The Cybermatrix Protocol: A link between classical aircraft design and formal multidisciplinary optimization approaches

Caslav Ilic*, Andrei Merle, Mohammad Abu-Zurayk
German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany
Email: caslav.ilic@dlr.de, andrei.merle@dlr.de, mohammed.abu-zurayk@dlr.de

Martin Leitner
German Aerospace Center (DLR), Institute of System Dynamics and Control
Münchenerstr. 20, 82234 Weßling, Germany
Email: martin.leitner@dlr.de

Matthias Schulze
German Aerospace Center (DLR), Institute of Aeroelasticity
Bunsenstr. 10, 37073 Göttingen, Germany
Email: matthias.schulze@dlr.de

Andreas Schuster
German Aerospace Center (DLR), Institute of Composite Structures and Adaptive Systems
Lilienthalplatz 7, 38108 Braunschweig, Germany, Germany
Email: andreas.schuster@dlr.de

Michael Petsch
German Aerospace Center (DLR), Institute of Structures and Design
Pfaffenwaldring 38-40, 70569 Stuttgart, Germany
Email: michael.petsch@dlr.de

Sebastian Gottfried
German Aerospace Center (DLR), Institute of Software Methods for Product Virtualization
Zwickauer Str. 46, 01069 Dresden, Germany
Email: sebastian.gottfried@dlr.de

Summary

This paper present the cybermatrix protocol, a novel approach to multidisciplinary design optimization in the contex of multiple-fidelity disciplinary analyses, many involved disciplines and high use of high-performance computing resources. The approach is presented from its formal mathematical background to actual on-disk implementation of running processes. As the demonstration case, a twin-engine long-range transport aircraft is optimized. Four disciplines are employed: overall aircraft wing planform design, aerodynamic airfoil design using 3D RANS computations, structural wing design using global shell-element FEM model, and loads selection and evaluation process based on low fidelity aerodynamics.

Keywords: *MDO, multidisciplinary optimization, HPC, high-performance computing, collaborative, large-scale, formal methods, integration frameworks, aerodynamics, loads, structure*

1 Introduction

In the field of transport aircraft design, a considerable number of groups in academia,[1] research[2,3] and industry[4] are invested in developing new approaches for designing future aircraft. A lot of the effort revolves around multidisciplinary optimization (MDO), which is perceived as the backbone, or at least a key element of these novel

design approaches.

There are two distinct directions from which this target is approached. The researchers coming from the classical aircraft design background are working on introducing formal optimization elements into their methods, but focus primarily on automatizing hitherto partly or mostly manual design workflows and enabling more effective collaboration between multitude of disciplinary experts.[2,4] The researches coming from the formal optimization background are beginning to experience the organizational issues with monolithic implementations of analysis-optimization processes, but are largely focused on discovering more efficient optimization methods for increased fidelity of physical modeling and increasingly sophisticated computational resources.[1,3]

The work described in this paper, the cybermatrix protocol, is intended to establish a practical, strong correspondence between these two directions. From the ground up, on the one hand side, it assumes use of well-tested, long-term developed disciplinary design processes within an overall aircraft design process, with disciplinary experts maintaining direct influence on their disciplinary design throughout the design effort. On the other hand side, it embeds these elements into a formal optimization context, such that not only aeronautical engineers, but also software engineers and mathematicians can be directly engaged in a design effort, adding computational infrastructure and providing hints on numerical issues, that arise during the effort. The cybermatrix protocol further enables smooth blending of disciplinary design processes employing formal optimization and custom discipline-specific methods, as well as extreme use of high-performance computational (HPC) resources. These aspects will be demonstrated in the rest of the paper, on an industry-relevant test case.

