacobian-like) information. This leads to multi-disciplinary suboptimal designs [2]. The second class are MDO approaches from the mathematical optimization background. They focus on increasing the fidelity of analysis, modeling of design constraints, and adding more disciplines [1,8]. While they explicitly consider optimality criteria and often heavily rely on HPC resources, they are poorly scalable in number of disciplines or involved experts. The employed tools tend to be simplified compared to those in industry, in order to suit the needs of formal optimization, such as computation of Jacobians.

The MDO approach presented here is developed within the DLR project VicToria and aims at making a balance between the two classes. It starts from formal optimality criteria, but applies them in a heuristic manner. It does not require that disciplines to lower their fidelity in order to fit into the overall MDO process. The achievable design is still multi-disciplinary suboptimal, but hopefully closer to optimal than in the classic aircraft design case; importantly, the approach explicitly considers approximate Jacobian-like information. Parallelism is built into the approach from the ground up, both in terms of engagement of experts and in terms of use of HPC resources. The assembly phase (human) and the execution phase (machine) employ analogous concepts of communication and control in a matrix-like structure, and are interleaved, which gives motivation for naming the approach: the *cybermatrix* protocol.

2 The Cybermatrix Protocol

The starting point is the observation that any kind of automatic design process can be viewed as an approximate optimization process, where Jacobians of

cost functions (goals and constraints) are either inexact or not even explicitly computed. Equation (1) represents one such process:

$$\widehat{\frac{df(p)}{dp}} - \lambda \widehat{\frac{dc(p)}{dp}} = 0, \quad c(p) = 0 \tag{1}$$

Here f is the goal function, c constraint functions, p design parameters, and λ optimum-to-constraint sensitivities (Lagrange multipliers in formal optimization terminology). This is, with some simplifications, the standard first-order Karush-Kuhn-Tucker (KKT) optimality condition. The hat over Jacobians df/dp and dc/dp denotes that they are approximate instead of exact.

Depending on the details of the process, the set of constraints may include not only the actual design constraints, but also internal consistency constraints, such as analysis constraints (state residuals); this further implies that the set of parameters would include not only the actual design parameters, but also internal consistency parameters, such as the analysis solutions (states). Otherwise, internal consistency constraints and parameters are solved for directly, which eliminates them from the optimization problem equation.

If there are three disciplines A, B, and C, with their assorted goal and constraint functions and design parameters, and there is a global goal function $F(f_A, f_B, f_C)$, then the KKT condition (1) can be expanded into the equation system (2), omitting $\bullet(p)$ from notation:

$$\begin{aligned}
&\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{df_A}{dp_A}} + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{df_B}{dp_A}} + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{df_C}{dp_A}} - \lambda_A \underline{\widehat{\frac{dc_A}{dp_A}}} - \lambda_B \widehat{\frac{dc_B}{dp_A}} - \lambda_C \widehat{\frac{dc_C}{dp_A}} = 0, \quad \underline{c_A} = 0 \\
&\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{df_A}{dp_B}} + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{df_B}{dp_B}} + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{df_C}{dp_B}} - \lambda_A \widehat{\frac{dc_A}{dp_B}} - \lambda_B \underline{\widehat{\frac{dc_B}{dp_B}}} - \lambda_C \widehat{\frac{dc_C}{dp_B}} = 0, \quad \underline{c_B} = 0 \\
&\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{df_A}{dp_C}} + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{df_B}{dp_C}} + \widehat{\frac{\partial F}{\partial f_C}}\underline{\widehat{\frac{df_C}{dp_C}}} - \lambda_A \widehat{\frac{dc_A}{dp_C}} - \lambda_B \widehat{\frac{dc_B}{dp_C}} - \lambda_C \underline{\widehat{\frac{dc_C}{dp_C}}} = 0, \quad \underline{c_C} = 0
\end{aligned} \tag{2}$$

By comparing Eqs. (1) and (2), it can be seen that the underlined terms in (2) on their own represent single-disciplinary design. The idea then is to "assign" each row in (2) to one single-disciplinary design process and to let it be performed almost as usual, while periodically exchanging data with other disciplines. However, the data to be exchanged includes not only *consistency* couplings (computed from p), as is usual in classic aircraft design, but also some amount of *design* couplings, i.e. approximate cross-disciplinary scaled Jacobians $(dF/df_i)(df_i/dp_j)$ and $\lambda_i dc_i/dp_j$. For example, a structural design process could provide to an aerodynamic design process the elastic deflection of the wing at a flight point, which would be a consistency coupling; but it could also provide the change in wing mass wrt. airfoil thickness at few key locations, which would be a design coupling.

