Cybermatrix Protocol: a Novel Approach to Highly Collaborative and Computationally Intensive Multidisciplinary Aircraft Optimization

Časlav Ilić, Andrei Merle, Mohammad Abu-Zurayk, Stefan Görtz
German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Braunschweig, Germany

Martin Leitner, Özge Süelözgen, Thiemo Kier
DLR, Institute of System Dynamics and Control, Weßling, Germany

Matthias Schulze, Thomas Klimmek, Christoph Kaiser, David Quero Martin
DLR, Institute of Aeroelasticity, Göttingen, Germany

Andreas Schuster
DLR, Institute of Composite Structures and Adaptive Systems, Braunschweig, Germany

Michael Petsch, Dieter Kohlgrüber
DLR, Institute of Structures and Design, Stuttgart, Germany

Jannik Häßy, Richard-Gregor Becker
DLR, Institute of Propulsion Technology, Cologne, Germany

Sebastian Gottfried
DLR, Institute of Software Methods for Product Virtualization, Dresden, Germany

Benjamin Fröhler, Johannes Hartmann
DLR, Institute of System Architectures in Aeronautics, Hamburg, Germany

Anke Tröltzsch
DLR, Institute of Software Technology, Cologne, Germany

This paper presents the cybermatrix protocol, a novel approach to multidisciplinary design optimization in the contex of many involved disciplinary experts and high use of high-performance computing resources. The approach is presented from its formal mathematical background to actual on-disk implementation of a running process. As the demonstration case, a twin-engine long-range transport aircraft is optimized. Two optimization processes are shown, an overall aircraft optimization (wing planform free) and a local parameter optimization (fixed planform), with five disciplines between them: overall aircraft wing planform design, aircraft synthesis and mission evaluation, aeroelastic wing airfoil design (RANS flow, trimmed static-aeroelastic state), maneuver loads evaluation and structural wing design (panel aerodynamics, global shell-element FEM model), and gust loads evaluation and selection (panel aerodynamics, dynamic FEM beam model). Optimized solutions are in line with those expected from previous studies under similar conditions, serving as a validation point for the new approach. Moreover, since the approach can deal flexibly with a higher-complexity loads subprocesses, such as configuration-specific selection of design load cases, also some interesting new effects are seen in optimized designs.

I. Nomenclature

f	=	objective function
c	=	constraint functions
p	=	design parameters
q	=	design influences (Lagrange multipliers)
F	=	global objective function
f	=	local objective function
R	=	residual of the state equations (physics constraints)
u	=	state variables (physics design parameters)
λ	=	adjoint state variables (physics design influences)
r_{FP}	=	mission block fuel-to-payload ratio
m_f	=	mission block fuel
m_{wing}	=	wing structural mass
C_L	=	aircraft lift coefficient
C_D	=	aircraft drag coefficient
M_x	=	bending cut-moment
M_y	=	torsional cut-moment
n_{LC}	=	number of critical gust load cases
$\bullet_A, \bullet_B, \bullet_C$	=	of discipline A, B, or C
$\bullet_a, \bullet_s$	=	of aerodynamic or structural discipline
$\bullet_L, \bullet_U, \bullet_E$	=	of lower, upper, or equality limit

II. Introduction

The field of research in aircraft design methodology has experienced a rapidly growing interest in multidiscplinary optimization (MDO) in the past decade. MDO is increasingly touted as the way to, or the key element in, design of future aircraft[1, 2]. However, this also lead to a great variety of activities being labeled MDO, where not all practitioners would agree to calling them such. There are also technical reasons for this cornucopia of opinions.

From the physical and mathematical viewpoint, MDO may take place whenever a system containing multiple *domains*, represented by different models and having a common interface, needs to have its parameters tuned to a desired effect[3]. For example, one domain may be governed by the Reynolds-averaged Navier-Stokes (RANS) partial-differential equations (PDEs) and another by the linear elasticity PDE, such as in fluid-structure coupling of elastic aircraft structures. However, the disciplinary split may also be along the lines of characteristic vehicle *subsystems*. For example, aircraft wing, tail, and fuselage design deal with the same set of physical domains, yet are often treated as different disciplines[4]. Similarly, the disciplines of engine design are not aerodynamics, structures, or acoustics, but rather fan and compressor, combustion chamber, or turbine design[5]. Finally, the act of bringing together separate automatic subsystem design processes into an automatic overall aircraft design process, by way of formalizing process interactions, defining domain ontologies, and establishing expert roles, too is often termed MDO[6].

For another axis of splitting, even the traditional design *phases* or organizational aspects may be labeled as different disciplines. Here, conceptual aircraft design is a different discipline from preliminary design, which is different from detailed design. Or, everything which is handled by one tightly-connected expert team, such as engine fan design or flight control system design, is considered one discipline, with multidisciplinarity defined as the interaction of several such teams.

Yet a third way is to differentiate by the scope of the problem. Here, MDO could be invoked as an overall aircraft design method in itself[7]. However, it could instead be put into context of more detailed design of subsystems, such as engine pylon shape design and internal structure sizing under elastic wing deformation effects[8]. MDO could also be assigned the role not in design as such, but in the preliminary exploration of advantages of new subsystem technologies, such as aggressive load alleviation or composite structure tailoring, in order to account for cumulative ("snowball") effects on the overall aircraft level[9].

Rather than arguing about definitions, the particular MDO viewpoint from which this work proceeds will be stated in some detail.

A. MDO in Present Work

The goal of this work (GOAL in the following), started within the DLR project Digital-X[10] and continued in the DLR project VicToria[11], is to establish a methodology for constructing an automatic design process which takes the top level aircraft requirements (TLARs) and a certain vehicle concept as input, and produces the complete preliminary aircraft design as output.

TLARs are the set of end-user requirements, such as number of passengers, payload, speed, range, etc. For convenience, also the relevant certification rules (such as FAR Part 25 or CS-25) and traffic categories (such as airport design codes by wing and main gear span) are included here. The vehicle concept constitutes those topological features which the automatic process itself cannot change. For example, in classical aircraft configurations, this could be high or low wing position, wing or sponson placement of landing gear, wing or tail engine placement, number of engines, selection and number of flight control surfaces, or wing spar topology. The concept could also include finer configuration aspects, such as the number of wing ribs or fuselage frames, or placement of ribs in inboard and outboard wing. The complete preliminary design encompasses all data required to begin with the detailed, or part design.

The GOAL has the following main consequences.

It is necessary to enable effective, parallel cooperation of many expert teams, in the assembly of an automatic process of high complexity. Thus, a discipline is defined to be everything for which there exists a particular expert team with their associated design process. It may be a domain-like team, such as a wing aerodynamic design team, a subsystem-like team, such as engine turbine design team, or a phase-like team, such as a conceptual design team. In effect, "multidisciplinary" becomes more accurately "many-actor", with each expert team being one actor.

It is necessary to employ physical models of multiple fidelities, from those that can be simulated within seconds on a single processor, to hours- and days-long solving of PDEs that require high-performance computing (HPC) resources. This further implies high parallelism in the execution of the automatic design process.

It must be possible to employ "clever" design methods. In the formal optimization context, there is no such thing as a design method: there is a way to evaluate quantities of interest (QIs), possibly their derivatives with respect to design parameters (DPs), and it is the duty of an optimization algorithm to find the best (optimal) design. However, this process may be susceptible to falling into a bad local optimum, taking too much time using available computational resources, or requiring further development to implement all needed elements (such as derivative evaluation). A clever design method, on the other hand, is a heuristic procedure that foregoes finding an exact optimum, but is more likely to produce a result near to a good local optimum, can execute in feasible time and is relying on existing simulation capability. The conjecture here is that no matter the state-of-the-art in formal optimization capability, producing competitive aircraft designs will always require a certain amount of clever design methods. More reasoning behind this conjecture and examples of clever design methods are given in Sec. III.D.

Employment of clever design methods furthermore introduces a bridge between the two "cultures" of MDO practitioners, those coming from the classic aircraft design background and those from the formal optimization background.

Given the complexity that an automatic design process of this kind would entail, the methodology should also enable "non-domain" experts to directly support a particular design project. This is further addressed in Sec. IV.C.

III. Design Equation

An optimal design is a solution of the set of equations called the Karush-Kuhn-Tucker (KKT) conditions[12]. In the following, they will be called *the design equation*. Works on highly collaborative MDO[6, 13] never start from the design equation, nor even mention it anywhere. It is also rarely raised in works specifically dealing with computationally intensive aspects of MDO[14]. Instead, the design equation is usually explicitly referred to in works on theory and development of optimization algorithms. However, since an equation is the most powerful method of abstraction and reasoning about a physical problem, to achieve the GOAL it is deemed necessary to start from the design equation.

The exposition of the design equation will use the formal optimization framework, but along the way links to classic aircraft design will be provided. Also, while derivatives will appear, the exposition holds for any

kind of optimization, whether gradient-based or not. Namely, every optimization algorithm solves the design equation, they just differ on how they do it, by picking a trade-off between simplicity of analysis, run time to completion, and globality of optimum. Even clever design methods implicitly solve an *approximation* of the design equation (Sec. III.D).

Let there be an objective function $f \in \mathbb{R}$ to minimize, a set of constraint functions $c \in \mathbb{R}^m$, and a set of design parameters $p \in \mathbb{R}^n$ (DPs) on which f and c depend. Evaluations of f and c produce quantities of interest (QIs). In aircraft design, f could be block fuel for highest frequency mission or (negative of) lift-to-drag ratio, while c could be maximum takeoff field length, minimum stall speed at zero altitude, or the yield stress of a structural material. A subset of c is produced by quantitative representation of TLARs, while how f fits the TLAR context is not so clear-cut; more on this in Sec. III.C. p could be wing span, wing sweep, airfoil thickness, or fuselage diameter. However, both c and p have a broader meaning, crucial in the present work, as explained in Sec. III.B.

Furthermore, let c be split into: $c_U \leq 0 \in \mathbb{R}^{m_U}$, constraints with an upper limit, such as take-off field length s_{TO} (a QI) being less than the target s^*_{TO} (a TLAR) so that $s_{TO} - s^*_{TO} \leq 0$; $c_L \geq 0 \in \mathbb{R}^{m_L}$, constraints with a lower limit, such as range with full payload $R_{P,full}$ being at least the target $R^*_{P,full}$, so that $R_{P,full} - R^*_{P,full} \geq 0$; and $c_E = 0 \in \mathbb{R}^{m_E}$, constraints with exact target, such as pitching moment in cruise M_y having to be balanced, so that $M_y = 0$. The total number of constraints is $m = m_U + m_L + m_E$.

With this in mind, the design problem, or optimization problem, is posed as

$$\min_p \; f, \tag{1a}$$

$$\text{s.t. } c_U \leq 0, \tag{1b}$$

$$c_L \geq 0, \tag{1c}$$

$$c_E = 0. \tag{1d}$$

Following[12], the solution to the design problem (1) is obtained by solving the design equation, or the KKT conditions

$$\frac{\partial f}{\partial p} - \frac{\partial c_U}{\partial p}^T q_L + \frac{\partial c_L}{\partial p}^T q_U - \frac{\partial c_E}{\partial p}^T q_E = 0, \tag{2a}$$

$$c_U \leq 0, \tag{2b}$$

$$c_L \geq 0, \tag{2c}$$

$$c_E = 0, \tag{2d}$$

$$q_U \leq 0, \tag{2e}$$

$$q_L \geq 0, \tag{2f}$$

$$q_U^T c_U = 0, \tag{2g}$$

$$q_L^T c_L = 0. \tag{2h}$$

The left hand side of (2a) is called the gradient of the Lagrangian, and it must be zero at the solution. $\partial f/\partial p \in \mathbb{R}^n$ and $\partial c_\bullet/\partial p \in \mathbb{R}^{m_\bullet, n}$ are the derivatives of objective and constraints wrt. DPs. $q_\bullet \in \mathbb{R}^{m_\bullet}$ are Lagrange multipliers, there is as many of them as constraints. The order of multiplication and transposition conforms to the chosen arrangement of $\partial c_\bullet/\partial p$ matrices, that of $m_\bullet$-rows (number of constraints) and n-columns (number of DPs). (2b–2d) simply restate conditions from the problem definition (1). (2g–2h) are called complementary conditions and state that, at the solution, each inequality constraint must either be zero if it is active, or, if inactive (and therefore non-zero), its corresponding Lagrange multiplier must be zero. The conditions (2e–2f) on Lagrange multipliers will be discussed shortly.

Unknowns in the design equation (2) are DPs p and Lagrange multipliers $q_\bullet$, for a total of $n + m$ of them. (2a) contains n equations, while (2b–2d) contain m equations, for the same total. Thus, the design equation might have a solution (if it does not, the design problem is not well posed). If some of f and c are non-linear functions of DPs, then there might be more than one solution. In that case, conditions (2e–2h) might eliminate some of the solutions. Those that remain are local design optimums, one of which is the global optimum.

From the viewpoint of classic aircraft design, Lagrange multipliers seem like an internal artifact of an optimization algorithm, not of relevance for practitioners. Optimization frameworks, that provide a uniform

interface to a collection of optimization algorithms, normally neither provide a way to set initial values of Lagrange multipliers nor a way to get their values at the end of optimization. On the other hand, from the viewpoint of formal optimization, the previous exposition is overly specific. Normally all inquality constraints are lumped together, with either $c_I \leq 0$ or $c_I \geq 0$ convention being used, and also the sign convention for constraint terms in (2a) differs between sources. The following section brings these two backgrounds together.

A. Interpretation of Lagrange Multipliers

Let there be an objective function and only one constraint, of upper limit type. The equation (2a) reads

$$\frac{\partial f}{\partial p} - \frac{\partial c_U}{\partial p}^T q_U = 0. \tag{3}$$

This can be rearranged to

$$q_U = \frac{\partial f / \partial p}{\partial c_U / \partial p} = \frac{\partial f}{\partial c_U}. \tag{4}$$

In other words, at the solution, the Lagrange multiplier shows what would be the change in the objective per unit change of constraint limit. For example, if the optimal value of mission block fuel m_f^* is found, the resulting Lagrange multiplier $q_{s_{TO}}$ associated to the takeoff field length limit s_{TO}^* is the trade-off, showing how large reduction in block fuel may be expected per unit of field length extension. The multiplier would therefore have a negative value, which is the condition (2e). This assumes that the takeoff field length constraint is actually active at the solution (it might not be, e.g. due to landing field length limit being more critical), or else the multiplier is zero, which is the condition (2g). Not only the activity of the constraint, but also the precise trade-off value may depend on other constraints, in case of non-linearity. This is summarized by the qualitative plot of Fig. 1. Another way to view this plot is as a Pareto front of m_f and s_{TO} as objectives; this perspective is addressed in Sec. III.C.

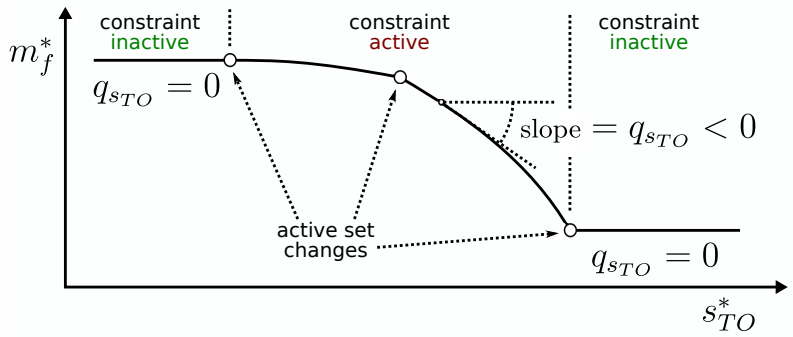


Fig. 1 Example of a Lagrange multiplier: trade-off between optimum mission block fuel m_f^* and maximum takeoff field length s_{TO}^* constraint (qualitative sketch only).

If the constraint is of the lower limit type, the analogous is the case. For example, for the lower limit on range at full payload $R_{P,full}^*$, the Lagrange multiplier shows how much block fuel on highest frequency mission can be reduced per unit range-at-full-load decrease.

This interpretation also holds in case there are multiple constraints. Showing this formally would require somewhat more derivation than just the Eq. (4), so it is skipped here. The convention for lower/upper and plus/minus signs in 2 were chosen such as to support this interpretation.

There was already at least one publication on aircraft design making this interpretation[15], including actual quantitative plots of the type of Fig. 1. In light of this, it is puzzling why Lagrange multipliers were mostly ignored so far in published works on aircraft-related optimization. The kind of quantitative information about design trade-offs, that multipliers reveal, is highly sought for by decision makers. An aircraft designer could see which of the constraints (requirements) is pushing the hardest against the objective, and focus on relaxing it by some means. This may be by changing parts of the design concept, adding more technologies (e.g. advanced materials), but also simply by adding more DPs.

In order to promote consideration of Lagrange multipliers, a specific name for the current context might be useful. For example, in economics they are called "shadow prices"[16]. For the rest of this work, the term *design influences* (DIs) is chosen. This choice implies a peer-relation with design parameters (DPs), which is motivated by them together forming solutions of the design equation (2), as well as by the concept of duality from the optimization theory. The symbols used for DPs and DIs so far, p and q, were motivated in the same way (the most common symbols in the literature are x and λ, respectively).