2 The Cybermatrix Protocol

Every numerical design method can be viewed as an *approximate* optimization. The method produces a solution (set of design parameters) for which the design target (goal) lies in the vicinity of the actual global optimum, for whatever criteria must be satisfied (constraints). The solution is described by the approximate first-order Karush-Kuhn-Tucker (KKT) optimality condition

$$\begin{aligned} \widehat{\frac{df(p)}{dp}} - \lambda \widehat{\frac{dc(p)}{dp}} &= 0, \\ c(p) &= 0. \end{aligned} \tag{1}$$

Here f is the goal function, c constraint functions, p design parameters, and λ optimum-to-constraint sensitivities (Lagrange multipliers in formal optimization terminology). $\widehat{df(p)/dp}$ and $\widehat{dc(p)/dp}$ are the approximate sensitivities of goal and constraints to design parameters. Constraint functions in this context may be either actual design constraints (such as maximum takeoff field length or minimum static stability margin) or analysis consistency constraints (such as residuals of RANS or linear elasticity equations). A design method might not explicitly solve equation (1) for unknown values of p and λ, nor even in any way explicitly consider it. Nevertheless, as soon as a computer implementation of the design method arises, it can be interpreted as an instance of solving such an equation.

The next step is to introduce several disciplines, let them be named A, B, and C. Consequently there are disciplinary goals f_A, f_B, f_C, constraints c_A, c_B, c_C, design parameters p_A, p_B, p_C, and optimum-to-constraint sensitivities λ_A, λ_B, λ_C. If the overall goal function F depends on disciplinary goals functions as $F(f_A, f_B, f_C)$ and therefore implicitly on disciplinary design parameters, applying equation (1) to F and using the chain rule produces

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

Equation (2) is describing the approximate solution of a multi-disciplinary design problem. Each two rows represent the single-disciplinary solution of the associated discipline (in order, A, B, and C) when embedded into a multidisciplinary context. The underlined terms describe the standalone single-disciplinary solutions, which would be produced when each discipline would be performing the design in isolation. Thus, it can instantly be seen that a disciplinary solution differs between the two contexts. The size of the difference depends on the relative size of "off-diagonal" (non-underlined) terms—the *design couplings*—to "diagonal" (underlined) terms. If all off-diagonal terms are in fact zero, then the disciplinary solutions in both contexts are the same.

For conventional aircraft configurations, the magnitudes design couplings are known in practice (when often not directly computed) and may not be too high. When it comes to unconventional configurations, such as which are being explored for future aircraft design, design couplings are by definition much less known. This implies the necessity to be able to estimate them. In formal optimization, design

couplings are always strictly computed; in classic aircraft design, the design couplings are mostly neglected.

Note that not explicitly shown in the equation (2) are the *consistency couplings*. These are any non-sensitivity-like terms that, for a given discipline, depend on design parameters of another discipline. For example, the structural deformation of the wing, significant for designing aerodynamic shape of the wing, depends both on aerodynamic and structural design parameters, and must be computed by the structural discipline for the sake of aerodynamic discipline. Getting consistency couplings correct is necessary to get a physically meaningful solution, regardless of the degree of its optimality. Thus, both in formal optimization and in classic aircraft design, consistency couplings are always well considered.

Since terms in the equation (2) are often not explicitly modelled, to merely get an overview or keep track of the elements of the design problem, it is sufficient look at an N2-like representation such as shown on figure 1. This representation is what is referred to as a "cybermatrix". Each row represents one disciplinary design embedded in the overall design. Diagonal boxes represent disciplinary designs, whereas off-diagonal boxes represent design couplings and consistency couplings needed by the given disciplinary design from other disciplines. The presence of a small inverted triangle in a box indicates that also some design couplings are taken into account, otherwise only consistency couplings are considered. Each row is run through by a line to indicate that rows (and not columns) are primary units of division of work between disciplines.

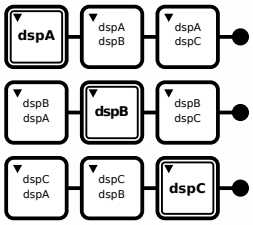

Figure 1: Cybermatrix representation of the approximate multidisciplinary KKT system (2).

In a worked-out software system, a human aircraft designer might be able to click on different boxes in a cybermatrix, examine methods and data exchange between the disciplines, and based on that reason about the design solution. Importantly, the computer implementation by which the solution is computed is completely decoupled from reasoning about physical and engineering characteristics of the solution.

For the computer implementation, one needs to assume only that each disciplinary design process is an iterative process of some sort, which takes as input also some consistency and design couplings from other disciplines. Then, each disciplinary process starts of with its own *estimate* of couplings, and periodically, after one or more iterations, gets *updates* of actually evaluated couplings by other disciplines. An example of such a scheme is depicted on figure 2. This is all that is necessary to solve the equation (2) and to compute the design solution.