The more of the design couplings exchanged and the higher their accuracy (which can be incrementally increased as well), the closer the final design will be to an actual optimum. On the other hand, applying exact design couplings also

at points far away from a good local optimum, such as in a fully gradient-based optimization process, may lead to getting stuck in a bad local optimum. Custom disciplinary design methods with heuristic design couplings, by the virtue of being problem-adapted by human experts, can be expected precisely to avoid bad local optima, at the cost of never finding an exact good local optimum but nearing it sufficiently. Thus, even if exact design couplings were available throughout, a practical MDO process would start with heuristic couplings less sensitive to local optima, and then fine-tune the solution with exact couplings. For the same reason, even a single-disciplinary gradient-based process may be tuned in a similar way [10].

Since the Eq. (2) holds for any kind of MDO process, and the terms in it are often expected to be implied by the process rather than explicitly computed, a more practical representation of it is given by an N^2-type matrix on Fig. 1. Here a block on the diagonal represents a disciplinary design processes, whereas off-diagonal blocks in the same row represent the data which that discipline takes from (rather than provides to) another discipline. At the most basic, this data includes consistency couplings; the inverted triangle in the top left corner, when present, indicates that some design couplings are included as well. The thick line that runs through the row symbolizes assignment of that row (rather than column) to one discipline.

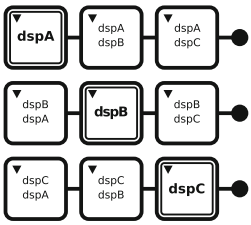

Fig. 1. Cybermatrix representation of the approximate KKT system (2).

It is important to note that the cybermatrix representation describes the solution of the problem, that which holds when the MDO process is fully converged, and not the process itself. In a worked-out system, an aircraft designer could examine a cybermatrix representation by, say, clicking on different blocks to read more details about them, and thus reason about the features of a converged solution, without having to know about the actual process which lead to it.

An actual MDO process would be specified simply by defining periods of data exchange between disciplinary subprocesses. This is equivalent to the block-lagged-solving schemes, applied often in the context of computing total gradients from Jacobians [13], whereas here they are applied to solving the overall approximate KKT system. If each of disciplines A, B, and C provides a computer implementation, in whatever form, of its iterative disciplinary design process, then the overall MDO process runs as shown on Fig. 2.

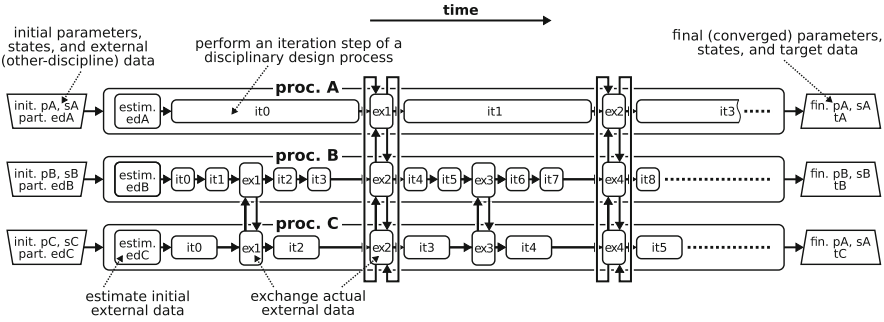

Fig. 2. Pattern of execution and data exchange between disciplinary design computational subprocesses that solves the approximate KKT system (2).

The it•-blocks represent disciplines' own iterations. The ex•-blocks are points of data exchange, which may have different frequency between different rows. The process start is special: since all disciplines start at the same time, they cannot rely on data from other disciplines being available, so they have to make their own lower-fidelity estimations of the missing data; this may however be no more than a manually prepared initial input set.

Matrix representations are eminently amenable to parallelization. In the MDO process assembly phase, disciplinary experts can talk in parallel among each other, to get their off-diagonal blocks defined and implemented. An overall process integrator in the technical sense, who would represent a serial bottleneck, is not needed at all. In the execution phase, disciplinary subproceses can run in parallel naturally as shown on Fig. 2. When problems crop up, a disciplinary group can scrutinize the full history of its subprocess execution, find a problem and either fix it or report invalid or unexpected couplings were provided by another discipline.