B. Sources of Constraints and Design Parameters

When one thinks of aircraft design in an optimization context, the first constraints that come to mind are those external to the design process, such as arising from end-user or certification requirements. The first DPs that come to mind are geometric measures, such as airfoil shape control points or structural member thicknesses. However, from the point of the design equation, other constraints and DPs appear too, such as which can be called "internal" to the process. In this work, the following classification of constraints and DPs by source is made, in order to support the discussion elsewhere in the text (such as in Sec. III.F). Since each constraint results in a DI within the design problem, the most immediately related DPs are mentioned together with constraints.

Physics constraints arise from physical modeling of problem domains. The RANS equations are one such constraint. These are often represented as the residual of the model equations being equal to zero. DPs are the domain state variables, such as density, velocity, energy and turbulent model quantities in a RANS flow.

Interfacing constraints arise at the interface of different problem domains. The principle of equal virtual work when transferring fluid load of a lifting surface to its internal structure is an example. These too are expressed as a residual equal to zero. DPs are the information transfer parameters, such as the mapping factors between a aerodynamic mesh points and structural model nodes.

Control constraints are produced by prescribing a certain motion to the aircraft. This is often called "trimming". Steady horizontal flight or a pull-up maneuver are some such motions. Constraint are formed by requiring the residual of equations of motion to be zero, that is, equality constraints. DPs include deflection of control surfaces or engine throttles.

Certification constraints are dictated by the safety considerations, codified by certification rules. The usual example is a limit on structural stress under any load within the flight envelope, as achievable or as regulated, including a safety factor. These constraints are usually inequality constraints. They are typically the first source of constraints treated as such in all optimization contexts. Counterpart DPs may be various; for the given example of stress constraints, structural region thicknesses are usually employed, but shape DPs are possible too.

Operational constraints are produced by other, higher level requirements. These include the TLARs in the narrow sense, such as maximum payload, range on maximum fuel, or take-off field length, as well as air traffic constraints, such as airport boxes, admissible cruise altitudes, and so on. They too are usually inequality constraints. DPs are any other geometric measures, from very global ones such as the shape and twist of wing planform, to more local ones, such as airfoil thickness and shape, or position of spars and ribs. However, here also *ageometric* DPs, such as wing loading or thrust-to-weight ratio, may be prominent.

C. Selection of the Objective Function

While the objective function is the most central element in the optimization theory, in classic aircraft design it might not appear at all. It is customary for TLARs to specify only a set of targets, which translate only to a set of constraints. For example, it might be required that the mission block fuel is no higher than a given value, and similarly for other QIs, rather than that any of them should be as low as possible. How should this be translated into the MDO context?

More generally, why must there be an objective function at all?

Here it is first interesting to consider the relation between the number of DPs (n) and the number of constraints m. Closer analysis of the design equation (2) reveals that the number of active constraints ($m_{active} \leq m$), i.e. those with non-zero DIs at the optimum, must be smaller or equal to the number of DPs ($m_{active} \leq n$)[12]. Intuitively, if the number of active constraints (non-zero DIs) is higher than the number of DPs, there is not enough "control" available to satisfy each of them. If this happens, the optimization problem, i.e. the design problem, is not well posed; either some constraints need to be modified, dropped, or

more DPs added. To the opposite, the objective itself cannot be dropped, as then there would be an infinite number of solutions. In clever design methods sometimes it appears that there is no objective, but it is there, implied and hidden; see Sec. III.D for examples.

Here it is conjectured that, in aircraft design, the number of active constraints is smaller than but near to the number of DPs. Recalling Sec. III.B, this includes all equality constraint types, and the rest of the degrees of freedom are mostly taken up by a selection of inequality constraints. In this situation, the choice of the objective actually determines in the first place which constraints will form the active set. For example, if a new design is started from an older design as the baseline, with improvement targets, a choice of objective might be minimal geometric difference between the old and new design (where geometrical measures are at all allowed to differ). This is called "regularization" in formal optimization theory.

It is also possible to have not one, but several QIs as objective functions. This leads to computing not a single, *point* design solution, but a *surface* of design solutions—the Pareto front. A design on the Pareto front corresponds to the composite objective of weighted objective functions, where the sum of weighting factors is equal to unity. This can be seen from the following: if one among the objectives were declared "referent", and its weight used to normalize other objectives' weights, then the normalized weights of other objectives would be equivalent to their DIs, had they been set as constraints to the referent objective. In other words, DIs form the tangent plane vector to the Pareto front[17].

Considering the GOAL, the practical answer to the question of objective selection is that the creators of the optimization process should enable setting any QI or combination of QIs as an objective, and any other QIs as constraints. The aircraft designers can then decide what objective, if none was implied by TLARs, is the most suitable for easing the process into reaching all targets. Or, if they determine there is sufficient degree of freedom remaining even after the targets are reached, they may propose to renegotiate the TLARs.

D. Clever Design Methods

When a formal optimization approach to design is not feasible at the current state-of-the-art, a clever design method may be available instead. As a rule, a clever method does not find an optimum design, but a good design, in a resonable time and using available simulation methods and computational resources. Eventually it may be replaced by formal optimization, but by that time, there will be other areas were new clever methods will appear. Since aircraft design continuously introduces new technologies, one can expect clever methods precisely in those, not yet thoroughly researched areas. Employment of clever methods is therefore crucial for producing competitive designs and the GOAL mandates the capability of including such methods into an overall aircraft optimization process.

No matter how a clever method works, the result of its run is a set of reasonable DPs for the subsystem at hand. This means that a clever method can be viewed as a solver of an *approximate* design equation, resulting in approximately optimal design. An approximate counterpart to the exact design equation (2a) has the same form

$$\widehat{\frac{\partial f}{\partial p}} - \widehat{\frac{\partial c_U}{\partial p}}^T q_L + \widehat{\frac{\partial c_L}{\partial p}}^T q_U - \widehat{\frac{\partial c_E}{\partial p}}^T q_E = 0, \tag{5a}$$

but the exact derivatives $\partial/\partial p$ are replaced with approximations $\widehat{\partial/\partial p}$. Equations (2b–2h) do not change, since a clever design method too will make sure that all constraints are upheld. This means that, so long as the actor (an expert team) behind the method is aware of the meaning of the design equation, it is in the position to figure out how to, also approximately, provide DIs and derivative-like information to other actors in an MDO context, and to take and make use of such information from other actors. Even if the actor does not provide for any of this, the overall MDO solution simply loses some of its optimality, i.e. its quality degrades gracefully.

Examples of clever design methods are:

- *Selective neglection of costly derivative evaluations* can be performed after an analysis of influence of certain derivatives on the result of optimization, for a given aircraft configuration and set of DPs. Here the approximate terms $\widehat{\partial/\partial p}$ are explicitly computed. For example, the derivative of tens of thousands of discretized structural constraints wrt. change in design loads due to change in airfoil shape DPs is costly to compute, but also not always crucial to have[18]; an alternative is to aggregate structural constraints into a few tens of composite constraints, and to compute exact derivatives of those[19]. In

the first case, an approximate derivative of exact constraints is computed, while in the second case an exact derivative of approximate (though conservatively so) constraints is computed. It depends on the design problem, available simulation tools and computing resources which approach is better suited.

- *Concatenated gradient-based structure optimizations with different constraints* may arise when the gradient-based optimizer has difficulties of handling a too diverse set of constraints and DPs. In one case of a structural sizing process[20], in the first step structural thicknesses are first sized with the usual strength and buckling failure criteria as constraints, and in the second step for maintaining required lateral control efficiency (i.e. staying well away from aileron reversal). Crucially, in the second step, the lower limit of thicknesses are their final values from the first step. This is a heuristic measure that was shown to yield feasible designs for this particular type of design problem. If the two steps were viewed as two "disciplines" (of phase-like kind from Sec. II), one could show which terms in the multidisciplinary derivative of the Lagrangian (Sec. III.E) were neglected.
- *Aerodynamic inverse design* seeks to adapt the aerodynamic shape such as to match a target pressure distribution over a lifting surface. Here the objective is to minimize the difference between the target and the current pressure distribution (e.g. measured by its square norm). The target pressure distribution implicitly represents the Lagrangian sum of the actual objective value, which might be best drag at multiple flight points, and the actual constraint values and their DIs, such as upper limit of pitching moment, lower limit of maximum lift, aerodynamic section thickness, or drag divergence Mach number. The shape update step in inverse design, such as that of the Takanashi method[21], is then an approximate derivative of the implicit Lagrangian.
- *Fully-stressed design* (FSD) of structures proceeds by updating the thicknesses of structural members according to a measure of lack or excess of feasibility in failure criteria associated to those members (such as strength and buckling limits), until updates converge to zero. It has been observed that this often yields feasible and lightweigt designs, even though mass is not at all considered by the method[22]. It is conjectured here that the update step is akin to using a diagonal-only part of the derivative of constraints, while the implicit objective is minimizing the difference between current and initial thicknesses (effectively a regularization, as mentioned in Sec. III.C). This is supported by the fact that FSD codes tend to start from manufacturing-mandated minimum thickness, i.e. from a heavily infeasible structure, and work their way to feasibility[23], rather than the other way around.
- *Load alleviation by wing root moments minimization* is performed in a flight control system (FCS) optimization process, in order to implicitly reduce mass of the wing structure[24]. It is conjectured here that the wing root moments derivative to FCS DPs is an approximation of the Lagrangian sum of structural failure criteria (strength, buckling) to FCS DPs derivatives weighted by their respective DIs. If this conjecture is correct, the DIs resulting from a gradient-based structural optimization without load alleviation will be higher than in the case when load alleviation is present. Adding FCS DPs serves precisely to reduce structural DIs, as hinted at in Sec. III.A.

E. Expansion of The Design Equation to Multiple Disciplines

To move into the multidisciplinary territory, the design equation (2) is exemplary expanded for three arbitrary disciplines. A "discipline" here may be any kind of domain-, phase-, or subsystem-like design process, as outlined in Sec. II, which can further be either a formal optimization or a clever design method (Sec. III.D)).

Let disciplines be named A, B, and C. Each discipline has its local objective f_A, f_B, and f_C (chosen according to Sec. III.C), constraints c_A, c_B, and c_C, DPs p_A, p_B, and p_C (of any of the types from Sec. III.B), and DIs q_A, q_B, and q_C (as interpreted in Sec. III.A). There is an overall objective F. In general, each discipline's objective (and constraints) depend on its own DPs and on all other disciplines' DPs, $f_\bullet = f_\bullet(p_A, p_B, p_C)$, while the global objective is taken to be a function of local objectives, $F = F(f_A, f_B, f_C)$. For simplicity of presentation, here only equality constraints are considered, dropping the lower/upper/equal-type index. Similarly, only the Lagrangian (2a) and the constraint part (2d) of the design equation are shown in the derivation. Immediatelly used is the notation of approximate terms, some of which may be exact, signalling a mix of formal optimization and clever methods.

Then, expanding F using the chain rule produces

$$\begin{aligned}
\widehat{\frac{\partial F}{\partial f_A}}\underline{\widehat{\frac{\partial f_A}{\partial p_A}}} + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{\partial f_B}{\partial p_A}} + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{\partial f_C}{\partial p_A}} - \underline{\widehat{\frac{\partial c_A}{\partial p_A}}^T q_A} - \widehat{\frac{\partial c_B}{\partial p_A}}^T q_B - \widehat{\frac{\partial c_C}{\partial p_A}}^T q_C &= 0, \\
\underline{c_A} &= 0, \\
\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{\partial f_A}{\partial p_B}} + \widehat{\frac{\partial F}{\partial f_B}}\underline{\widehat{\frac{\partial f_B}{\partial p_B}}} + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{\partial f_C}{\partial p_B}} - \widehat{\frac{\partial c_A}{\partial p_B}}^T q_A - \underline{\widehat{\frac{\partial c_B}{\partial p_B}}^T q_B} - \widehat{\frac{\partial c_C}{\partial p_B}}^T q_C &= 0, \\
\underline{c_B} &= 0, \\
\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{\partial f_A}{\partial p_C}} + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{\partial f_B}{\partial p_C}} + \widehat{\frac{\partial F}{\partial f_C}}\underline{\widehat{\frac{\partial f_C}{\partial p_C}}} - \widehat{\frac{\partial c_A}{\partial p_C}}^T q_A - \widehat{\frac{\partial c_B}{\partial p_C}}^T q_B - \underline{\widehat{\frac{\partial c_C}{\partial p_C}}^T q_C} &= 0, \\
\underline{c_C} &= 0.
\end{aligned} \tag{6}$$

Each two paired rows define a disciplinary solution (in order, of disciplines A, B, and C) when the discipline is embedded in the multidisciplinary context. The underlined terms define disciplinary solutions in the single-disciplinary context, i.e. if the discipline would be performing its design in isolation. Thus, it can be seen that disciplinary solutions differ between the single-disciplinary and the multidisciplinary context. The magnitude 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 would happen to be in fact zero, then discipline solutions in both contexts would be same.

For conventional aircraft configurations, the magnitudes of 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, design couplings are mostly neglected.

Not explicitly shown in the equation (6) are the *consistency couplings*. These are any non-derivative-like terms that, for a given discipline, depend on DPs 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 DPs, 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.

F. Analogy With The Coupled-Adjoint Method

The coupled-adjoint method refers to a gradient-based multidisciplinary optimization in which "total" derivatives (gradients) of QIs (objective and constraints) are produced by an effective procedure, where the computational cost does not depend on the number of DPs[3], but on the number of QIs. The opposite holds, for example, when computing derivatives via finite differences or complex step[25]. In this context, DPs are understood to be geometric parameters.

A classic case for coupled-adjoint method is wing aero-structural optimization[7]. The method can be briefly illustrated as follows. Let geometric DPs be aerodynamic shape parameters p_a and structural member thicknesses p_s. Let the flow state variables (density, velocity, etc.) be $u_a(p_a, p_s)$ and the structural state variables (displacements) be $u_s(p_a, p_s)$, a function of DPs. The flow solution to be computed (such as RANS or Euler) is expressed as residual equations $R_a(u_a, u_s) = 0$ and the structural solution (such as linear elasticity) through $R_s(u_a, u_s) = 0$, which are functions of state variables. Consider only a simplified case, that of minimizing drag coefficient $C_D(u_a, u_s)$, without any constraints. (More relevant objectives might be mission block fuel or aircraft empty mass, while typical constraints would be balance of forces for steady flight, structural failure criteria under multiple load cases, leading edge radius for good high lift properties, or available fuel volume.)

Using notation $p_{a,s}$ for the concatenated vector of p_a and p_s, a gradient-based optimizer needs to evaluate the total derivatives $dC_D/dp_{a,s}$ at particular points along its way to the optimal solution, where it holds $dC_D/dp_{a,s} = 0$ (in the simplified, unconstrained case). To compute these derivatives, first, the adjointed

objective $\overline{C_D}$ is formed as

$$\overline{C_D} = C_D + \lambda_a^T R_a + \lambda_s^T R_s, \tag{7}$$

where λ_a and λ_s are arbitrary vectors, called adjoint vectors. This can be done without changing the objective, since $R_a = 0$ and $R_s = 0$ holds for any $p_{a,s}$ and therefore

$$\frac{d\overline{C_D}}{dp_{a,s}} \equiv \frac{dC_D}{dp_{a,s}}. \tag{8}$$

Expanding equation (8) via chain rule produces

$$\begin{aligned}\frac{d\overline{C_D}}{dp_{a,s}} = \frac{\partial C_D}{\partial p_{a,s}} &+ \left(\frac{\partial C_D}{\partial u_a}^T \frac{du_a}{dp_{a,s}}\right)^T + \left(\frac{\partial C_D}{\partial u_s}^T \frac{du_s}{dp_{a,s}}\right)^T \\ &+ \frac{\partial R_a}{\partial p_{a,s}}^T \lambda_a + \left(\frac{\partial R_a}{\partial u_a}\frac{du_a}{dp_{a,s}}\right)^T \lambda_a + \left(\frac{\partial R_a}{\partial u_s}\frac{du_s}{dp_{a,s}}\right)^T \lambda_a \\ &+ \frac{\partial R_s}{\partial p_{a,s}}^T \lambda_s + \left(\frac{\partial R_s}{\partial u_a}\frac{du_a}{dp_{a,s}}\right)^T \lambda_s + \left(\frac{\partial R_s}{\partial u_s}\frac{du_s}{dp_{a,s}}\right)^T \lambda_s.\end{aligned} \tag{9}$$

Now the terms are grouped by $du_a/dp_{a,s}$ and $du_s/dp_{a,s}$ to get

$$\begin{aligned}\frac{d\overline{C_D}}{dp_{a,s}} = \frac{\partial C_D}{\partial p_{a,s}} &+ \frac{\partial R_a}{\partial p_{a,s}}^T \lambda_a + \frac{\partial R_s}{\partial p_{a,s}}^T \lambda_s \\ &+ \frac{du_a}{dp_{a,s}}^T \left(\frac{\partial C_D}{\partial u_a} + \frac{\partial R_a}{\partial u_a}^T \lambda_a + \frac{\partial R_s}{\partial u_a}^T \lambda_s\right) \\ &+ \frac{du_s}{dp_{a,s}}^T \left(\frac{\partial C_D}{\partial u_s} + \frac{\partial R_a}{\partial u_s}^T \lambda_a + \frac{\partial R_s}{\partial u_s}^T \lambda_s\right).\end{aligned} \tag{10}$$

It is $du_a/dp_{a,s}$ and $du_s/dp_{a,s}$ terms that would be expensive to compute, since they would need one solving of state equations per DP. However, if the adjoint vectors λ_a and λ_s are computed from the system of equations obtained by setting the bracketed terms in (10) to zero,

$$\begin{aligned}\frac{\partial C_D}{\partial u_a} + \frac{\partial R_a}{\partial u_a}^T \lambda_a + \frac{\partial R_s}{\partial u_a}^T \lambda_s = 0, \\ \frac{\partial C_D}{\partial u_s} + \frac{\partial R_a}{\partial u_s}^T \lambda_a + \frac{\partial R_s}{\partial u_s}^T \lambda_s = 0,\end{aligned} \tag{11}$$

then the expensive terms are cancelled out. The total derivative can be evaluated from what remains as

$$\frac{d\overline{C_D}}{dp_{a,s}} = \frac{\partial C_D}{\partial p_{a,s}} + \frac{\partial R_a}{\partial p_{a,s}}^T \lambda_a + \frac{\partial R_s}{\partial p_{a,s}}^T \lambda_s. \tag{12}$$

The $\partial/\partial p_{a,s}$ terms are much cheaper to evaluate, because they often do not require solving any state equations. (If some of them do, the adjoined objective (7) can be extended to include their state equations too; for example, for some kinds of aerodynamic mesh deformation[26, 27].)