As depicted on the figure, the pattern of data exchange, at which iteration which disciplinary processes exchanges data with another discipline, is not uniform and is decided per design problem. This is done based on the relative convergence characteristics, runtime and computational resource requirements of the involved disciplinary processes. For example, some disciplinary processes, if much faster than the other, may fully converge their design before each exchange of data.

This brings up the question of how the outlined implementation relates to "multidiscplinary architectures" often mentioned in literature.[5] The relation is that of generalization: depending on the selected pattern of data exchange, and on which discipline controls which design parameters, all special-case architectures can be recovered. For example, the "multidisciplinary feasible" (MDF) architecture is obtained when the discipline A controls all design parameters, takes exact consistency and design couplings from disciplines B and C, whereas disciplines B and C control no design parameters; here discipline A is in fact a gradient-based optimizer. The conjecture is, however, that any practical process will require such choices that will not represent any of the special-case architectures.

No classical algorithmic workflow figures, or more modern presentations such as XDSM,[6] are needed to represent the process. Such figures would provide no useful additional information about the process, and sometimes would not even be able to fully represent the process. Instead, a stylized space-time schematics of an *internal period* can be displayed. For the process on figure 2, the internal period is shown on figure 3. The thickness of the iteration bars in the time-space schematics can be set to represent relative resource requirements (e.g. number of processing cores) and their width the relative or absolute iteration run times.

Up to this point, there was no mention of actual software tools (process integration frameworks, scientific workflow frameworks) or actual hardware resources (single workstations, HPC clusters, area-distributed compute nodes, etc.) It is for this reason that the term *protocol* is used in naming the approach. The software and hardware support can be tailored to the particular computational and organizational environment, and even to a particular design problem within that environment.

3 An Example Overall Aircraft Optimization

The example problem is that of optimizing a typical long-range twin-engine transport aircraft. The CAD model of outer shape is shown on figure 4. The overall design goal is to minimize mission block fuel. Disciplinary goals and constraints will be stated per disciplinary subprocess in

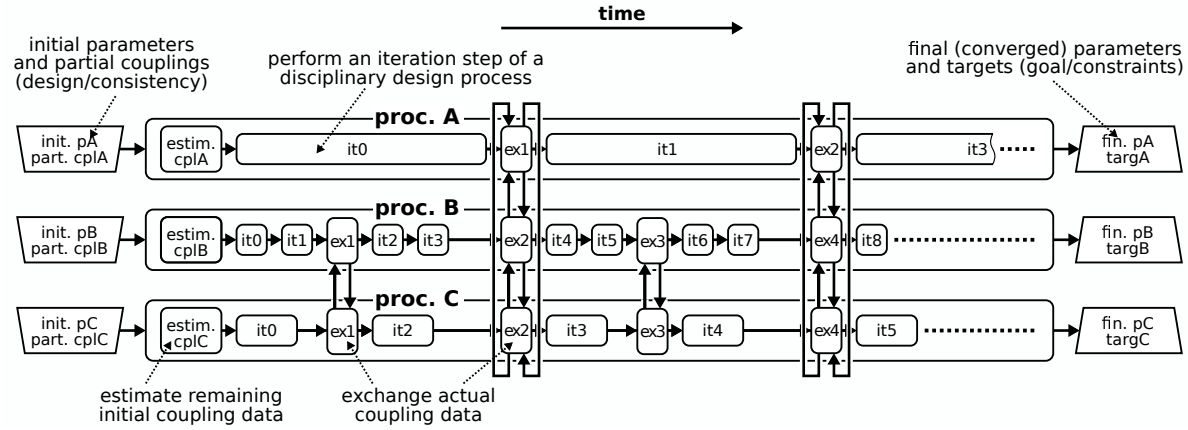

Figure 2: Pattern of execution and data exchange between disciplinary design computational subprocesses that solves the approximate multidisciplinary KKT system (2).