It is important to note that the cybermatrix protocol is not a particular optimization algorithm or problem decomposition strategy in and of itself. Compared to MDO architectures known from the literature [12], the cybermatrix protocol encompasses them all, since every MDO architecture must in the end solve a KKT system. For example, a multi-disciplinary feasible (MDF) architecture is obtained by having as the first matrix row an optimization algorithm controlling all design variables of the problem, and other rows handling only residual and cost function computation; in case of a gradient-based algorithm, the top row would also assemble the total gradient from the partial derivatives supplied by other rows. Beyond known MDO architectures, any number of "hybrid" architectures are possible, as advantageous for the problem setup or forced by the properties of disciplinary design processes. This, together with independence of a particular computer realization, is why the present approach is termed a "protocol".

3 Realization on an HPC Cluster

Together with the cybermatrix protocol, a HPC process integration framework that can drive a cybermatrix MDO process, called MDO-Driver, is currently being developed. MDO-Driver starts the disciplinary subprocesses, assigning them required computational resources on an HPC cluster, monitors their progress, triggers data exchanges, and stops on convergence.

Disciplinary experts do not work with the integration framework directly. Instead, for each off-diagonal block in the matrix, one has only to provide an *input collector*, an executable program written in any programming language whatsoever, that takes as input a path to another discipline's output data, and copies from there what it needs from that discipline. The process implementation is then just a file-system directory with one subdirectory per discipline, each of which contains a set of input collectors. This is additionaly equipped with some meta-data, such as a command which invokes MDO-Driver on the root directory of input collectors.

An MDO process defined in this way is maintainable by standard software engineering practices, such as version control. Different disciplinary experts can maintain their own input collectors in a source repository, without stepping over process files of other disciplines. MDO-Driver acts as an interpreter of this source, rather than being a tool *inside* which the process is being assembled.

MDO-Driver currently performs data exchanges over parallel on-disk file system, which is typically available on contemporary HPC clusters. However, if the file input/output performance would prove to take significant part of overall processing, MDO-Driver could be made to mount one of the extant HPC parallel in-memory file systems underneath the disciplinary subprocesses. In either case, subprocesses and input collectors do not need to be changed in any way; to them it would always appear that they are accessing a file system.

The relation between MDO-Driver and MDO frameworks such as OpenMDAO [7], GEMS [5], or KADMOS [6], is not that of a competitor, but rather of a "super-executor", which controls parallel execution and HPC resource allocation. For example, if it is desired to implement a gradient-based MDF architecture, then the actual tool behind the top row of the cybermatrix, as described in the previous section, can be OpenMDAO. The same holds for more advanced, multi-level architectures, such as those implemented by GEMS.

4 An Example Application

At present, only a preliminary example application has been performed. Here only the basic problem setup, disciplinary processes and overall convergence are presented. Detailed examination of the resulting design has not yet been carried out, pending resolving remaining issues in disciplinary tools and couplings.

The configuration in question is a typical long-range twin-engine airliner. Its CAD model is shown on Fig. 3. The goal is to optimize it for performance, as measured by mission block fuel. The goals and constraints will be mentioned

below, per disciplinary subprocess. There are three disciplines involved: aerodynamic design of airfoil shapes (**aero**), structural member sizing of wing (**struct**), and determination and evaluation of design loads for structure sizing (**loads**). The cybermatrix representation of the MDO process is given on Fig. 4.

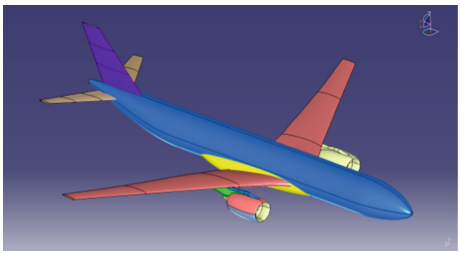

Fig. 3. Parametrized long-range twin-engine airliner in CATIA. The wing-body-tail-nacelle-pylon configuration as depicted is used for aerodynamic optimization.

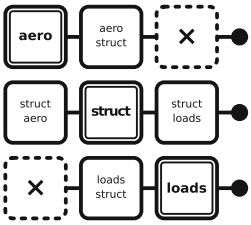

Fig. 4. Cybermatrix representation of the example problem.

There are no inverted triangles in any block, which means that currently there are no design couplings, including no couplings to the global goal function of mission block fuel; there are only consistency couplings.

Aerodynamic design of airfoil shapes is based on the FlowSimulator environment and uses an adjoint-gradient based aerodynamic optimization method. It can optimize a statically trimmed full aircraft configuration with a powered engine [14] in RANS flow; in the present work a flow-through nacelle is used. The goal is drag minimization, with aeroelastic coupling and trimming constraints handled through an internal iteration. Only aerodynamic gradient is evaluated at the static-aeroelastic equilibrium shape. The aerodynamic shape parametrization is a reduced-order model (ROM) of a parametrized CATIA V5 model (as seen on Fig. 3) mapped to the CFD mesh, so that the CAD system does not have to be used in the optimization loop. There are 126 design parameters, which are z-coordinates of b-spline control points on several wing airfoil sections. The CFD mesh is hybrid-unstructured and contains 544,000 points and 1,130,000 elements.