A gradient-based optimizer will solve for $p_{a,s}$ to make the expression (12) equal to zero (in the simplifed, unconstrained case). Then, together with equations (11) and state residual equations, all equations to be

solved in this example of coupled-adjoint gradient-based optimization can be listed as

$$\begin{aligned}
\frac{\partial C_D}{\partial p_{a,s}} + \frac{\partial R_a}{\partial p_{a,s}}^T \lambda_a + \frac{\partial R_s}{\partial p_{a,s}}^T \lambda_s &= 0, \\
\frac{\partial C_D}{\partial u_a} + \frac{\partial R_a}{\partial u_a}^T \lambda_a + \frac{\partial R_s}{\partial u_a}^T \lambda_s &= 0, \\
R_a &= 0, \\
\frac{\partial C_D}{\partial u_s} + \frac{\partial R_a}{\partial u_s}^T \lambda_a + \frac{\partial R_s}{\partial u_s}^T \lambda_s &= 0, \\
R_s &= 0.
\end{aligned} \tag{13}$$

The first of the two key observations in this analogy is that the equation system (13) is structurally exactly the same as the multidisciplinary design equation system (6). To match them, one only needs set p_A to $p_{a,s}$, p_B to u_a, p_C to u_s, c_A to nothing, c_B to R_a, c_C to R_s, q_A to nothing, q_B to λ_a, q_C to λ_s, and $F = f_A = f_B = f_C$ to C_D. Put another way, state variables $u_\bullet$ are just another kind of DPs and state equations $R_\bullet$ are just another kind of constraints, exactly as classified (and which motivated the classification) in Sec. III.B. Conveniently, representation (13) also removes the need to talk about total derivatives. When other kinds of constraints are taken into account as well, coupled-adjoint system becomes a subsystem of the overall design equation system.

The coupled-adjoint system (11) is always solved in the following way[3, 28, 29]. It is first rearranged as

$$\begin{aligned}
\frac{\partial R_a}{\partial u_a}^T \lambda_a &= \frac{\partial C_D}{\partial u_a} + \frac{\partial R_s}{\partial u_a}^T \lambda_s, \\
\frac{\partial R_s}{\partial u_s}^T \lambda_s &= \frac{\partial C_D}{\partial u_s} + \frac{\partial R_a}{\partial u_s}^T \lambda_a.
\end{aligned} \tag{14}$$

The left-hand side terms $\partial R_a/\partial u_a$ and $\partial R_s/\partial u_s$, in full or in part, are typically already used in Newton-like methods to solve the state equations $R_a = 0$ and $R_s = 0$. This means that the left-hand sides inherit numerical properties of respective disciplines, and that, for a given right-hand side, they are most efficiently and most robustly solved by applying the same discipline-specific solvers. Thus the system is solved in a block-Jacobi or block-Gauss-Seidel ("lagged update") manner, where each equation is first solved in turn, and then the λ_a and λ_s solutions are exchanged to update the right-hand sides for the next block-iteration. This repeats until convergence.

The second key observation is this: if letting each discipline solve its part is the most efficient and most robust way to go about in the coupled-adjoint method, and if coupled-adjoint equation is a subsystem of the design equation, then why not upscale the same principle to the whole design equation?

This observation gives rise to the cybermatrix protocol.

IV. Cybermatrix Protocol

The cybermatrix protocol is an approach to MDO comprising the following three principles:

1) *Reason about the design problem directly through the design equation.*
2) *Distribute modeling and solving of design equation rows between disciplines.*
3) *Parallelize human collaboration and machine execution in an analogous manner.*

These principles are discussed in detail in sections IV.A–IV.C. The approach is termed a "protocol" because it does not, as such, establish any particular software implementation for disciplinary processes, nor use of any particular software framework or platform across each of them. In fact, as one crucial point for achieving the GOAL, the cybermatrix protocol enables linking disciplinary processes with dissimilar software backgrounds (especially seen in Sec. IV.C).

The name "cybermatrix" is derived from the third principle: treating complex human-machine interaction in analogous and intermixed ways was famously made explicit by the cross-disciplinary concept of cybernetics[30], and in the present context this analogy is played out within the matrix-like environment of the multidisciplinary design equation (6).

A. Reasoning Through Design Equation

All disciplinary experts immediately involved in solving a multidisciplinary design problem should understand the elements of the design equation as outlined in Sec. III. This includes especially the meaning of DIs (Sec. III.A), the generalized view of constraints and DPs (Sec. III.B), and the regularization role of the objective (Sec. III.C). If a disciplinary design process is an instance of a clever design method (Sec. III.D) rather than a formal optimization process, disciplinary experts in charge of it can map its elements to approximations of terms in the design equation, enabling their clever method to participate in solving of the overall problem.

This principle is entirely novel. Even MDO frameworks with a strong formal approach to setting up design problems[31–33] do not expect their users to understand the design equation. At the furthest, they stop at the point of assembly of total derivatives (Sec. III.F) for a gradient-based optimizer and, for example, leave DIs be optimizer's internal matter. MDO frameworks coming from the classic aircraft design background tacitly use an analogy of disciplinary tools as physical devices, and of MDO processes as a physical machine composed of them. Here the focus is on process assembly automation[6] and organization of collaboration[34], with MDO processes being a maze of nested and branched "converger" loops (fixed-point iterators) and requiring use of advanced graphical workflow environments[14].

Since terms of the multidisciplinary design equation (6) need not be explicitly represented (as actual vectors and matrices), to get a grasp on and keep track of the design problem it is sufficient to have an N2-like representation shown by Fig. 2. This representation is referred to as a cybermatrix of the design problem. Each row represents one disciplinary design embedded in the overall design. Diagonal boxes represent disciplinary designs, whereas off-diagonal boxes represent design and consistency couplings needed by the given disciplinary design from other disciplines. Presence of a small inverted triangle in a box indicates that some design couplings are indeed taken into account (empty triangle for basic couplings, full triangle for complete couplings, with stages in between), otherwise only consistency couplings are considered. Each row is run through by a line to point out that rows (and not columns) are primary units of division of work between disciplines. Above the rows stands the name of the process.

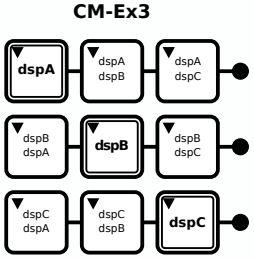


Fig. 2 Cybermatrix of the design problem, i.e. practical representation of the multidisciplinary design equation (6).

In a worked-out software system, an aircraft designer might be able to activate different boxes in a cybermatrix, examine methods and couplings exchanged between the disciplines, and based on that reason about the design solution. Importantly, the machine implementation, the computational process, by which the solution is computed is decoupled from reasoning about possible physical and engineering characteristics of the solution.

Regarding human collaboration, each row represents a disciplinary expert team. The diagonal box stands for the core team. Each off-diagonal box is assigned to at least one disciplinary team member, who needs to look into influences that the other discipline may exert, discuss this and maintain contact with the other discipline's team. This team member is also in charge of implementing the data collection from the other discipline, as shown in Sec. IV.C, and with the core team integrating this contribution into the disciplinary process.

An important property of the cybermatrix of the design problem (and cybermatrix protocol in general) towards achieving the GOAL is recursiveness. Behind one cybermatrix row (a disciplinary design) a whole sub-cybermatrix may be hidden. For example, on the aircraft level a discipline may be engine design, while within engine design, subdisciplines may be compressor, combustion chamber, and turbine design. In the

other direction, the whole of aircraft design may be one row in an airline operations design. Recursiveness plays the role of hierarchy in other MDO approaches, allowing for modular management of complexity, but without the associated degradation of design coupling. Special care is taken to maintain the recursive property in the following two principles.

B. Distributed Modeling and Solving

The multidisciplinary design equation (6) from Sec. III.E can be rearranged to mirror the coupled-adjoint method (Sec. III.F) as

$$\begin{aligned}
\widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{\partial f_A}{\partial p_A}} - \widehat{\frac{\partial c_A}{\partial p_A}}^T q_A &= \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{\partial f_B}{\partial p_A}} - \widehat{\frac{\partial c_B}{\partial p_A}}^T q_B + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{\partial f_C}{\partial p_A}} - \widehat{\frac{\partial c_C}{\partial p_A}}^T q_C, \\
c_A &= 0, \\
\widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{\partial f_B}{\partial p_B}} - \widehat{\frac{\partial c_B}{\partial p_B}}^T q_B &= \widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{\partial f_A}{\partial p_B}} - \widehat{\frac{\partial c_A}{\partial p_B}}^T q_A + \widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{\partial f_C}{\partial p_B}} - \widehat{\frac{\partial c_C}{\partial p_B}}^T q_C, \\
c_B &= 0, \\
\widehat{\frac{\partial F}{\partial f_C}}\widehat{\frac{\partial f_C}{\partial p_C}} - \widehat{\frac{\partial c_C}{\partial p_C}}^T q_C &= \widehat{\frac{\partial F}{\partial f_A}}\widehat{\frac{\partial f_A}{\partial p_C}} - \widehat{\frac{\partial c_A}{\partial p_C}}^T q_A + \widehat{\frac{\partial F}{\partial f_B}}\widehat{\frac{\partial f_B}{\partial p_C}} - \widehat{\frac{\partial c_B}{\partial p_C}}^T q_B, \\
c_C &= 0.
\end{aligned} \tag{15}$$

Compared to (15), a disciplinary design process operating in a single-disciplinary context would have zero right-hand side. In a multidisciplinary context, the right-hand sides are built up from contributions from other disciplines. If the disciplinary process is formal optimization, it is straightforward to extend it to support non-zero right-hand sides. If it is a clever design method instead, some consideration is needed, but since all right-hand side elements have a physical interpretation (Sec. III.A–III.C), a heuristic-approximate extension must be possible. Disciplinary processes supplanted in this way are run in a block-iterative fashion, until convergence.

In the event that some right-hand side terms are missing or are too coarsely approximated—which is the expected practical case—the multidisciplinary process still works, but with a degraded optimality of the converged solution. In fact, the process may even converge faster, since missing or underestimated terms increase effective system diagonal dominance (i.e. decrease design coupling). The amount of optimality degradation, i.e. recognizing to which terms more attention should be given, can be tested for numerically, as described in Sec. IV.C.

This principle can be compared, to some extent, to elements of distributed optimization approaches. In particular, in the original BLISS approach[35] a similar right-hand side correction appears. The difference however is that the present principle is not hierarchical (BL in BLISS stands for "bi-level") but level across disciplines. The question of "levels", or phase-like disciplines such as conceptual—preliminary design, is addressed in Sec. VI.

For the machine 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. Each disciplinary process, if not already capable of it, is upgraded to be able to start off using its own *estimate* of these couplings. Then, periodically, after one or more internal iterations, the disciplinary process gets *updates* of actually evaluated couplings by other disciplines. An illustration of this scheme is depicted by Fig. 3. This is all that is necessary to solve the multidisicplinary design equation, i.e. to compute the design solution.

The points at which disciplinary processes exchange data with other disciplines are not uniform. They may be different within different overall processes, or even per specific design problem treated by the overall process. The data exchange points are decided based on the relative convergence characteristics and computational run times and resource requirements of the disciplinary processes. For example, those disciplinary processes that are much fester than the other may even fully converge their design between each two exchanges.

The crux of Fig. 3 is its *base period*, the topmost repeating pattern of data exchange. The base period may be represented by a simplifed space-time diagram as in Fig. 4. On the right of execution rows, the base period diagram may also show typical resource needs (such as the number of computational cores) and run

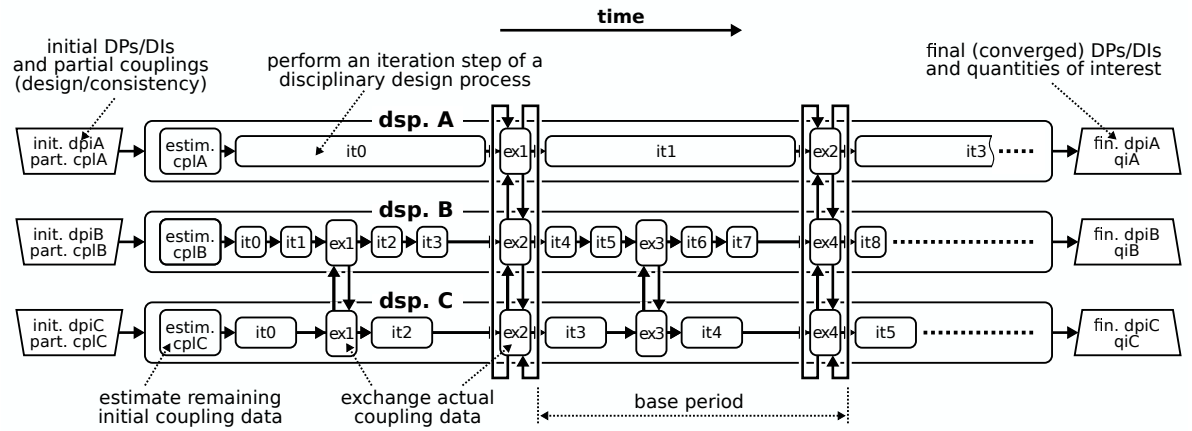


Fig. 3 One possible pattern of execution and data exchange between disciplinary design processes that implements the cybermatrix from Fig. 2.

times per disciplinary process, for a particular or typical design problem. In combination with an estimate of number of base periods to convergence, this provides a quick feeling of the overall process run time. The base period diagram may contain some more elements, which will be introduced with the demonstration cases in Sec. V. Sec. VI comments on other possible execution visualisations (such as classic flow diagrams).

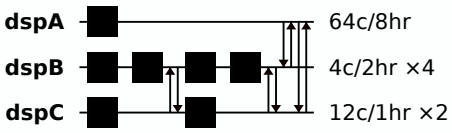


Fig. 4 One possible base period diagram for the cybermatrix from Fig. 2, a stylized form of Fig. 3.

To underline, the base period diagram is only used to present and reason about process convergence and computational resource usage. It is not necessary to understand the properties of the design solution; the cybermatrix view (Fig. 2) suffices for that purpose.

The actual software implementation is further decoupled, in the last principle.

C. Parallel Collaboration and Execution

There are many implementation environments possible for the cybermatrix of the design problem (Fig. 2) and an associated execution base period (Fig. 4). One environment in line with the GOAL is presented here.

Firstly, it assumed that each disciplinary process can be started in batch mode, on a computer with a filesystem, in a particular run directory. To start it, a single batch command, without any arguments, is executed from a path in the process deployment file tree. The process proceeds to read *everything* determining the result of computation from the `input` subdirectory of the run directory; no other data source is allowed to influence the result (such as user or system global configuration files or operating system environment variables). Input directory may contain also the computational resource assignment (such as compute node names and numbers of cores). As it runs, the process must write any temporary files it needs into the `work` subdirectory; they may not be written anywhere else (such as in operating system's default temporary storage space). All result files must be written into the `output` directory. What the process does otherwise during execution is wholly its own matter; for example, it may start a containerized environment, spring a remote-execution subframework, notify and await expert's interaction, or whatever else. The same holds for the programming language or framework of its implementation: the batch command must be self-configuring, clearing and setting upon start any software dependencies, so that it can run independently of the user's environment. This convention, including a host of finer details, has been named *sertlib* (short for "scientific

and engineering research tool library").

