Combining Pareto Optimality with Nash Games in Multi-Objective Prioritized Optimization of an Aircraft Flight Performance

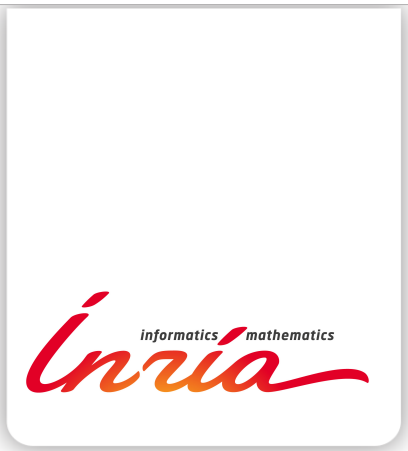
Jean-Antoine Désidéri, Julien Wintz, Nathalie Bartoli, Christophe David, Sébastien Defoort

Combining Pareto Optimality with Nash Games in Multi-Objective Prioritized Optimization of an Aircraft Flight Performance

Jean-Antoine Désidéri, Julien Wintz,
Nathalie Bartoli, Christophe David, Sébastien Defoort

RESEARCH REPORT
N° 9490
October 17, 2022
Project-Team Acumes

ISSN 0249-6399 ISRN INRIA/RR--9490--FR+ENG



Combining Pareto Optimality with Nash Games in Multi-Objective Prioritized Optimization of an Aircraft Flight Performance

Jean-Antoine Désidéri*, Julien Wintz†,
Nathalie Bartoli, Christophe David, Sébastien Defoort ‡

Project-Team Acumes

Research Report n° 9490 — October 17, 2022 — 29 pages

Abstract: This report reflects a cooperation between the Inria Project Team Acumes and the Information Processing and Systems Department (DTIS) of Onera Toulouse. We present a case study of prioritized multi-objective optimization of the flight performance of an Airbus-A320-type aircraft in terms of take-off fuel mass (f_1), operational empty weight (f_2) and ascent-to-cruise-altitude duration (f_3). The optimization is subject to functional constraints: interval bound on static margin and upper bound on wing span. Six design variables are considered defining geometry, mean aerodynamic chord and take-off thrust potential. Designs are evaluated by means of the FAST-OAD open-source software developed by ONERA and ISAE-SUPAERO, and the prioritized optimization is conducted by the Inria Nash-MGDA software. From a point of Pareto-optimality of the sole pair (f_1, f_2), considered preponderant, the climb duration f_3 is reduced drastically by a one-shot procedure establishing a continuum of Nash equilibria. The application of this procedure is far less computationally demanding than is the numerical establishment of the Pareto-front associated with the three cost functions together, while the Pareto-optimality of the preponderant cost functions (f_1, f_2) is only marginally degraded.

Key-words: differentiable multi-objective optimization, criterion prioritization, Nash game

* Inria Research Director, emeritus, Acumes Project Team
† Inria Research Engineer, Experimentation and Development Service (SED)
‡ Research Engineers, ONERA/DTIS, Université de Toulouse, Toulouse, France

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Pareto-optimalité combinée à des jeux de Nash en optimisation multiobjectif hiérarchisée de performance vol d'un avion

Résumé : Ce rapport est le fruit d'une coopération entre l'équipe-projet Inria Acumes et le Département Traitement et Systèmes de l'Information (DTIS) de l'Onera Toulouse. Nous présentons une étude de cas d'optimisation multi-objectif hiérarchisée de performance vol d'un avion de type Airbus-A320 en termes de masse de carburant au décollage (f_1), de masse opérationnelle à vide (f_2) et de durée de montée à l'altitude de croisière (f_3). L'optimisation est soumise à des contraintes fonctionnelles: borne d'intervalle sur la marge statique et borne supérieure sur l'envergure de l'aile. Six variables de conception sont considérées définissant la géométrie, la corde aérodynamique moyenne et le potentiel de poussée au décollage. Les concepts sont évalués au moyen du logiciel open-source FAST-OAD développé par l'ONERA et l'ISAE-SUPAERO, et l'optimisation hiérarchisée est conduite par le logiciel Inria Nash-MGDA. À partir d'un point de Pareto-optimalité du seul couple (f_1, f_2), considéré comme prépondérant, la durée de montée f_3 est réduite drastiquement par une procédure directe de calcul d'un continuum d'équilibres de Nash. L'application de cette procédure prend beaucoup moins de temps calcul que l'établissement numérique du front de Pareto associé aux trois fonctions de coût ensemble, tandis que l'optimalité de Pareto des fonctions de coût prépondérantes (f_1, f_2) n'est que marginalement dégradée.

Mots-clés : optimisation multiobjectif différentiable, hierachisation des critères, jeu de Nash

1 Introduction

In multi-objective optimization, the notion of Pareto-optimality is central [13]. In order to recall this concept, we consider the following multi-objective continuous optimization problem

$$\begin{cases} \min_{\mathbf{x}\in\Omega_{ad}} \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x})\} \\ \text{subject to: } c_k(\mathbf{x}) = 0 \quad (k = 1, \dots, K) \end{cases} \tag{1}$$

in which Ω_{ad} is the admissible domain, usually a subset of $\mathbb{R}^n$, possibly bound by inequality or interval constraints on the components of the optimization variable $\mathbf{x}$, and $\{f_j(\mathbf{x})\}$ $(j = 1, \dots, M)$ and $\{c_k\}$ $(k = 1, \dots, K)$ are the cost and (functional) constraint functions. We assume that the possible bound or interval constraints on the optimization variables are accounted for by a special parameterization as in Section 4. Hence we refer here to nonlinear constraints of other types, that we designate as "functional constraints", in the sense that they are calculated together with the cost functions by some nonlinear procedures, subject or not to the solution of a system of partial-differential equations (PDEs).