A single flight point is used for optimization, at Mach 0.83 and altitude of 11000 m. During optimization, updated aircraft global finite element model (GFEM) is obtained from the structural subprocess. Data exchange occurs after one gradient evaluation and the associated non-gradient line search, which takes 3.8 h on average on 48 CPU cores.

Structural sizing of wing structure is performed using the MONA structural modeling, loads analysis, and structural optimization framework [11]. It combines the in-house model generator tool ModGen, which generates simulation and optimization models, and the commercial FEA-software Nastran, which performs loads analysis (static maneuvers, solution 144) and structural optimization (solution 200). Externally provided loads can be added to those internally computed as well. The goal is mass minimization, using 368 region thicknesses (sections of upper and lower wing shell, spars and ribs) as design parameters. Constraints are structural strength limit per finite element and filtered load case, which results in about 700.000 constraint values. A constrained gradient-based optimizer is used. The GFEM consists of 18.000 FE-nodes and 42.000 FE-elements; the condensed dynamic finite-element model (DFEM) for loads analysis consist of 471 FE-nodes and 134 FE-elements (RBE2). Aerodynamic loads are evaluated using a vortex/doublet-lattice method (VLM/DLM). During optimization, updated airfoil shapes are obtained from the aerodynamic subprocess and updated external design loads (such as gust loads) from the loads subprocess. Data exchange occurs after one full structural sizing, which takes 2.7 h on 2 CPU cores.

Determination and evaluation of design loads is performed by the VarLoads framework [9]. Beyond typical quasi-steady maneuvers, VarLoads also performs transient dynamic simulations of gust and turbulence excitations as well as selected closed loop maneuvers using INDI-based flight control laws. It uses the DFEM, as well as mass cases relevant for loads calculation, obtained from an FEM generators such as MONA. Aerodynamic forces are computed with a doublet-lattice method. The model has 1068 structural degrees of freedom and 1163 aerodynamic boxes. A total of 1284 load cases and 2 mass cases (operating empty, maximum zero fuel) are considered. There are no design parameters. During optimization, updated DFEM is obtained from the structural subprocess. Data exchange occurs after one complete design loads determination and evaluation, which takes 1.2 h on 12 CPU cores.

The convergence of the MDO process is shown on Fig. 5, by each of the characteristic quantities of interest in each subprocess: drag coefficient of the whole configuration C_D, mass of the wing m_{wing}, and number of selected design load cases n_{LC}. Beside the normal optimization run, labeled "optimized", another run is shown too, labeled "baseline". In the baseline run airfoil shapes are kept constant, i.e. the aerodynamic design process is reduced to aerodynamic evaluation only; this run is performed to establish the comparison, since only the aerodynamic shape of the original design is known, but not its structural properties and loads envelope. The drag coefficient is somewhat high due to the coarse CFD mesh.

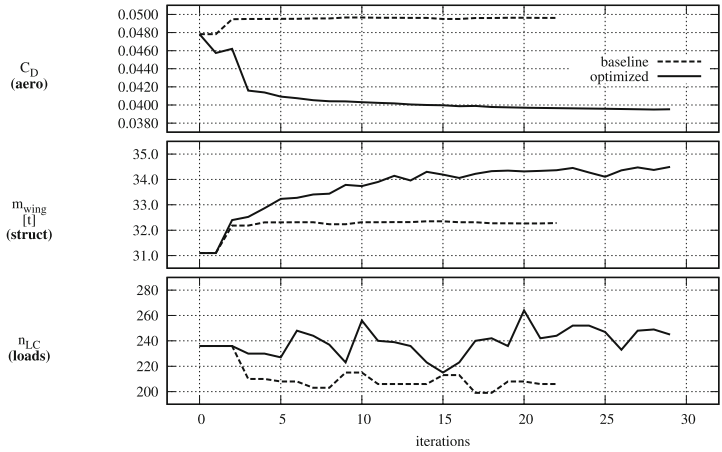

Fig. 5. Convergence history of the overall MDO process, total run time 110 h on 64 cores

The KKT-system solving perspective can be seen in the jumps of disciplinary goal functions at some iterations: since each subprocess' convergence may be affected by the data exchange between disciplines, an improvement in own goal function is not necessarily expected after every iteration, or at all. The number of design load cases is generally larger than for the baseline design, which can be expected due to tighter interdisciplinary balancing in the optimized design. Another special aspect is that n_{LC} changes through iterations inside **loads** due to changes in wing shape and structure, causing in turn changes in the number of structural failure constraints inside **struct**. This does not break the overall process, because this implementation of **struct** does not keep a constraint-related "memory" between iterations (such as an accumulated approximate Hessian) that would require a fixed number of constraints.