Secondly, it is assumed that the computer where the sertlib-compliant disciplinary process are deployed is a typical contemporary HPC cluster. Other than being a massively parallel machine, most relevantly it runs under a variant of Linux operating system, implements a job scheduler for user computations, and provides a large-capacity parallel file system accessible by all nodes. For example, the cases in Sec. V were run on a cluster composed of 2280 compute nodes (AMD Epyc 7601) having 145,920 cores in total, with about 2000 cores/user at weekly average occupancy, 15 PiB parallel file system, and SLURM[36] job scheduler. Disciplinary process maintainers are expected themselves to deploy and maintain the processes on such a machine, with the help of the cluster administration team. This includes making sure that any external communication needed by the processes, such as contacting license servers for license-limited tools within the process, functions. Since the GOAL includes preliminary design, which requires heavy use of computationally intensive tools, this kind of processing power is necessary.

With these two assumptions in hand, a cybermatrix multidisciplinary process on disk is simply a collection of *input collector* scripts. A deployed process directory tree with input collectors is shown by Fig. 5. For each box in the cybermatrix (Fig. 2), there exists one input collector, which are the `from-*` files in the figure. For the off-diagonal boxes, the input collector of discipline A from discipline B (`exec-dspA/from-dspB`) fetches data (consistency and design couplings) from the `output` directory of discipline B subprocess into the `input` directory of discipline A subprocess. For the diagonal box, there is a special input collector `from-input`, that fetches data from the top process `input` directory into discipline's `input` directory. In addition to input collectors, there is a metadata file `toolspec` (in YAML format) attached to each discipline, with information such as the location of the disciplinary process batch command and definition of its iteration and data exchange pattern.

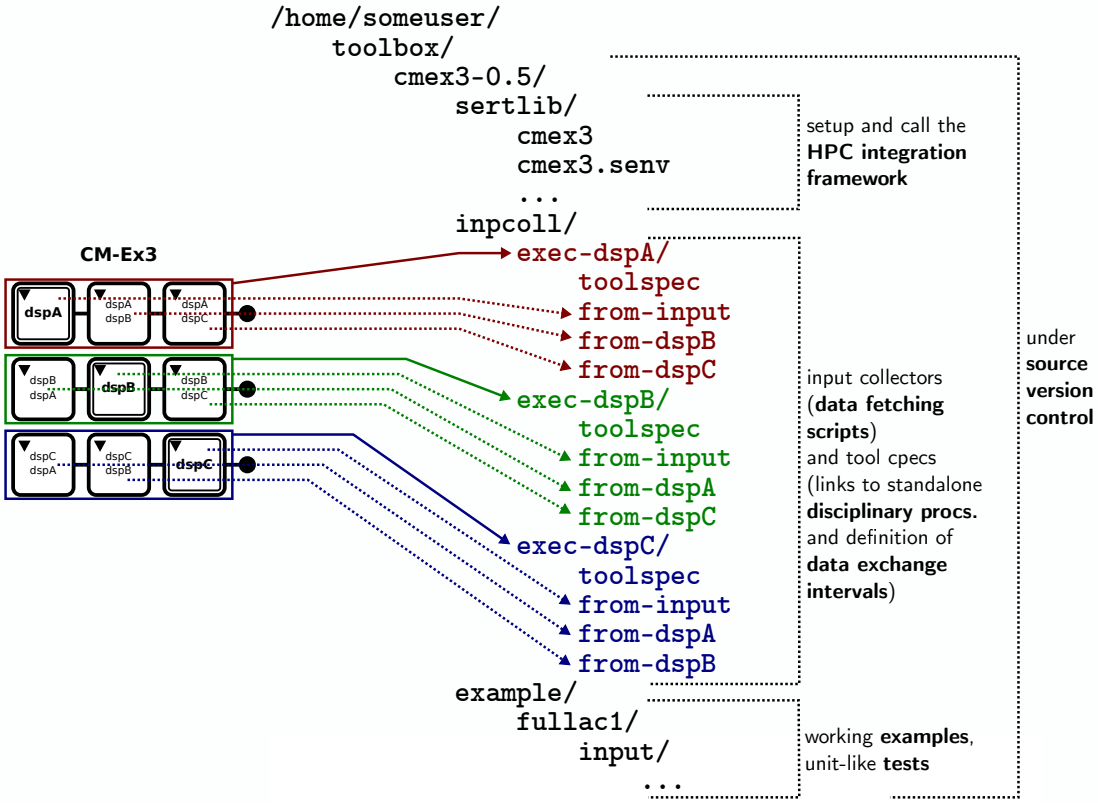


Fig. 5 A directory tree of a multidisciplinary process implementation corresponding to the cybermatrix structure.

Like disciplinary subprocesses themselves, input collector scripts of different disciplines need not use the same programming language, even within a single cybermatrix process. They only need to be executable

commands, and to accept arbitrary `output` directory path and few specific iteration indices as arguments. This means that disciplinary expert teams do not need to deal with any process integration framework software, nor do they need to use any specific application programming interface (API) to wrap their tools.

Instead, the multidisciplinary process defined in this way is *interpreted* by a cybermatrix-compliant process integration framework. The framework provides a batch command which takes as argument the process directory root, and it reads all necessary execution information from the structure of input collectors and metadata files. This batch call can be done manually, but it can also be wrapped within the process, withing its `sertlib` directory, as in Fig. 5. This means that the cybermatrix process itself becomes just a sertlib-compliant process (or, synonymously, tool), available for direct use or for integration into a super-process. Since the super-process has no knowledge of the subprocess implementation—all communication is done using the sertlib convention—it may combine several processes that use different integration frameworks (or different releases of the same framework). This maintains the recursive property mentioned in Sec. IV.A.

A cybermatrix-compliant HPC integration framework MDO Driver[37], which drove the demonstration cases in Sec. V, was developed concurrently with the work presented here. As a consequence of the first principle, reasoning through design equation, it was sufficient to use a plain linear system of equations as the single test case for developing all major functionality of MDO Driver.

MDO Driver currently relies on existence of a parallel file system, seen from all cluster compute nodes, for data exchange between disciplinary processes. This is however by no means a must. An integration framework could, for example, orchestrate data exchange among subprocesses running on different HPC clusters by temporarily copying relevant `output` directories from one machine, to the machine where the corresponding input collector needs to be run. Or, it could slip under the disciplinary processes an in-memory virtual file-system (such as Linux `/dev/shm` in the simplest form), if file-system performance is of concern. In each case, the process implementation, i.e. the input collector scripts, do not need to be modified in any way.

For best use of cluster job scheduler, i.e. for highest computation throughput, MDO Driver can execute designated disciplinary iterations as separate cluster jobs. This is in particular important for long-running overall processes composed of disciplinary subprocesses with dissimilar computational resource needs, such as the case in Sec. V.A. Whether a disciplinary subprocess iteration runs together with other disciplines in the same job, or as a separate job, is again transparent to the disciplinary subprocess and its associated input collectors.

A cybermatrix multidisciplinary process is directly amenable to treatment with standard software engineering practices. Each disciplinary team assignes their team members to implement and maintain input collectors and subprocess metadata within the cybermatrix process. As Fig. 5 shows, the file structure is tailored such that in normal circumstances it does not happen that multiple people need to edit same files. This is the best case scenario for application of distributed source-control management and versioning systems and corresponding distributed collaboration infrastructure. It furthers allow for direct use of any extant quality assurance frameworks, such as continuous integration (CI), or software certification methods in general.

Collaboration between team members of different disciplines can be done in parallel. Any two disciplines' members may discuss and implement the input collector of the corresponding off-diagonal cybermatrix box independently of other such member pairs. The timeline of collaboration on process assembly can be represented by the same kind of figure as Fig. 3, where "iteration" is replaced with "implementation" and "data exchange point" with "discussion session". In practice this will have to happen intermixed with trials of process execution, making the two phases—human assembly and machine execution—blend smoothly together. During a particular aircraft design activity, after several execution base periods, a number of "human base periods" may take place, where designers discuss running results and modify disciplinary processes (e.g. add or remove DPs, tune numerical methods, refine data exchange), followed by more execution base periods, and so on until design converges (or, more likely, a hard time deadline is hit).

Every disciplinary team must be able to run the overall process on their own, in order to be able to discover, via numerical experiments, missing or to coarsely estimated consistency and design couplings. For example, starting from a converged overall design, a disciplinary team member may manually modify and set as constant a suspect DP under that discipline's control, then run the process to converge around this point. If the reconverged overall design improves non-negligibly, that means that an examination of discipline's couplings to other disciplines is warranted. This is made trivially possible by a combination of availability of the process from a source-control management repository and the self-configuring nature of its sertlib

compliance.

If disciplinary teams are called "subject domain experts", "non-domain experts" would be experts in numerical methods, optimization algorithms, software engineering, HPC implementations, or computer networking. Traditionally, experts from these fields work independently on missing pieces between and in parallel to running design projects, so that their output may be used in future projects. In the cybermatrix protocol, they can participate directly in a design activity, by only having an understanding of the three principles. For example, a numerical methods expert may observe convergence behavior of the important QIs and advise disciplinary teams on introducing relaxation factors; a computer networking expert may arrange for remote execution of a particular disciplinary process (without any changes to the process implementation, other than an entry in a metadata file); a HPC expert may observe resource usage and advise on job scheduling strategies.

V. Demonstration Cases

Demonstration cases target design of a long-range twin-engine airliner. The baseline outer shape is that of XRF-1 (eXternal Research Forum), an Airbus-provided research aircraft configuration representing a typical long-range wide-body aircraft. For the purposes of current work, the shape was reparametrized in CATIA with certain simplifications, such as a less detailed planform (especially in the wing tip and root area) and omission of flap track fairings. The reparametrized CAD model is shown by Fig. 6. It comprises wing, body (including belly fairing), horizontal and vertical tail, engine pylons, and flow-through nacelles.

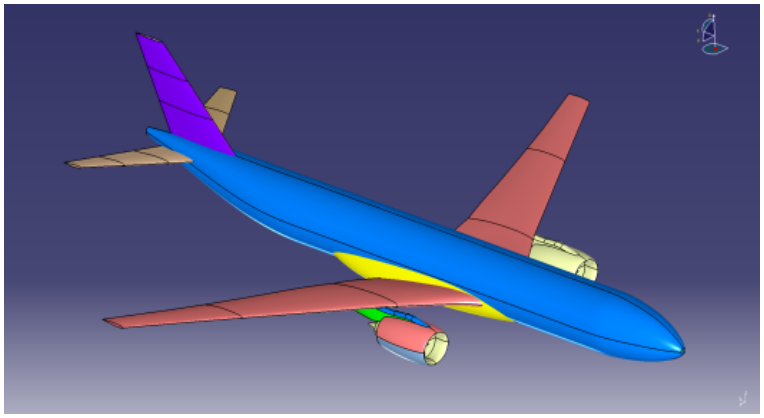


Fig. 6 Parametric CAD model of a long-range twin-engine airliner in CATIA, derived from Airbus XRF-1. The configuration as depicted is used inside disciplinary design processes.

The CAD model is fully parametric, with shape parameters for wing planform, twist, airfoils, belly fairing, horizontal and vertical tail planform and airfoils, pylon, and nacelle. Which of these are let free as DPs in optimization will be stated with respective case.

However, due to licensing costs and unavailability of CATIA on Linux, it was decided to construct a shape reduced-order model (ROM) out of the CAD model. This is named *CAD-ROM* [38] and is done in the offline mode, on a standalone Windows workstation. Computing the design-of-experiment (DOE) for constructing a CAD-ROM, depending on the selection of DPs, may last from several hours to a few days. Each sample input is a set of DPs drawn from a quasi-random sequence, while output is a set of "helper mesh" points, as shown by Fig. 7. The helper mesh is a fine structured surface mesh defined to follow tightly model shape changes, without a direct relationship to any of the disciplines. Instead, it is used as a medium to interpolate surface deformations of actual disciplinary meshes and models, between the baseline and perturbed set of DPs. In the case of aerodynamic disicpline, CAD-ROM is used to adapt the surface CFD mesh, while a linear elasticity analogy is used to propagate the deformation into the CFD mesh volume. For structural disciplines, CAD-ROM is used to adapt aircraft component cross-sections (about 10 to 30 sections per component), which are used to generate FE models of those components. So far it has proven not straightforward to create an accurate and robust CAD-ROM, especially in the context of deforming the CFD mesh.

For all cases a performance overall design objective is used, that of minimizing mission block fuel. The

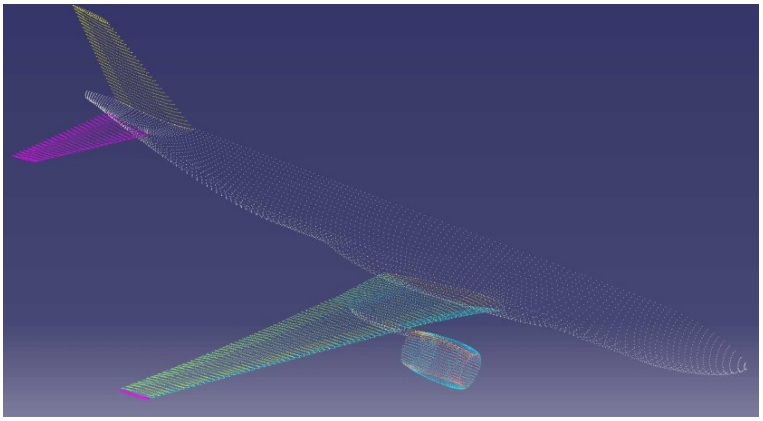


Fig. 7 A cloud of points from the helper mesh which adapts to CAD geometry changes and serves for interpolation of disciplinary meshes.

design Mach number is 0.83, for a mission with 10500 [km] range and 40 [t] payload. Disciplinary objectives, constraints and DPs will be stated in the following, per disciplinary subprocess. Engine size and performance is kept fixed.

The "baseline" design with which optimized designs are to be compared is not straightforward to define. Only a limited set of XRF-1 TLARs is known, both its outer shape and internal structure have been adapted and reparametrized, and a detailed set of design constraints defined, for the purposes of this work. This means that any initial DP values represent in part infeasible, and in part overly feasible aircraft. Therefore, without a lot to say for it, the baseline is taken to be a design where geometric DPs (outer shape) are set equal to the initial values and held constant, while other DPs (structure DPs, control DPs) are optimized by the same process used to optimize with all DPs; this yields at least a fully feasible baseline design.

A better comparison would be between two optimized designs, one arising from a process which neglects most of the design couplings ("sequential optimization"[39], akin to the classic aircraft design) and the other from a process which includes a number of significant design couplings. This would show the potential that MDO can bring to design. Such a comparison was not yet done because the processes presented here are still in a too early state of development to posses a significant amount of design couplings that could be switched on or off.

Recalling Sec. III.B, in the following only the constraints and associated DPs which are customary considered as such (control, certification and operational) are mentioned, while more basic constraints and DPs (physics and interfacing) are left implied by description of employed numerical methods.

A. Overall Aircraft Design

The cybermatrix of the overall aircraft design problem is shown by Fig. 8. Some off-diagonal boxes being empty means that the discipline of that row does not draw any couplings from the respective other discipline; this may be because physically there are none, or (as in this case) because they still were not modeled. An empty triangle in a box indicates presence of some, but not full design couplings. Elements of the cybermatrix are described in Listing 1.

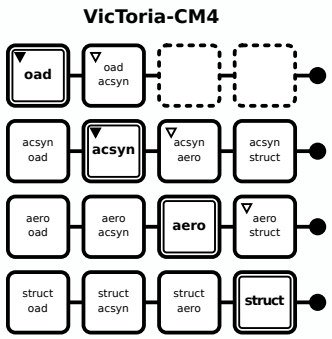


Fig. 8 Cybermatrix representation of the overall aircraft design.

Listing 1 Disciplines and couplings of the cybermatrix from Fig. 8

oad, *overall aircraft design*

The role of overall aircraft design is performed by a derivative-free constrained optimizer SQPDFO[40]. There are 2 DPs, wing aspect ratio and wing leading edge sweep angle. Wing reference area is a CAD parameter and is being kept constant. Several planform variations are shown by Fig. 9. There are no explict constraints. The objective is to minimize mission block fuel. Although the optimizer can handle non-linear constraints, at current state of the work none were defined. SQPDFO implements a trust-region sequential quadratic programming (SQP) approach, with a stepping strategy to maintain a well-poised set of design evaluations for estimating Jacobian and Hessian information. The designer must set the initial axis step for each DP, which must be well outside of the noise scale of design evaluations; it can be even of the same order of magnitude as the parameter min-max span. After making a design step, the optimizer is set to request two perturbed design evaluations per DP around the new design, resulting in 5 parallel design evaluations (5 different wing planforms) per step. From the classic aircraft design viewpoint, SQPDFO's stepping strategy could be termed a "rolling trade study".

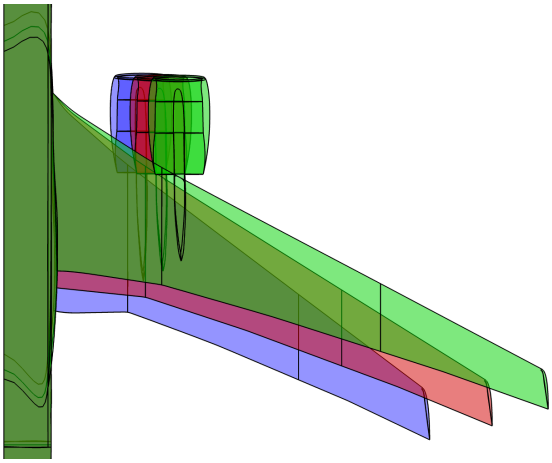


Fig. 9 Wing planform variations with changes of aspect ratio in steps of 2 and of sweep in steps of -4 [deg]. Wing reference area kept constant.