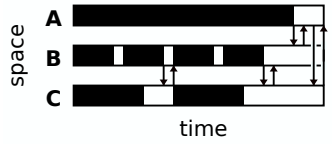

Figure 3: Schematic view of an internal period for the multidisciplinary process from figure 2.

the following. There are four disciplines involved: overall aircraft design (`oad`), 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 process is given on figure 5.

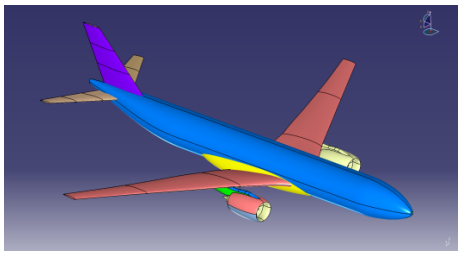

Figure 4: Parametrized long-range twin-engine transport aircraft in CATIA. The wing-body-tail-nacelle-pylon configuration as depicted is used inside the aerodynamic design disciplinary process.

3.1 *Disciplines*

Overall aircraft design (`oad`) is a custom algorithm for stepping wing planform global design variables, such as aspect ratio, sweep, taper ratios, etc. from an initial design to the optimized (figure 6). The algorithm works by making a limited parameter study around a given design. It fits a prescribed curve through 3 points for each parameter (center, negative step, and positive step) and

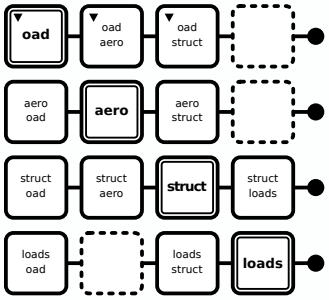

Figure 5: Cybermatrix representation of the example problem. An empty box means coupling either not present or not modeled.

determines the total design step based on the combination of the curves. From the classical aircraft design viewpoint this could be termed a “rolling trade study”; from the formal optimizatation viewpoint, it is a derivative-free quasi-Newton trust-region method, with explicit modeling of Hessian information. The goal is minimizing mission block fuel, which is evaluated from the Breguet range equation. There are no constraints on this level at the moment. Design couplings are extracted from the fitted curves. During optimization, updated lift-to-drag ratio is taken from `aero` and wing mass from `struct`. Data exchange occurs after each combined design step, that is, after all candidate designs are fully localy optimized by `aero` and `struct`.

Aerodynamic design of airfoil shapes (`aero`) employs an adjoint-gradient based aerodynamic optimization method within the FlowSimulator HPC framework.[7] It can optimize a statically trimmed full aircraft configuration with a powered engine in RANS flow, though in the present work a flow-through nacelle is used. The goal is drag minimization, where aeroelastic coupling and trimming constraints are satisfied internally. Only aerodynamic gradient is evaluated at the static-aeroelastic equilibrium.

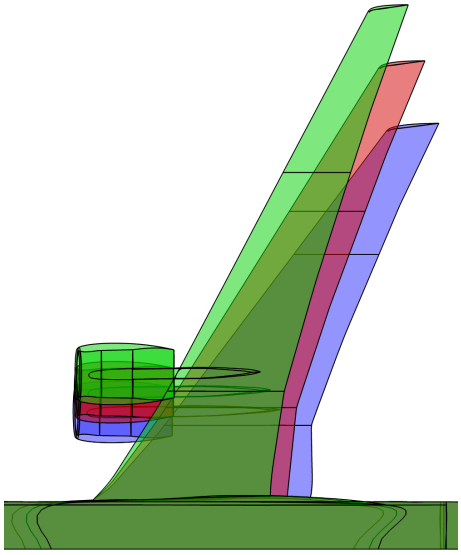

Figure 6: Planform variations for the CAD model from figure 4. Aspect ratio changes in steps of 2 and sweep in steps of 4[deg], keeping referent area constant.

The aerodynamic shape parametrization is a reduced-order model (ROM) of a parametrized CATIA V5 model (as seen on figure 4) mapped to the CFD mesh, so that the CAD system does not have to be used inside an optimization run. 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 wing planform geometry is taken from `oad` and aircraft global finite element model (GFEM) from `struct`. Data exchange with `struct` occurs after one gradient evaluation and the associated non-gradient line search, and with `oad` after full convergence.