5 Conclusion and Outlook

The cybermatrix protocol for MDO has been described, including the overall concept and the particular realization details on HPC machines. A first example application has been presented, of a long-range airliner aero-structural optimization, which also included iterative determination of sizing load cases according to the changes in aerodynamic shape and structural stiffness.

In the ongoing work, the presented example itself will be improved by increasing the fidelity of couplings and the robustness of the subprocesses. The off-diagonal elements connecting aerodynamic and loads subprocesses, currently empty on Fig. 4, may be filled too, if the high-fidelity aerodynamic corrections are introduced into the loads process as planned in the project.

Beyond that, two disciplinary extensions of the cybermatrix process are planned by the conclusion of the project. In the first extension, a higher fidelity wing and fuselage structural design subprocess and an aircraft synthesis subprocess will be added, to have better coverage of masses, center of gravity limits, and stability considerations. In the second extension, also an overall aircraft design process will be added, to be able to change platform parameters such as aspect ratio or sweep, as well as a flutter analysis and engine design subprocesses.

References

1. Brezillon, J., Ronzheimer, A., Haar, D., Abu-Zurayk, M., Lummer, M., Krüger, W., Natterer, F.J.: Development and application of multi-disciplinary optimization capabilities based on high-fidelity methods. In: 8th AIAA Multidisciplinary Design Optimization Specialist Conference, AIAA-2012-1757 (2012)
2. Chittick, I.R., Martins, J.R.R.A.: An asymmetric suboptimization approach to aerostructural optimization. Optim. Eng. **10**, 133–152 (2009)
3. Ciampa, P., Nagel, B.: AGILE project: towards the next generation in collaborative MDO. In: 6th EASN International Conference (2016)
4. European Commission, Directorate-General for Mobility and Transport: Flightpath 2050: Europe's Vision for Aviation. Publications Office of the European Union (2011)
5. Gallard, F., Vanaret, C., Guenot, D., Gachelin, V., Lafage, R., Pauwels, B., Barjhoux, P.J., Gazaix, A.: GEMS: a Python library for automation of multidisciplinary design optimization process generation. In: AIAA SciTech 2018, AIAA 2018-0657 (2018)
6. van Gent, I., Rocca, G.L., Veldhuis, L.L.M.: Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems. In: AIAA Aviation 2017, AIAA 2017-3663 (2017)
7. Gray, J.S., Hearn, T.A., Moore, K.T., Hwang, J., Martins, J., Ning, A.: Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO. In: AIAA Aviation 2015 (2014)
8. Kenway, G.K.W., Martins, J.R.R.A.: Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. J. Aircr. **51**, 144–160 (2014)
9. Kier, T., Looye, G.: Unifying manoeuvre and gust loads analysis models. In: IFASD 2009 (2009)
10. Klimmek, T.: Parametric set-up of a structural model for FERMAT configuration for aeroelastic and loads analysis. ASD J. **3**(2), 31–49 (2014)
11. Liepelt, R., Handojo, V., Klimmek, T.: Aeroelastic analysis modelling process to predict the critical loads in an MDO environment. In: IFASD (2015)
12. Martins, J.R., Lambe, A.B.: Multidisciplinary design optimization: a survey of architectures. AIAA J. **51**(9), 2049–2075 (2013)
13. Martins, J.R.R.A., Alonso, J.J., Reuther, J.J.: High-fidelity aerostructural design optimization of a supersonic business jet. J. Aircr. **41**(3), 523–530 (2004)
14. Merle, A., Stueck, A., Rempke, A.: An adjoint-based aerodynamic shape optimization strategy for trimmed aircraft with active engines. In: AIAA Aviation 2017 (2017)

AQ2

15. Piperni, P., DeBlois, A., Henderson, R.: Development of a multilevel multidisciplinary-optimization capability for an industrial environment. AIAA J. **51**(10), 2335–2352 (2013)
16. Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., Mavriplis, D.: CFD vision 2030 study: a path to revolutionary computational aerosciences. Technical report NASA/CR-2014-218178, National Aeronautics and Space Administration (NASA) (2013)