While the optimizer here is operating on raw data from other disciplines, ideally, to speed up convergence and reduce computational resource usage, it would be applied to a domain-specific multi-fidelity method taking corrections from other disciplines. This is further discussed in Sec VI. As a compromise, a fast conceptual design process OpenAD[41] was used to compute a low-fidelity DOE of approximately similar overall aircraft design problem, from which an offline ROM of relevant QIs was constructed and used to tune the options of SQPDFO. The target was to minimize the number of

sequential design steps, i.e. the overall process wall run time, by trading off with increased number of parallel perturbed design evaluations per design step. Once the options have been tuned in this way, they were not touched any more for the high-fidelity optimizations presented here.

oad←`input`

Initial values and steps of DPs.

By virtue of the local objective being the same as the global objective, the design coupling between local and global objective is exact.

oad←**acsyn**

Mission block fuel, used directly as the optimization objective.

Design coupling with **acsyn** is present to some extent, because the optimizer builds Jacobian information from multiple design evaluations, but not fully, since the DIs of **acsyn** physics constraints (state equation) are omitted.

acsyn, *aircraft synthesis*

As aircraft synthesis a simple textbook calculation was used, made specifically for this MDO process as a minimum "closure" for the fact that overall aircraft design takes place. (In the future a thorough synthesis process should take this place, for example based on the same tools that were used to derive the ROMs for tuning **oad** optimizer options.) It evaluates the mission using a fixed-point iteration on the Breguet range equation, where the payload and range are fixed, and the block fuel-to-payload mass ratio r_{FP} is the iterated unknown (i.e. state variable). Based on the result, in the mass accounting step, it produces the total cruise mass and maximum take-off mass.

Mission payload and range are selected according to the expected highest-frequency mission; this is normally neither maximum payload (but a high fraction of it, e.g. 0.80) nor the maximum range. Maximum take-off mass is computed by summing operating empty mass, a fixed fraction of maximum payload mass (e.g. 0.30), and maximum fuel mass which can fit into tanks.

acsyn←`input`

Mission definition (range, Mach number, cruise altitude, etc). Maximum payload and payload at maximum fuel load. Thrust-specific fuel consumption. Oswald span efficiency factor. Initial wing reference area, initial wing aspect ratio, initial operating empty mass, initial maximum fuel mass, initial fuel-to-payload ratio.

Since **acsyn** computes directly the mission block fuel that is the global objective (and used in **oad**), this is trivially exact coupling with the global objective.

acsyn←**oad**

Wing reference area (in case **oad** happens to modify it, which is not the case in the current process) and wing aspect ratio. The latter is used to estimate a design coupling from **aero**.

acsyn←**aero**

Lift and drag coefficient at the cruise flight point.

The lift cofficient will change in the fixed-point iteration over Breguet equation. Before the iteration, a zero-lift drag coefficient is computed from the aspect ratio and Oswald factor assuming the parabolic drag polar. Within the iteration, the polar is reused to recompute the drag coefficient. This makes for a partial design coupling between **aero** and **acsyn**, in the form of $\partial C_D/\partial r_{FP}$. The estimate speeds up convergence: in the final state the lift and drag coefficient as seen by **acsyn** and **aero** will match.

acsyn←**struct**

Operating empty mass, maximum fuel mass (based on volume of fuel tanks).

aero, *aerodynamic airfoil design*

An adjoint gradient-based aeroelastic optimization process, implemented within the FlowSimulator HPC framework [42]. It can optimize an elastic, statically trimmed full aircraft configuration in RANS flow, with flow through nacelle or a powered engine; in the present work a flow-through nacelle is used. The objective is drag minimization, with trimming constraints and aeroelastic coupling satisfied through a fixed-point loop. An aeroelastic derivative is evaluated after the trimmed static-aeroelastic equilibrium is reached. For shape parametrization a CAD-ROM is used, so that the CAD system does not have to be used during the optimization run.

In the current process, there are 126 airfoil shape DPs, the z-coordinates of b-spline control points of 7 wing sections across the whole span. The CFD mesh is hybrid-unstructured and contains 5,900,000 points (Fig. 10). For the present demonstration a single cruise flight point was used, to limit the computing resource usage, though a multi-point capability is implemented as well.

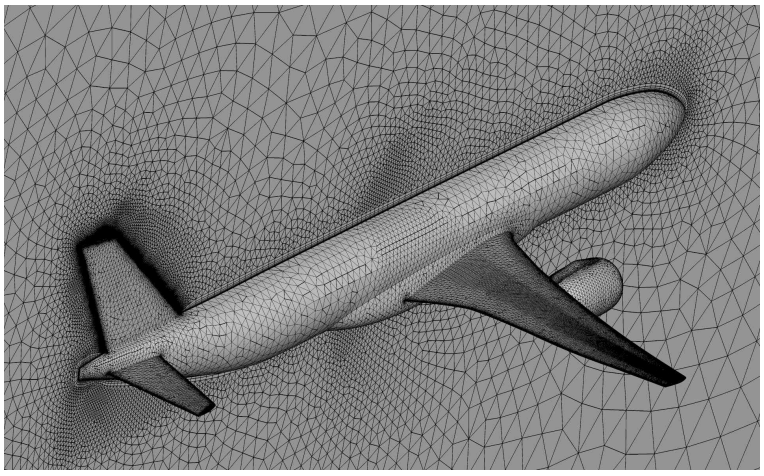


Fig. 10 CFD mesh of the full aircraft. Hybrid-unstructured, flow-through nacelles.

aero←`input`
CAD-ROM data. Initial CFD mesh, initial FE model, initial trimming values, initial component cuts (for structural wing/tail box outer shapes), initial free stream conditions.

aero←oad
Wing reference area (not modified at the moment), wing aspect ratio, wing sweep.

aero←acsyn
Referent free stream conditions (Mach, pressure, density).

aero←struct
Global FE model of the full aircraft. Aircraft mass and center of gravity at cruise point.

Since the FE model contains the stiffness matrix which is used in aeroelastic adjoint computation, to take into account the effect of elasticity on the derivative of aerodynamic QIs, there is a partial design coupling to **struct**. It is not complete, because structural constraints are ignored.

struct, *design loads and structural sizing*
Computation of design loads (static maneuvers and landing) and sizing of lifting surface structures is performed by the aeroelastic structural design process cpacs-MONA [43]. It combines a model generator for simulation and optimization models with the commercial FEA-software NASTRAN for loads analysis of static maneuvers and structural optimization. Loads analysis is performed via vortex/doublet-lattice method (VLM/DLM), with possibility of high-fidelity corretions from Euler or RANS solutions (the correction functionality was not used in current work). The objective is mass minimization, using design region thicknesses as DPs, over sections of upper and lower lifting surface shell, spars and ribs. Strutural sizing is performed with a gradient-based optimizer in this work, but a fully-stressed design (FSD) method can be selected as well. Constraints are structural strength and buckling limits, per finite element and load case, resulting in hundreds of thousands of constraints under typical conditions.

A structural global FE model (GFEM) used in this work is shown in Fig. 11. Wing and tails are modeled with beam and shell elements, while the fuselage is condensed to a beam. There are 392 region thickness DPs. Masses are condensed to point loads along load reference axes. In addition to the GFEM for sizing, a dynamic FE model (DFEM) and an aerodynamic panel model for loads evaluation are generated as well. There are 20 preselected load cases used throughout the MDO process run. They were selected from the total of 1080 load cases, based on an offline DOE of varying planforms[44]. Constraints are strength and buckling, per finite element and design region, about 700,000 in total. The GFEM consists of 18.000 FE-nodes and 42.000 FE-elements; the DFEM consist of 471 FE-nodes and 134

FE-elements (RBE2).

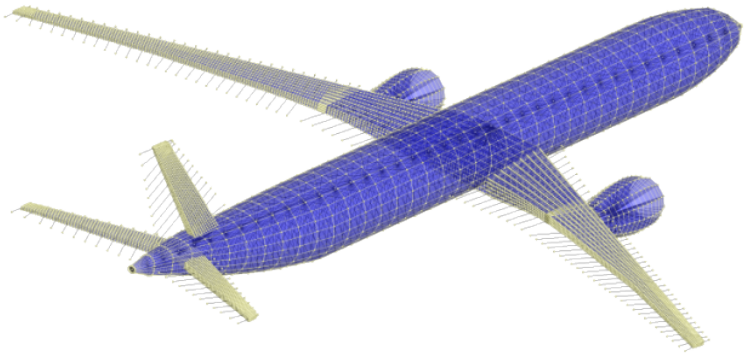


Fig. 11 Global FE model of the full aircraft. Beam and shell elements for lifting surfaces, condensed beam for the fuselage.

struct←`input`
: Definition of mass and load cases. Initial component cuts, initial wing reference area, initial maximum take-off mass, initial mission payload mass, initial fuel mass at cruise point.

struct←**oad**
: Wing reference area (not modified at the moment), wing aspect ratio, wing sweep.

  These datums are fetched only to possibly (depending on the base period setup) speed up convergence and increase robustness at initial iterations after **oad** issues new designs, since they are also implicitly acquired through component cuts from **aero**, after the first exchange with it.

struct←**acsyn**
: Maximum take-off mass, maximum payload mass, mission payload mass, fuel mass at cruise point.

struct←**aero**
: Component cuts.

The execution base period of the process is shown by Fig. 12. The multiplexer/demultiplexer symbol ⌓ in data exchange between **oad** and other disciplines indicates that **oad** issues multiple data sets (planform parameter sets) per iteration, so that all other discipline tracks need to be multiplied once per set; afterwards, **oad** gathers the results from multiplied tracks. Discipline track represented by a thick line means that disciplinary process stays in memory and sleeps between its two iterations. This is here the case for **oad**, which keeps running and sleeping during whole optimization process (it runs on the cluster frontend), as well as for pairs of **struct** iterations between which there is no data exchange with other disciplines. The dashed line of a discipline track means that the disciplinary process exits and serializes to disk (necessary parts of) its internal state, to be started again and state deserialized in the next iteration. This enables, for example, iterations to be submitted as separate cluster jobs. The track switch indicator ⍒ shows that a multiplexed track with a given multiplicity index does not have to continue (recover serialized state) from the track with the same index in the previous base period. Instead, **oad** issues also a pairing, stating which planform from the previous base period was nearest to the given planform in the current base period, to be continued from (e.g. taking previous airfoil shapes and structural thicknesses). In the resource usage labels on the right, the `sj` indicator appears, stating that iterations are executed as separate cluster jobs. The resource label for **struct** means that each two iteration pairs are executed within single separate cluster job, for two such jobs per base period. Listing 2 describes what each discipline is doing in its single iteration.

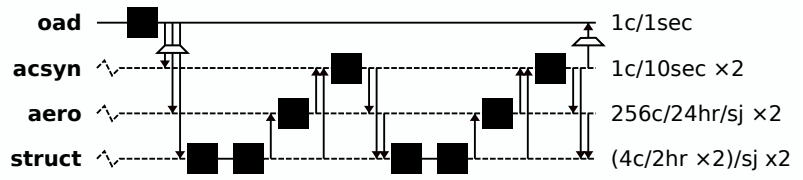


Fig. 12 Selected base period for the overall aircraft design.

Listing 2 Single iteration of a discipline in the base period from Fig. 12

oad, *overall aircraft design*
A step in DPs in the direction determined by a local quadratic approximation using derivative-free estimated Jacobian and Hessian information, accompanied by a multiple sets of perturbed DPs for updating Jacobian.

acsyn, *aircraft synthesis*
Full fixed-point convergence over block fuel-to-payload mass ratio, including single-point drag polar estimation, followed by aircraft mass accounting.

aero, *aerodynamic airfoil design*
Evaluation of the static-aeroelastic solution and adjoint for trimming constraints using current airfoil shape DPs, followed by a line search in the steepest descent direction (based on the total gradient of drag coefficient) to obtain update of DPs. Line search is derivative-free, typically two to four new static-aeroelastic solutions are evaluated.

struct, *design loads and structural sizing*
One full evaluation of preselected loads using structure based on current set of thickness DPs, followed by one fully converged constrained gradient-based optimization at fixed loads to obtain the update of DPs.

This setup of the base period was mostly driven by an unusual culprit: limited availability of NASTRAN licenses. Measuring by ratio of its cumulative runtime to iteration's total run time, NASTRAN is used heavily in **struct** and lightly in **aero**. The bare minimum of **struct**, **aero**, and **acsyn** iterations is performed within a base period, for **oad** still to be able to reasonably compare partly-evaluated planforms, without getting into numerical noise. Had the licensing limitation not been present, to be on the safe side for **oad**, at least a double of **struct**, **aero**, and **acsyn** iterations would have been performed, and probably running parallel to each other (see Sec. V.B for rationale). Although unfortunate, this illustrates nicely strange driving factors which may be expected in practical use.

Reconfiguration of the base period within MDO Driver is a matter of changing a few lines or values in different discipline's `toolspec` files within the process. For example, the `toolspec` file for **struct** is shown in Listing 3. It can be seen that **struct** is set to run after **oad** (`run-serial-after`), that it performs 2 iterations between data exchanges (`data-update-interval`), and what the execution order of its input collectors should be (`input-collector-order`). Here another special input collector appears, `from-self`, which is used to fetch serialized state from the previous iteration (sleeping-only disciplinary processes, such as **oad** in this case, do not need this collector). The `from-input` collector is by default run only once at start of the overall process, but for serializing tools it may need to be run at each data exchange (as here), to fetch the constant elements that are not included in the serialized state fetched by `from-self`.

Listing 3 `toolspec` file for struct that establishes its track in Fig. 12

```
integrator: Matthias Schulze <matthias.schulze@dlr.de>

tool: mona:1.6
type: serializing
data-update-interval: 2
```

```
run-serial-after:
  - oad

input-collector-order:
  - from-input
  - from-self
  - from-oad
  - from-acsyn
  - from-aero

input-collector-options:
    from-input:
        run-always: true
```

Convergence of the overall process is shown by Fig. 13. For each discipline, a development of one of its characteristic QIs is shown, including the mission block fuel-to-payload ratio r_{FP} as the global objective. Number on the x-axis means that the corresponding y-axis values are those at the end of so many base periods. Zero on the x-axis has no meaning in this context, and has been conveniently abused to show the baseline value of the respective QI (recall that the baseline design arises from a separate optimization process with outer shape DPs fixed to their initial values). The norm of the step in **oad** is the square norm of normalized DP perturbations around the current best design, used to estimate Jacobian/Hessian information. The final step norm of about 0.08 corresponds to wing aspect ratio step of 0.08 and wing sweep step of 0.16 [deg]; at this point the optimization was stopped, because it entered the region of numerical noise. Peak resource usage of 1280 cores occured whenever **aero** iterations were running in parallel with each other across all different planforms. During the run there were several interruptions, caused by cluster maintenance, unrobustness of convergence settings in some disciplines, and fixes to data recovery behavior on process restarts. When the interruptions are eliminated, process run time (for depicted 5 base periods) was 12 days; with interruptions included, 16 days.

The baseline and optimized planforms are compared in Fig. 14. The optimized design has significantly higher aspect ratio and somewhat higher sweep, which could have been expected for a pure performance objective such as mission block fuel and lack of constraints such as landing gear integration, flutter, or take-off and landing performance[7, 45]. Another constraint not taken into account is the lateral controllability in the engine-out condition, which would have necessitated a larger vertical tail in the optimized design due to the more outward engine position, or, alternatively, another DP to control the spanwise planform break location, to which the engine is relatively positioned. Furthermore, airfoil sections have been modified to be slightly thinner at some span segments, especially inboard, and to have more rear loading, which leads to reduced shock strength. (A visual comparison of airfoil shapes and pressure distributions at the design cruise point cannot be shown due to the XRF-1 non-disclosure agreement; for the same reason some values in figures and tables, such as drag coefficients, are shown only as absolute difference to baseline.)

On the other hand side, the increase in aspect ratio and sweep in the optimized design was not as large as might have been expected from the previous studies. A closer look at the results of **oad** revealed a steep jump in optimized wing mass values coming from the perturbed designs for Jacobian estimation within SQPDFO at aspect ratios nearing 12, as well as at smaller aspect ratios as sweep increased towards 34 [deg] and beyond. Therefore a separate wing mass optimization study was performed with **struct** alone, using stepwise planform DPs with fixed airfoil DPs, as depicted by Fig. 15. Mass jumps that can be seen happen due to a critical ground load case coming from the embedded conceptual ground- and landing-load estimation tool within cpacs-MONA. The main landing gear is attached to the wing at a fixed relative span-wise position, so that absolute landing gear position changes accordingly with planform changes (similar to how the nacelle moves in Fig. 14). For ground stability a certain wheelbase (y-position) and track width (x-position) of the landing gear according to the center of gravity (CG) have to be ensured. A higher aspect ratio at constant wing area leads to a smaller wing chord and a more rearward located CG-position, which decreases the wheelbase. This further activates the critical case ground load case and produces an abrupt jump in the landing gear loads. In essence, although there were no explicit constraints regarding landing gear integration, an implicit penalty on the overall objective was introduced into the overall process through the more detailed loads process.