Structural sizing of wing structure (`struct`) is performed using a structural modeling, loads analysis, and structural optimization framework MONA.[8] It combines an in-house model generator for simulation and optimization models and the commercial FEA-software Nastran for loads analysis of static maneuvers and for structural optimization. The goal is mass minimization, using 368 region thicknesses as design parameters, over sections of upper and lower wing shell, spars and ribs. Constraints are structural strength limit per finite element, which results in about 700.000 constraint values. A fully stressed design (FSD) design method is used for structural sizing. 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 wing planform geometry is taken from `oad`, updated airfoil shapes from `aero`, and updated external design loads (such as gust loads) from `loads`. Data exchange with occurs after one full structural sizing.

Determination and evaluation of design loads (`oad`) is performed by the VarLoads framework.[9] VarLoads performs quasi-steady maneuvers as well as transient dynamic simulations of gust and turbulence excitations. Some of the maneuvers may be evaluated in a closed loop, using INDI-based flight control laws. The DFEM, as well as mass cases relevant for loads calculation, are obtained from 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 wing planform geometry is taken from `oad` and updated DFEM from `struct`. Data exchange occurs after one complete design loads determination and evaluation.

3.2 Computer implementation

Each disciplinary design process is deployed in form of a batch execution tool to an HPC cluster. These tools behave such that, when executed in a given run directory, they read their input data from the `input` subdirectory, write any temporary data during a run into the `work` subdirectory, and all output data into the `output` directory. Run directories reside on a distributed parallel filesystem, reachable by all compute nodes in the cluster.

An HPC integration framework MDO Driver[10] is being developed concurrently with the work presented here, which can steer computation according to the cybermatrix protocol and with tools deployed in the outlined fashion. It was possible to carry out this work in parallel due to the system-of-equations perspective: for development it was sufficient throughout to use a plain linear system of equations as a test case.

The multidisciplinary process on disk is simply a collection of scripts, that fetch the data from output directory of one discipline into the input directory of another discipline. These scripts are named *input collectors*. For each block in the cybermatrix representation on figure 5, there exists one input collector. The expert group providing a disciplinary process also provides input collectors of its discipline, as well as one metadata file with information such as the location of the disciplinary tool and the data exchange pattern. The directory tree with input collectors is shown listing 1. MDO Driver can then be viewed as an interpreter of such a tree of input collectors.

The multidisciplinary process implemented in this way is directly amenable to treatment with standard software engineering practices. This includes being developed and maintained by multiple disciplinary experts using a distributed source-control management and versioning system and corresponding distributed collaboration infrastructure.

Listing 1: A directory tree of a multidisciplinary process implementation corresponding to cybermatrix on figure 5.

```
design_process_xyz/
    ...
    inpcoll/
        exec-oad/
            toolspec
            from-input
            from-aero
            from-struct
        exec-aero/
            toolspec
            from-input
            from-oad
            from-struct
        exec-struct/
            toolspec
            from-input
            from-oad
            from-aero
            from-loads
        exec-loads/
            toolspec
            from-input
            from-oad
            from-struct
```

The space-time schematics of the process, decided upon by specific characteristics of disciplinary processes, is given on figure 7. This architecture does not correspond to any special-case architecture from the literature.

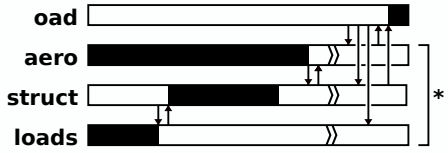

Figure 7: The space-time schematics for the cybermatrix from figure 5. * indicates that the rows are executed in multiple instances in parallel and » that multiple periods are executed before next data exchange, here for locally optimizing design candidates required by `oad`.

3.3 Results

At the time of this writing, only the optimization of the submatrix of local design parameters in `aero`, `struct`, and `loads`, has been run. Global design parameters of `oad` have been kept fixed and only the overall goal function development through iterations has been evaluated. The convergence of this optimization process is shown by figure 8. For each discipline, a characteristic quantity is shown: mass block fuel m_{fuel} for `oad`, drag coefficient C_D for `aero`, wing structural mass m_{wing} for `struct`, and number of selected load cases n_{LC} for `loads`.