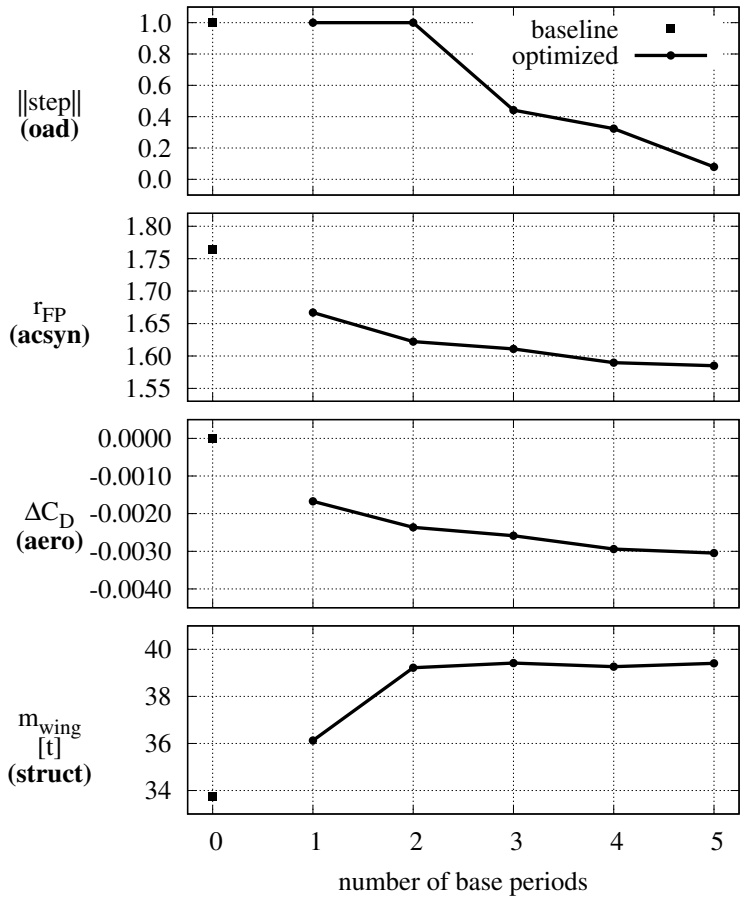


Fig. 13 Optimization convergence history of overall aircraft design by selected disciplinary QIs: optimizer step norm $||step||$, block fuel-to-payload mass ratio r_{FP}, drag coefficient at cruise C_D, and wing structural mass w_{wing}. 1280 computational cores were used at peak, total "clean" run time was 12 days.

In Fig. 16 the normalized maximum and minimum cut-moments for bending (M_x) and torsion (M_y) of the main wing over its half-span are shown for the baseline and the optimized design. Loads are normalized with respect to the maximum or minimum values of the baseline variant. The maximum negative bending moment has been increased, while the maximum positive bending moment has nearly stayed the same for the optimized design. Torsional cut-moments have been changed more prominently, being decreased over most of the span.

Fig. 17 shows qualitatively the difference in structural thickness distribution of the load-carrying wing structure between baseline and optimized designs. Red fields indicate an increase of thickness, green fields indicate a decrease, while black fields are about equal between the two designs. Even though the dimensioning cut-moments have been slightly decreased (in the case of torsional moment), the optimized wing has a distinctly heavier structure, as shown by the increase in thickness nearly over the whole wing. Only the front and rear spars, and the outer section of the wing, stayed nearly unchanged. One reason for the increase of skin and rib thickness are thinner airfoils and smaller wing chords.

A selection of QIs, including planform DPs, has been compared between designs in Table 1. Beside the baseline and optimized design, also another design, optimized under same conditions but with planform DPs fixed to baseline (i.e. where **oad** does nothing), has been included. This design serves to illustrate how much has been gained between optimizing only the airfoil shape versus planform and airfoil shape together.

The table shows the expected trade-off between performance quantities (lift-to-drag ratio or fuel-to-payload ratio) and cost quantities (operating empty or wing structure mass), when the design is optimized

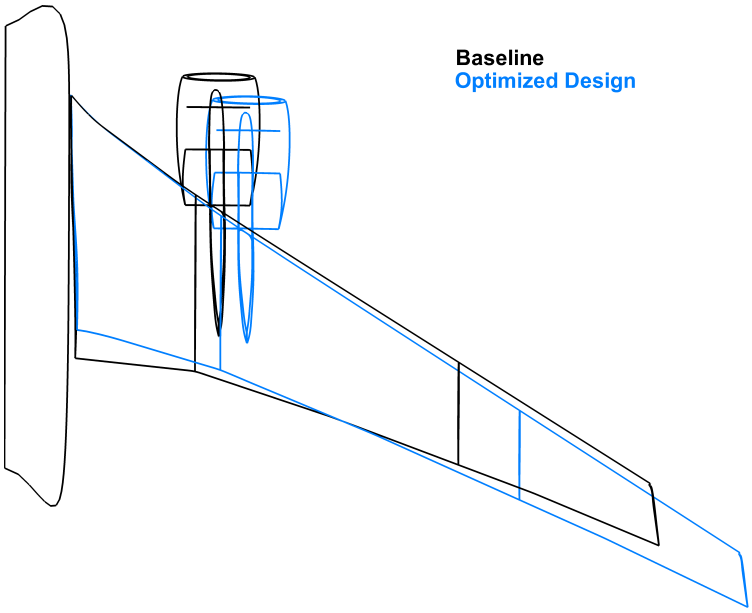


Fig. 14 Comparison of baseline and optimized planform shape, including repositioned engine as implied by the parametric CAD model. The wing reference area is kept constant.

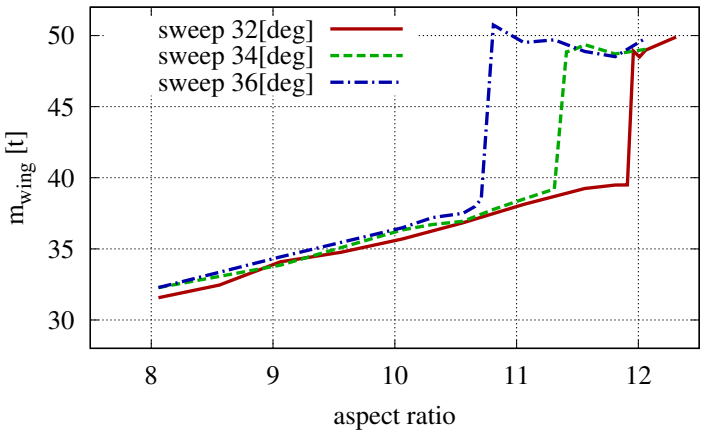
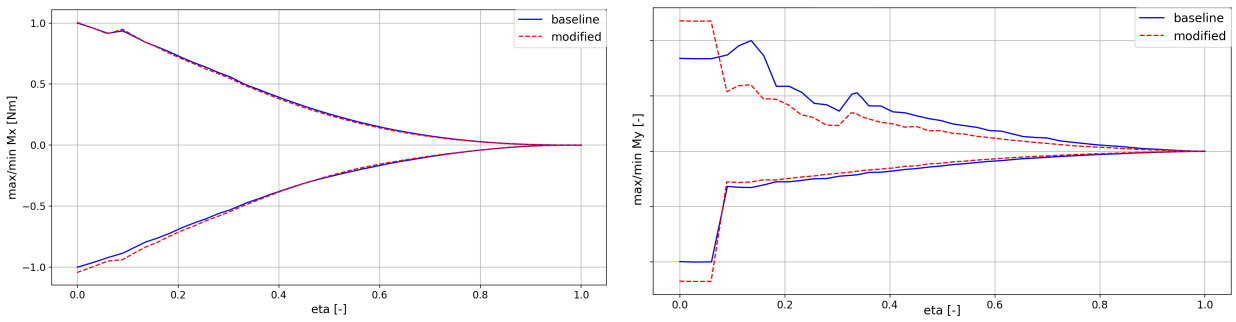


Fig. 15 Optimized wing mass with variations in aspect ratio and sweep, showing the effect of activation of landing load cases within structural optimization due to changing position of main landing gear.

for performance. Mission block fuel has been reduced by -10.2% (-5.6% for fixed-planform optimization), for increase of 8.6% (2.3% fixed-planform) in operating empty mass.

Somewhat unexpected is that the maximum take-off mass *decreased*. The cause of this is twofold. Firstly, there is the way **acsyn** computes maximum take-off mass, as mentioned earlier: it fixes by construction the vertical position of the maximum-fuel corner point in the payload-range diagram, but not its horizontal position. Secondly, the geometry definition is such that a more slender wing will have less space for fuel, and of course the same applies for a thinner wing. The optimizer was therefore allowed to trade-off range at maximum-fuel mission (sometimes called "design mission") with range at the highest-frequency mission ("study mission", mission under optimization), if advantageous. The table indeed shows that maximum fuel mass (i.e. wing tank volume) has been significantly reduced in the optimized design, by -14.1%. However, aerodynamic performance has been increased (lift-to-drag ratio +12.5%), so it remains to be post-processed



(a) bending cut-moments **(b) torsional cut-moments**

Fig. 16 Maximum and minimum cut-moments over the wing span normalized to baseline.

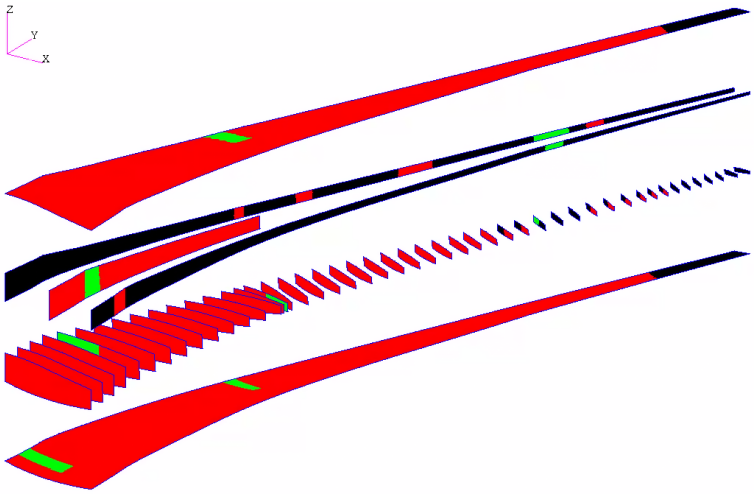


Fig. 17 Qualitative comparison of baseline and optimized structural thicknesses. Red fields stand for significant increase, green fields for significant decrease, and black fields for small changes.

what has actually happened with the range at maximum-fuel mission.

B. Local Subsystem Design

Preceding the overall aircraft design process of Sec. V.B, another, in some aspects simpler but in other aspects more complex MDO process was assembled and tested. It dealt with optimizing only local subsystem DPs, i.e. keeping the wing planform DPs fixed. For this reason also missing is **acsyn**, because careful mass accounting was not deemed crucial in case of fixed planform (e.g. maximum take-off mass for the purpose of defining load cases in **struct** is kept fixed). Results from this process are included in the presentation as well, for two reasons: to illustrate the flexibility that allows of including yet more complex loads process, split across different disciplines, and to underline decoupling of problem representation and process execution, that stems from the design-equation reasoning. Values are not directly comparable with those from Sec. V.B (e.g. baseline values differ too).

The cybermatrix of the local sybsystem design problem is given by Fig. 18 and the elements described in Listing 4. The base period is depicted in Fig. 20, and single iteration computation in Listing 5. There are no new representational elements in its cybermatrix or base period figures.

quantity of interest	unit	**baseline**	**optimized w. fixed planform**	**optimized**
wing aspect ratio		9.06	9.06	11.60
wing sweep	[deg]	32.0	32.0	32.7
block fuel-to-payload mass ratio		**1.763**	**1.664 (−5.6%)**	**1.583 (−10.2%)**
mid-cruise lift coefficient		0.5386	0.5374	0.5429
mid-cruise lift-to-drag ratio		(0.0%)	(+5.8%)	(+12.5%)
mid-cruise angle of attack	[deg]	2.915	2.818	3.294
mid-cruise horizontal tail trim angle	[deg]	−2.364	−2.485	−2.882
wing structural mass	[t]	33.8	35.3 (+4.4%)	39.1 (+15.7%)
operating empty mass	[t]	124.0	126.9 (+2.3%)	134.7 (+8.6%)
maximum take-off mass	[t]	246.8	247.1 (+0.12%)	242.1 (−1.90%)
maximum fuel mass	[t]	108.4	105.8 (−2.4%)	93.1 (−14.1%)

Table 1 Comparison of selected QIs between baseline, optimized with fixed planform and optimized design. Relative difference to baseline shown in parenthesis.

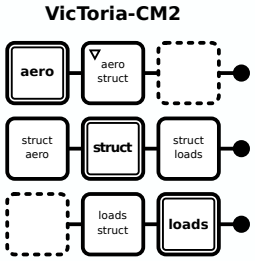


Fig. 18 Cybermatrix representation of the local subsystem design.

Listing 4 Disciplines and couplings of the cybermatrix from Fig. 18

aero, *aerodynamic airfoil design*

Same as in Sec. V.A. For the runs showing discipline order independency (Fig. 23), a very coarse CFD mesh was used instead, containing only 0.6 million points.

aero←`input`

Same as in Sec. V.A.

aero←**struct**

Same as in Sec. V.A.

struct, *design loads and structural sizing*

Same as in Sec. V.A.

struct←`input`

Same as in Sec. V.A.

struct←**aero**

Same as in Sec. V.A.

struct←**loads**

Gust cut-loads.

loads, *gust loads evaluation and selection*

Gust loads are treated by the VarLoads framework [46]. 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 cpacs-MONA). Aerodynamic forces are computed with a doublet-lattice method. Fig. 19 depicts typical panel aerodynamics and DFEM models used.

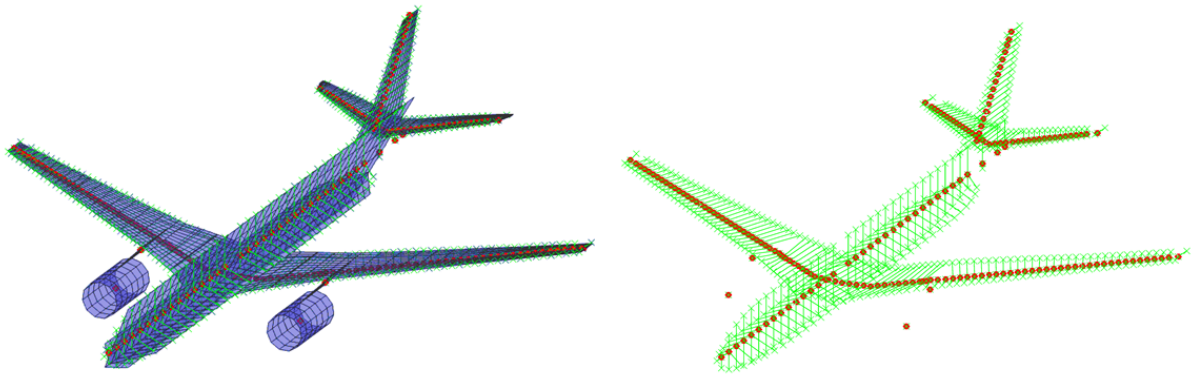


Fig. 19 Panel aerodynamics model (left) and DFEM (right) of the full aircraft. DLM panels and beam elements.

In this work **loads** adds the more complex, gust load cases to the maneuver cases already computed by **struct**. 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.

load←`input`

Definition of load cases for gusts. Initial DFEM, initial mass cases, initial panel aerodynamic model.

loads←**struct**

Mass cases and DFEM.

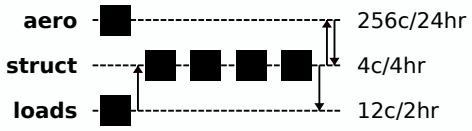


Fig. 20 Selected base period for the local subsystem design.

Listing 5 Single iteration of a discipline in the base period from Fig. 20

---

aero, *aerodynamic airfoil design*

Same as in Sec. V.A.

struct, *design loads and structural sizing*

Same as in Sec. V.A, however here FSD was used instead of gradient-based optimization. This was done because FSD was less prone to numerical noise than gradient-based optimizer, which was advantageous for tight convergence when demonstrating block-solving and discipline order independencies (Fig. 22 and Fig. 23).

loads, *gust loads evaluation and selection*

One complete loads evaluation and selection of critical loads, based on the 1D selection of maximum bending, torsion, and shear cut-loads, per component cut.

Why the base period on Fig. 20 was set up in that particular way—**loads** running in serial with **struct**, **aero** in parallel with them—will be explained shortly. In the resource labels of the base period on Fig. 20 there are no separate job (`sj`) indicators, which means that the complete optimization run as a single cluster job. Compared to base period of overall aircraft design (Fig. 12), there are more **struct** iterations. One reason for this is that FSD converges more slowly than gradient-based optimization. The other reason is that there is a lot of time available while **aero** runs in parallel with **loads** and **struct**, which might as well be utilized to have a more converged structure at the next point of data exchange.

The only reason why **loads** was not included in the overall aircraft design optimization of Sec. V.B was organizational: at the time of writing, it was still in progress of being ported to the new HPC cluster that came on-line in between. It were again the licensing issues, in this case of MATLAB, that made the effort somewhat more complicated.

The general result of local subsystem design is briefly shown, through the convergence plot in Fig. 21. The interesting element not present in Sec. V.B is that the number of selected gust load cases, produced by **loads** for use by **struct**, was changing between base periods. This means, effectively, the number of structural constraints in **struct** was different in each base period. This would have been a problem for some of the classic MDO approaches, where a single optimizer is handling all DPs and DIs across all disciplines, but does not present a problem with distributed optimizers of the cybermatrix approach. The **struct** discipline simply carries over the design from the previous base period to use as the initial design in the current period, and applies the new set of loads. (A probable price to pay is reduced convergence rate, since any accumulated Hessian information the structural optimizer may have had is discarded between base periods.)

From the viewpoint of reasoning through design equation, it is obvious that the set up of the base period can only influence the convergence path of the optimization, and not the optimized design. (Unless there are multiple optima within the applied range of DPs, in which case different convergence paths may lead to different optima.) This is however not as obvious from the viewpoint of classic aircraft design. The following two studies demonstrate the independency of optimized solution, and thus also decoupling of the cybermatrix problem representation and implementation, from the choice of the base period.

Fig. 22 shows what happened with the solution when the base period was arranged like a block-Jacobi iteration, a block-Gauss-Seidel, or a mix between the two. Note that a very coarse CFD mesh of 0.6 million points was used here, therefore C_D values are different from Fig. 21. The same optimized design was obtained, and there was practically no difference between convergence rates. This is an interesting result, as one would expect Gauss-Seidel to be faster than Jacobi; lack of this behavior reveals something about the nature of the coupling between disciplines. While base period convergence rates were the same, run times were not. Since Gauss-Seidel executes disciplines in serial and Jacobi in parallel, due to average run times of disciplines, Gauss-Seidel needed twice as much time per base period as Jacobi. Since it was **aero** that was the run-time bottleneck here, the mixed base period ended up needing same run time as Jacobi. But the mixed period had an advantage: smaller peak computational core usage than Jacobi, since **loads** and **struct** never run at the same time. Therefore, the mixed period was both the most time- and most resource-conserving of the three, and because of that it was chosen for application with a finer CFD mesh (Fig. 20) and used to produce the general result of this section (Fig. 21).

In the block-Gauss-Seidel mode, for example, it is also possible to have different order of execution of disciplines. Fig. 23 demonstrates that also in this case the same final design was reached, the difference only being in the initial convergence rate. This effect is expected from the system-of-equations point of view, since different row orderings do not change the stiffness properties of the system matrix (left-hand side), and thus also not the asymptotic convergence rate. The interesting part here is that the effect was demonstrated on full-blown disciplinary processes, numerically and implementationaly dissimilar, based solely on design-equation reasoning.

VI. Relation to Other Concepts

Due to the expansiveness of the GOAL, it is worthwile to discuss how some other concepts commonly found in other MDO approaches relate to the approach presented in this work. One important aspect of the GOAL to keep in mind for the following is the requirement of large-scale parallelism and recursiveness, in both

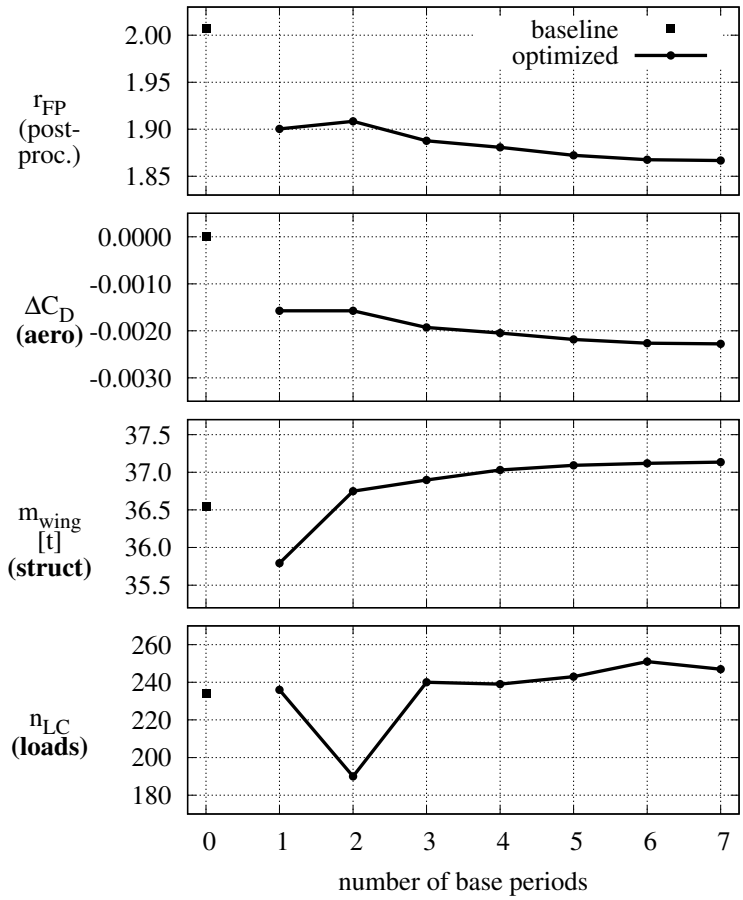


Fig. 21 Optimization convergence history of local subsystem design by selected QIs: block fuel-to-payload mass ratio r_{FP} (post-processed, no aircraft synthesis discipline during optimization), drag coefficient at cruise C_D, wing structural mass w_{wing}, and number of selected critical gust load cases n_{LC}.

collaboration and computation.

MDO formulations (architectures) are a set of well-examined patterns for creating MDO processes[47], one of which can be chosen for a particular design problem based on its disciplinary and coupling properties[33]. A cybermatrix MDO process does not generally fit any of the known formulations. It may be akin to the interdisciplinary-feasible (IDF) formulation, except that there is neither a common optimizer nor is clear-cut what is feasible at which point of the process. Exchange of design couplings may be similar to the original BLISS formulation[35], except that there are no levels, but the coupled-adjoint analogy is followed. In fact, a cybermatrix process such in Sec. V.A is conceptually nearest to the one-shot approaches[48, 49], only applied at a coarser scale. The base period of any practical cybermatrix process will depend on a host of factors, including non-technical ones, such as restrictions due to software licensing that were prominent in Sec. V.

MDO process visualizations in wider use, such as algorithmic flow charts, or even more representations such as XDSM[50] diagrams, cannot represent everything that may happen in a cybermatrix process. What they can, may end up needlessly complex from the design-equation viewpoint. They also often imply the loops-within-loops implementation, whereas in a cybermatrix process there is always only a single loop, the repetition of base period. Thus, the base period visualization introduced in Sec. IV.B is sufficient; and that only where it is needed, since to reason about the properties of the optimized design it is enough to examine the cybermatrix of the design problem.

Centralized geometry definition for all involved disciplines and fidelity levels[51, 52] is often touted as the means of assuring interdisciplinary consistency in an MDO context. In terms of the GOAL, however, it

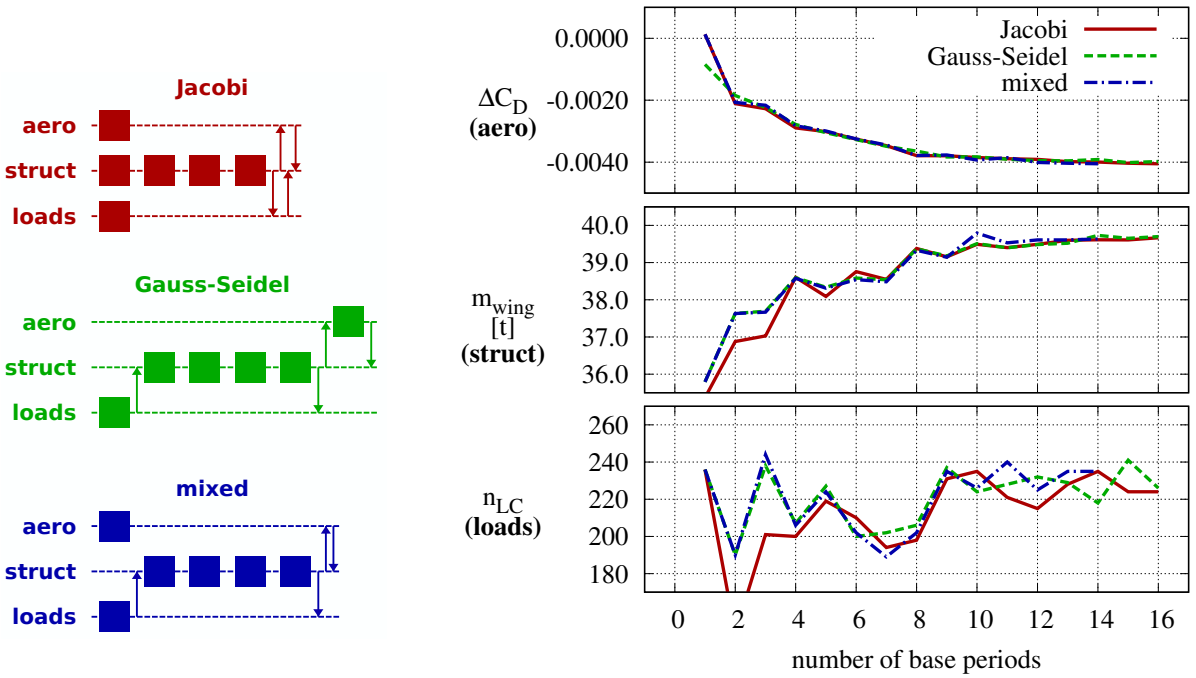


Fig. 22 Base periods and convergence history for different solving modes. (C_D span significantly higher than in other figures because of a very coarse CFD mesh.)

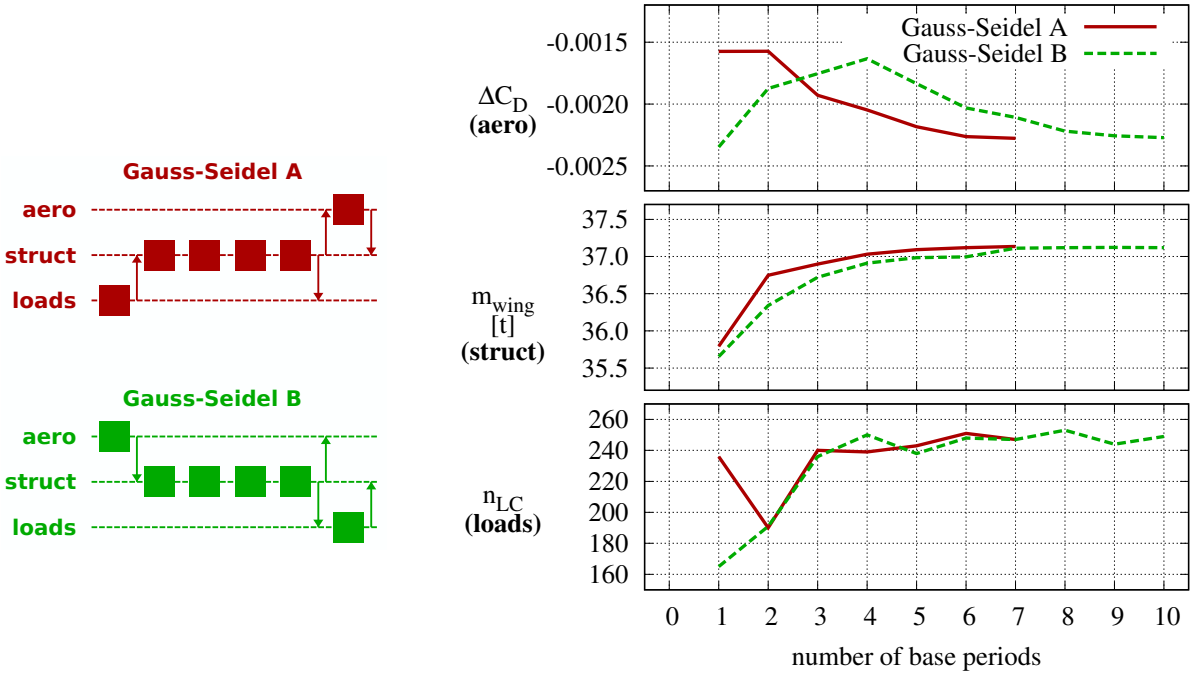


Fig. 23 Base periods and convergence history for different orders of disciplines in the block-Gauss-Seidel solving mode.

is disadvantageous. On the one hand side, centralization introduces a bottleneck in a parallel activity in general. For geometry in particular, it forces all disciplines into a design-by-committee common geometry definition, rather than allowing them to focus on software and concepts most pertinent to the discipline. On the other hand side, common *source* of geometry does not at all assure consistent *interpretation* of the source. The proper check for consistency is an explicit automatic evaluation of differences between final disciplinary models (such as between a wing surface CFD mesh and a wing-box FE model)—analogous to unit tests in terms of software engineering. Such automatic evaluation can be used in a cybermatrix process either to force consistency by construction, or by becoming one of interfacing constraints.

Ageometric design parameters, such as thrust-to-weight ratio or wing loading, may indeed be *parameters*. That is, they are not derived from some other quantities, but other quantities are derived from them. For example, if wing loading is a DP and maximum take-off mass is computed, the reference wing area will be derived from the two, and not computed from some geometric model (particularly important to recognize when discussing geometry definition). A question might be raised, why ageometric DPs are needed at all, especially given that the GOAL leads to the complete preliminary design, including all relevant geometric measures. The reason is that ageometric DPs in the design-equation reasoning have the role of preconditioners. They enable faster convergence and avoidance of local optima, as well as not getting bogged-down in numerical noise (in case if shape were defined through a huge number of fine local parameters). Therefore supporting them is crucial in any MDO approach. (The same holds for any global-scale parameters, such as wing aspect ratio or sweep.)

Design feasibility during optimization is desirable to have at certain points, rather than only at the end of optimization, in case (as would be expected in practice) there is a hard time deadline for ending a design activity. What a feasible design is, is a matter of some interpretation. For example, even in a multidisciplinary feasible (MDF) MDO formulation, control constraints for steady flight (lift balance, pitching moment balance, etc.) may be left to the top optimizer[47], or made feasible in each step[42]. However, whatever feasibility is defined to mean, cybermatrix processes are intrinsically infeasible-until-convergence. To remedy this, when desired, at the end of each base period another process may be "spun off" in parallel, in which only those DPs needed to satisfy the constraints would be left free (for example, wing planform DPs would be fixed, wing structural thicknesses left free). The objective function of the spin-off process would not be the original objective function, but the norm of the difference between leftover free DPs and their values at the spin-off point. This would produce a time sequence of feasible designs without slowing down the main optimization.

Common aircraft descriptions (ontologies), such as CPACS[53] or AiX[54], are strictly-defined data formats for all things related to aircraft design, spanning multiple disciplines, including geometry, mission, ground operation, maintenance, etc. A centralized geometry definition is often a part of such ontologies. Some MDO frameworks directly rely on ontologies [32] to produce MDO processes. In the cybermatrix approach, as in any data-exchange scenario, it is always advantageous to have and use common formats. However, due to the GOAL there will always be new elements, new involved disciplines and new technologies, whose requirements are not yet covered by the ontology at hand. Since precisely these elements are what drives novel designs, the cybermatrix protocol is especially focused on minimizing the effort in collaboration where no common formats exists. This goes hand-in-hand with employment of clever design methods from Sec. III.D.

Involvement of conceptual design in an MDO process that is supposed to produce a preliminary design is still an open matter. Extant preliminary-design MDO processes subsume conceptual design within them[7, 9, 44], by directly handling DPs that would traditionally be under conceptual design control (such as wing planform shape). At most, conceptual design tools are used to compute QIs at low fidelity[44], for which high-fidelity methods are not yet available or are too expensive. Conceptual design phase in any case retains the role of exploring TLARs and vehicle concepts, selecting most promising candidates for preliminary-design MDO process. But how could conceptual design also directly participate in an automatic preliminary-design MDO process? The conjecture is that in a cybermatrix process it could serve as a way to speed up convergence and avoid local optima, in a manner analogous to coarser grid in a multi-grid method; or, somewhat equivalently, as the low-fidelity counterpart in a multi-fidelity method. The key difference to traditional conceptual design in this scenario is that it would not be left to control some DPs by itself; it would have to take DP corrections (explicitly or implicitly through other quantities), from the higher-fidelity disciplines during the optimization.

Distinction between tools and processes is often made in the context of workflow managers and process integration frameworks. In effect, disciplinary tools and MDO processes built from them tend to have very

different implementation and execution properties. In the cybermatrix protocol, this is not the case: an implemented MDO process is a tool with exactly the same interface and runtime properties as the tools it is composed of, and is ready to be used itself as a tool in a superprocess, as explained in Sec. IV.C. This recursiveness is crucial for the large-scale integration required by the GOAL.