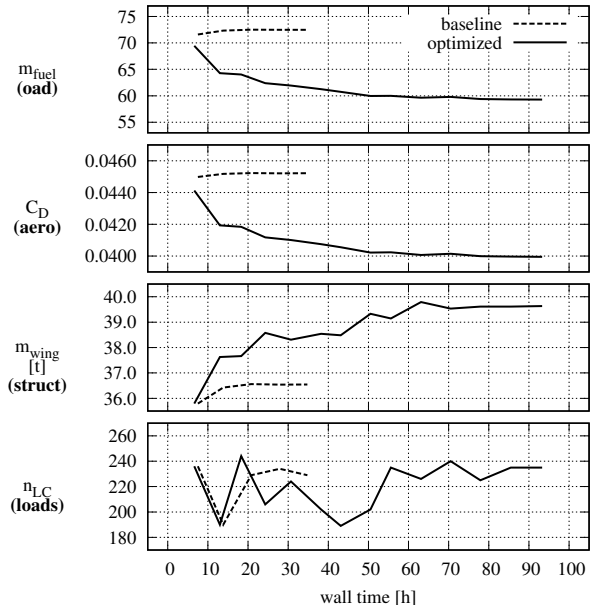

Figure 8: Convergence history of the multidisciplinary process. 72 CPU cores used at peak.

The distribution of CPU resources was 48 CPU cores for `aero`, 16 for `struct`, and 24 for `loads`. However, on figure 7 it can be seen that `struct` and `loads` never run at the same time, thus CPU cores can be reused, leading to peak 72 cores used instead of 88.

It is also not quite obvious what the baseline for comparison with optimized solution should be. To this end, another optimization is performed, where all aerodynamic shape parameters are kept fixed, that is, the complete aircraft outer shape does not change. This is "baseline" plot shown on figure 8. It can be seen that the "optimized" result saves about 10 tons of fuel, or about 14% compared to baseline. However, this is mostly caused by the very large decrease of drag coefficient, caused by using a very coarse mesh for flow analysis.

4 Conclusion

In the full paper, the missing OAD disciplinary process will be added, and a full aircraft optimization (with global and local variables together) will be performed. For this first try, OAD will most likely modify only two parameters, aspect ratio and sweep of the wing. Further quantitative details on employed disciplinary models and optimization solution post-processing will be shown. Also it may be shown how some other space-time arrangements of optimization runs compare to the one chosen here.

References

[1] Kenway, G. K. W. and Martins, J. R. R. A. Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. *Journal of Aircraft* **51** (2014).

[2] Ciampa, P. and Nagel, B. AGILE project: Towards the next generation in collaborative MDO. In *6th EASN International Conference*, (2016).

[3] Brezillon, J., Ronzheimer, A., Haar, D., Abu-Zurayk, M., Lummer, M., Krüger, W., and Natterer, F.-J. Development and application of multi-disciplinary optimization capabilities based on high-fidelity methods. In *8th AIAA Multidisciplinary Design Optimization Specialist Conference*, number AIAA-2012-1757, (2012).

[4] Piperni, P., DeBlois, A., and Henderson, R. Development of a multilevel multidisciplinary-optimization capability for an industrial environment. *AIAA Journal* **51**(10) (2013).

[5] Martins, J. R. and Lambe, A. B. Multidisciplinary design optimization: a survey of architectures. *AIAA Journal* **51**(9), 2049–2075 (2013).

[6] Lambe, A. B. and Martins, J. R. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization* **2**(46), 273–284 (2012).

[7] Merle, A., Stueck, A., and Rempke, A. An adjoint-based aerodynamic shape optimization strategy for trimmed aircraft with active engines. In *AIAA Aviation 2017*, (2017).

[8] Liepelt, R., Handojo, V., and Klimmek, T. Aeroelastic analysis modelling process to predict the critical loads in an MDO environment. In *IFASD*, (2015).

[9] Kier, T. and Looye, G. Unifying manoeuvre and gust loads analysis models. In *IFASD 2009*, (2009).

[10] Backhaus, T., Gottfried, S., Ilic, C., Merle, A., and Stück, A. Integration of high-fidelity analysis tools in MDO frameworks for HPC. In *AIAA AVIATION Forum and Exposition 2019*, (2019).