Workflow manager frameworks are pieces of software "in" or "with" which an MDO process is assembled and run. They come with many highlights, including providing advanced graphical-user interfaces and over-the-network tool execution and process steering[55], or direct support for HPC environments[56]. However, an MDO process made in one framework can almost never be transferred into another framework without a painstaking conversion efort, they often establish a strong process–tool distinction, and require disciplinary experts to provide specialized wrappers for their tools. This is all counter-productive to the GOAL, especially to the recursiveness property. With the cybermatrix protocol, an MDO process as shown in Sec. IV.C can be implemented without the knowledge of a particular framework, and then interpreted by any framework supporting the protocol, as suited to the computational environment. This is analogous to the way program code a in well-specified programming language can be executed by different interpreters (or compiled by different compilers).

In-memory process integration frameworks provide a highly-parallel in-memory data storage core and a programming interface, which multiple disciplines can use to exchange data in an MDO process without suffering slow disk file operations[57, 58]. These frameworks in effect serve to maximize the performance of an MDO process, as measured, for example, in wall-time FLOPS on a given design problem size. Unfortunately, this comes at a cost. Often there is no storage separation between disciplines (each discipline can read and write what it wants), leading to hard-to-find bugs. When bugs do occur and the running process crashes, runtime data that could have been used to trace the issue has disappeared. For time- and resource-intensive computations, with many disciplinary experts—who are typically not software engineers—involved in the implementation, debugging becomes a problem. Since the whole run is one (parallel) process, it implies a single cluster job; with the expected dissimilarity of disciplinary tools, significant part of the assigned resources may wait doing nothing at times. This reduces not only the total throughput of the cluster, but likely also any given user's throughput, due to longer waiting times for large monolithic jobs to start. Reduced throughput may in practice eat through the higher performance that the framework affords, which becomes the more likely the larger the integration scale of the process. Thus, while the cybermatrix protocol allows for use of any cybermatrix-capable in-memory framework, already for the very small demonstration case (relative to the GOAL) of Sec. V.A use of such framework alone (as opposed to as a subprocess framework) would have been prohibitive.

Model based systems engineering (MBSE) is a practice of reasoning about and steering complex engineering systems by establishing and maintaining strict formal models of the subject domain[59]. In this sense it seems perfectly suited to the scale requirements of the GOAL. However, contemporary MBSE publications rarely, if ever, deal directly either the design equation (i.e. the KKT optimality conditions) or with parallelism and large computational resources. An MBSE approach that would consider these two elements together however should be advantageous to the cybermatrix context as well.

Dedicated MDO teams are often introduced in connection with efforts to bring MDO into practical use[1, 6]. The cybermatrix protocol does not require such teams in the technical sense, as it would present a bottleneck to the parallelism (similar to a central geometry description). Instead, the multidisciplinary knowledge is distributed among disciplinary experts, who need to understand how DPs under their control affect other disciplines, and collaborate with their counterparts from other disciplines on defining couplings. In particular, as explained in Sec. IV.C, for discovering missing design couplings it is crucial for all disciplinary teams to be able to run the whole MDO process between themselves symmetrically. (A dedicated MDO team can instead be advantageous in the administrative sense, of bringing the disciplines together, providing the work pacing, or managing computational platforms.)

VII. Conclusion and Outlook

The cybermatrix protocol, a novel MDO approach aiming at enabling construction of automatic design processes that start from TLARs and a vehicle concept and produce a complete preliminary design, was presented and demonstrated on the case of overall aircraft optimization of a long-range twin-engine airliner. The optimized designs were according to expectations from previous studies, but also exhibited some new

effects due to employment of a more complex loads process.

For further demonstrations, three directions will be followed.

Firstly, increase of complexity within the presented disciplines, including a powered engine in the aerodynamic airfoil design, higher-fidelity corrections to loads processes, and more detailed consideration of landing gear integration. For the latter, for example, new DPs can be introduced for independent positioning of the main landing gear, while in the structural design a side brace can be added to it, to partially relieve the wing by directing a part of the landing load into the fuselage.

Secondly, a number of other disciplines are in various stages of adaptation to HPC resources and coupling into cybermatrix processes. These are a more detailed aircraft synthesis using OpenAD[41], separate disciplines for more detailed static GFEM modeling and structural sizing of fuselage and lifting surface through Pandora[60] and DELiS[23], and engine conceptual design using GTlab[5].

Thirdly, more design couplings should be added to the disciplines. This is challenging, because it requires that disciplinary experts "buy-in" into the design-equation reasoning and to consider how to extend existing processes with that in mind. For example, the aerodynamic airfoil design could use an approximate sensitivity of wing mass to airfoil thickness, as well as that of structural constraints, to be provided by the structural design.

Beyond disciplinary additions, further work on the cybermatrix protocol definition and process integration frameworks will be carried out.

Acknowledgments

The authors would like to thank Airbus for providing the XRF-1 test case for demonstration of the approach presented in this paper. Beyond the authors, this work was made possible by a collaborative effort of many colleagues from the nine listed DLR institutes, who all contributed over time with various levels of involvement.

References

[1] Schuhmacher, G., Daoud, F., Petersson, O., and Wagner, M., "Multidisciplinary airframe design optimisation," *28th Congress of the International Council of the Aeronautical Sciences*, 2012.

[2] Piperni, P., DeBlois, A., and Henderson, R., "Development of a multilevel multidisciplinary-optimization capability for an industrial environment," *AIAA Journal*, Vol. 51, No. 10, 2013.

[3] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design," *Optimization and Engineering*, Vol. 6, No. 1, 2005. doi:10.1023/B:OPTE.0000048536.47956.62.

[4] Scherer, J., Kohlgrüber, D., Dorbath, F., and Sorour, M., "A finite element based tool chain for structural sizing of transport aircraft in preliminary aircraft design," *DLRK 2013*, 2013.

[5] Becker, R.-G., Wolters, F., Nauroz, M., and Otten, T., "Development of a gas turbine performance code and its application to preliminary engine design," *Deutscher Luft- und Raumfahrtkongress 2011*, 2011.

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

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

[8] Gazaix, A., Gallard, F., Gachelin, V., Druot, T., Grihon, S., Ambert, V., Guénot, D., Lafage, R., Vanaret, C., Pauwels, B., Bartoli, N., Lefèbvre, T., Sarouille, P., Desfachelles, N., Brézillon, J., Hamadi, M., and Gürol, S., "Towards the industrialization of new MDO methodologies and tools for aircraft design," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.

[9] Wunderlich, T., and Dähne, S., "Aeroelastic tailoring of an NLF forward swept wing," *CEAS Aeronautical Journal*, 2017. doi:10.1007/s13272-017-0251-6.

[10] Kroll, N., Abu-Zurayk, M., Dimitrov, D., Franz, T., Führer, T., Gerhold, T., Görtz, S., Heinrich, R., Ilic, C., Jepsen, J., Jägersküpper, J., Kruse, M., Krumbein, A., Langer, S., Liu, D., Liepelt, R., Reimer, L., Ritter, M., Schwöppe, A., Scherer, J., Spiering, F., Thormann, R., Togiti, V., Vollmer, D., and Wendisch, J.-H., "DLR Project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods," Vol. 7, 2018. doi:10.1007/s13272-015-0179-7.

[11] Görtz, S., Abu-Zurayk, M., Ilic, C., Wunderlich, T., Keye, S., Schulze, M., Kaiser, C., Özge Süelözgen, Schuster, A., Dähne, S., Petsch, M., Häßy, J., Mischke, R., Knechtges, P., and Gottfried, S., "Overview of collaborative multi-fidelity multidisciplinary design optimization activities in the DLR project VicToria," *AIAA Aviation 2020*, 2020.

[12] Nocedal, J., and Wright, S. J., *Numerical optimization*, 2nd ed., Springer, 2006.

[13] Kroo, I., "Distributed multidisciplinary design and collaborative optimization," *VKI lecture series on Optimization Methods and Tools for Multicriteria/Multidisciplinary Design*, 2004.

[14] Görtz, S., Ilic, C., Jepsen, J., Leitner, M., Schulze, M., Schuster, A., Scherer, J., Becker, R.-G., Zur, S.-A., and Petsch, M., "Multi-level MDO of a long-range transport aircraft using a distributed analysis framework," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.

[15] Brown, R. D., Kroo, I. M., and Gage, P. J., "Post-optimality analysis in aerospace vehicle design," *AIAA Aircraft Design, Systems and Operations Meeting*, 1993.

[16] Samuelson, P. A., "Mathematics of speculative price," *SIAM Review*, Vol. 15, No. 1, 1973, pp. 1–42.

[17] Audet, C., Dennis Jr., J., and Digabela, S. L., "Trade-off studies in blackbox optimization," *Optimization Methods and Software*, Vol. 27, No. 4-5, 2012, pp. 613–624. doi:10.1080/10556788.2011.571687.

[18] Abu-Zurayk, M., Merle, A., Ilic, C., Goertz, S., Schulze, M., and Klimmek, T., "A comparison of different formulations for aero-structural optimization of trimmed transport aircraft," *EUROGEN*, 2019.

[19] Lambe, A. B., Kennedy, G. J., and Martins, J. R., "An evaluation of constraint aggregation strategies for wing box mass minimization," *Structural and Multidisciplinary Optimization*, Vol. 55, 2016. doi:10.1007/s00158-016-1495-1.

[20] Klimmek, T., "Parametric set-up of a structural model for FERMAT configuration for aeroelastic and loads analysis," *ASD Journal*, Vol. 3, No. 2, 2014. doi:10.3293/asdj.2014.27.

[21] Takanashi, S., "Iterative three-dimensional transonic wing design using integral equations," *Journal of Aircraft*, Vol. 22, No. 8, 1985, pp. 655–660. doi:10.2514/3.45182.

[22] Patnaik, S. N., and Hopkins, D. A., "Optimality of a fully stressed design," *Computer Methods in Applied Mechanics and Engineering*, Vol. 165, No. 1, 1998, pp. 215–221. doi:10.1016/S0045-7825(98)00041-3.

[23] Führer, T., Willberg, C. B., Freund, S., and Heinecke, F., "Automated model generation and sizing of aircraft structures," *Aircraft Engineering and Aerospace Technology*, Vol. 88, 2016, pp. 268–276. doi:10.1108/AEAT-02-2015-0054.

[24] Pusch, M., Knoblach, A., and Kier, T., "Integrated optimization of control surface layout for gust load alleviation," *CEAS Aeronautical Journal*, Vol. 10, No. 4, 2019, pp. 1059–1069. doi:10.1007/s13272-019-00367-4.

[25] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The complex-step derivative approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003.

[26] Widhalm, M., Brezillon, J., Ilić, Č., and Leicht, T., "Investigation on adjoint based gradient computations for realistic 3D aero-optimization," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.

[27] Hicken, J. E., and Zingg, D. W., "Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement," *AIAA Journal*, Vol. 48, No. 2, 2010, pp. 400–413.

[28] Ghazlane, I., Carrier, G., Dumont, A., Marcelet, M., and Désidéri, J.-A., "Aerostructural optimization with the adjoint method," *EUROGEN 2011*, 2011.

[29] Abu-Zurayk, M., and Brezillon, J., "Aero-elastic multipoint optimization using the coupled adjoint approach," *18th STAB/DGLR Symposium*, 2012.

[30] Wiener, N., *Cybernetics: or control and communication in the animal and the machine*, Technology Press, 1948.

[31] Gray, J. S., Hearn, T. A., Moore, K. T., Hwang, J., Martins, J., and Ning, A., "Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO," *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, 2014. doi:doi:10.2514/6.2014-2042.

[32] van Gent, I., Rocca, G. L., and Veldhuis, L. L. M., "Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems," *AIAA Aviation 2017*, 2017.

[33] Gallard, F., Vanaret, C., Guenot, D., Gachelin, V., Lafage, R., Pauwels, B., Barjhoux, P.-J., and Gazaix, A., "GEMS: a Python library for automation of multidisciplinary design optimization process generation," *AIAA SciTech 2018*, 2018.

[34] Nagel, B., Böhnke, D., Gollnick, V., Schmollgruber, P., Rizzi, A., Rocca, G. L., and Alonso, J. J., "Communication in aircraft design: can we establish a common language?" *28th ICAS Congress*, 2012.

[35] Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky Jr., R. R., "Bi-level integrated system synthesis," Tech. Rep. NASA/TM-1998-208715, National Aeronautics and Space Administration (NASA), 1998.

[36] Yoo, A. B., Jette, M. A., and Grondona, M., "SLURM: Simple Linux Utility for Resource Management," *JSSPP 9th International Workshop*, 2003, pp. 44–60. doi:10.1007/10968987_3.

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

[38] Merle, A., Ronzheimer, A., Bekemeyer, P., Görtz, S., Keye, S., and Reimer, L., "Gradient-based optimization of a flexible long-range transport aircraft using a high-dimensional CAD-ROM parameterization," *Deutscher Luft- und Raumfahrtkongress 2018*, 2018.

[39] Chittick, I. R., and Martins, J. R. R. A., "An asymmetric suboptimization approach to aerostructural optimization," *Optimization and Engineering*, Vol. 10, 2009, pp. 133–152.

[40] Tröltzsch, A., "A sequential quadratic programming algorithm for equality-constrained optimization without derivatives," *Optimization Letters*, Vol. 10, 2016. doi:10.1007/s11590-014-0830-y.

[41] Atanasov, G., van Vensween, J., Peter, F., and Zill, T., "Electric commuter transport concept enabled by combustion engine range extender," *DLRK 2019*, 2019.

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

[43] Klimmek, T., Schulze, M., Abu-Zurayk, M., Ilic, C., and Merle, A., "An independent and in high-fidelity based MDO tasks integrated process for the structural and aeroelastic design of aircraft configurations," *IFASD Conference 2019*, 2019.

[44] Abu-Zurayk, M., Merle, A., Ilic, C., Keye, S., Goertz, S., Schulze, M., Klimmek, T., Kaiser, C., Martin, D. Q., Häßy, J., Becker, R.-G., Fröhler, B., and Hartmann, J., "Sensitivity-based multifidelity multidisciplinary optimization of a powered aircraft subject to a comprehensive set of loads," *AIAA Aviation 2020*, 2020.

[45] Salah El Din, I., Dumont, A., and Blondeau, C., "Transonic wing-body civil transport aircraft aero-structural design optimization using a bi-level high fidelity approach – a focus on the aerodynamic process," *51st AIAA Aerospace Sciences Meeting*, 2013.

[46] Kier, T., and Looye, G., "Unifying manoeuvre and gust loads analysis models," *IFASD 2009*, 2009.

[47] Martins, J. R., and Lambe, A. B., "Multidisciplinary design optimization: a survey of architectures," *AIAA Journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

[48] Gauger, N. R., Griewank, A., and Riehme, J., "Extension of fixed point PDE solvers for optimal design by one-shot method," *European Journal of Computational Mechanics*, Vol. 17, No. 1–2, 2008, pp. 87–102. doi: 10.3166/remn.17.87-102.

[49] Schulz, V., and Gherman, I., "One-shot methods for aerodynamic shape optimization," *MEGADESIGN and MegaOpt - German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design*, Vol. 107, 2009, pp. 207–220. doi:10.1007/978-3-642-04093-1_15.

[50] 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*, Vol. 2, No. 46, 2012, pp. 273–284.

[51] Ronzheimer, A., Natterer, F. J., and Brezillon, J., "Aircraft wing optimization using high fidelity closely coupled CFD and CSM methods," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.

[52] Bryson, D. E., Haimes, R., and Dannenhoffer, J. F., "Toward the realization of a highly integrated, multidisciplinary, multifidelity design environment," *AIAA Scitech 2019 Forum*, 2019. doi:10.2514/6.2019-2225.

[53] Liersch, C., and Hepperle, M., "A distributed toolbox for multidisciplinary preliminary aircraft design," *CEAS Aeronautical Journal*, Vol. 2, No. 1, 2011.

[54] Risse, K., Anton, E., Lammering, T., Franz, K., and Hörnschemeyer, R., "An integrated environment for preliminary aircraft design and optimization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2012. doi:10.2514/6.2012-1675.

[55] Seider, D., Fischer, P., Litz, M., Schreiber, A., and Gerndt, A., "Open source software framework for applications in aeronautics and space," *IEEE Aerospace Conference*, 2012.

[56] Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., and Wenger, K., "Pegasus: a workflow management system for science automation," *Future Generation Computer Systems*, Vol. 46, 2015. doi:10.1016/j.future.2014.10.008.

[57] Meinel, M., and Einarsson, G. O., "The FlowSimulator framework for massively parallel CFD applications," *Para 2010*, 2010.

[58] Martins, J. R., and Kennedy, G. J., "A parallel aerostructural optimization framework for aircraft design studies," *Structural and Multidisciplinary Optimization*, Vol. 50, No. 6, 2014, pp. 1079–1101. doi:10.1007/s00158-014-1108-9.

[59] Dickerson, C. E., and Mavris, D., "A Brief history of models and model based systems engineering and the case for relational orientation," *IEEE Systems Journal*, Vol. 7, No. 4, 2013, pp. 581–592.

[60] Petsch, M., Kohlgrüber, D., and Heubischl, J., "PANDORA - A Python based framework for modelling and structural sizing of transport aircraft," *8th EASN-CEAS International Workshop*, 2018. doi:10.1051/matecconf/201823300013.