Note that bound or interval (functional) constraints can always be formulated as equality constraints by the introduction of slack variables, thus increasing the dimension n, and in fact, this technique is used and made explicit in the present application (see Section 4). Hence the above formulation is fairly general.

For the above multi-objective problem, a design-point $\mathbf{x} \in \Omega_{ad}$ is said to "dominate in efficiency" the design-point $\mathbf{y} \in \Omega_{ad}$, if and ony if (iff) its performance is at least equal to the performance of $\mathbf{y}$ with respect to (w.r.t.) all cost functions,

$$\forall j = 1, \dots, M : \quad f_j(\mathbf{x}) \leq f_j(\mathbf{y}) \tag{2}$$

and strictly superior w.r.t one cost function at least (strict inequality for one j at least). Clearly, this notion defines a relationship of partial order. The "Pareto set" is the subset of the admissible domain made of the non-dominated design-points, and the "Pareto front" its image in function space. Strictly speaking, the Pareto set and front constitute the mathematical solution to the problem (1), and determining these sets is considered by many as the ultimate designer's graal.

However this viewpoint may be nuanced in the context of complex multi-disciplinary optimization often named concurrent engineering. For purpose of discussion, let us consider the somewhat idealized, nevertheless instructive situation in which a given population of design-points is evaluated several times w.r.t. an ordered list of cost functions $\{f_1, f_2, \dots\}$ assumed to all be unimodal, convex and differentiable. The kth evaluation consists in identifying the Pareto set S_k w.r.t. the first k functions, $\{f_1, f_2, \dots, f_k\}$ with the understanding that S_1 is the singleton made of the minimizing design-point of f_1 alone, which is unique by unimodality. Then we have the inclusions

$$S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k \dots \tag{3}$$

(see Appendix A for a quick proof of this classical result). Hence each new evaluation with a larger number of cost functions can only have the effect of enlarging the Pareto set. One could imagine some absurd situation in which at the end, no design point from an original finite population could be excluded by the Pareto-optimality criterion, and the entire given population would constitute the Pareto set itself [1]. Without going to this unrealistic extreme, we conclude that the Pareto set becomes less and less informative as the number of cost functions increases, and this is counter-productive in the design of a complex engineering system. An alternate conclusion is that if the number of cost functions is increased, in a numerical context, the population of candidate design-points to be tested should be enriched substantially, thus adding another element that increases the computational demand to determine the Pareto set.

Let us return to (1) in the case of a typical engineering system for which the runtime of the computational procedure that evaluates cost and constraint functions is not negligible. Then, this

[1] This would be the case for the finite population $\{\mathbf{x}_i\}$ $(i = 1, \dots, N)$, w.r.t. the cost functions $f_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_j\|^2$ $(j = 1, \dots, N)$.

factor may critically condition the type of potential methods to be used to determine the Pareto set. Even in case of a moderate runtime, say of the order of the minute, the approximate determination of the Pareto set may be computationally demanding enough to exclude the practitioner's interactive action in the course of the optimization process. Anticipating Section 7, in the optimization of an aircraft flight-performance considered in this report, the average runtime for one evaluation on a laptop of the sole cost functions, without their gradients, has been measured to be about 30 s with our software, and for the numerical determination of the Pareto front associated with two cost functions by the classical NSGA-II [4] to more than 22 hrs for a population of 50 individuals. Hence developing computationally efficient optimization strategies in such a context remains of great importance, particularly in the perspective of the treatment of systems governed by partial-differential equations (PDEs), most often out of reach today.

Nevertheless, if the Pareto set has been identified numerically, even laboriously, the decision by the designer of which non-dominated design-point to elect is still to be made. In a commonly-used basic decision scheme, the choice is made by evaluating subsequently an additional criterion at the Pareto-optimal design-points, and choosing one of the best points w.r.t. this new criterion. In such a scheme, the decision is made in two steps. The introduction of an auxiliary criterion, in effect, relies on a hierarchical split made between the cost functions for which computing the Pareto front is consented, designated as "primary cost functions" hereafter, and the new cost function (or functions) used to finalize the decision, designated as "secondary cost function(s)". The prioritized multi-objective optimization method promoted here generalizes this split in a more elaborate and systematic way and has been demonstrated to be far more efficient than the basic decision scheme in several applications to engineering.

Our general method includes two optimization phases:

1. First Phase: numerical identification of the Pareto front associated with the constrained optimization of the "primary cost functions", and the election of a starting point $\mathbf{x}_A^\star$.

2. Second Phase: one-shot computation of a continuum of Nash equilibria originating from $\mathbf{x}_A^\star$ permitting to reduce the "secondary cost functions" while best preserving the Pareto-stationarity of the primary cost functions.

The construction of the algorithm permitting to achieve this is briefly recalled in Section 2 for sake of completeness, while more detailed descriptions can be found in several publications: the standard method, associated software, and first case studies were initially introduced in [8, 6]; the optimization test-case of the Super-Sonic Business Jet (SSBJ) was also presented in [9]; [11] relates to the sizing of an aluminum sandwich panel with honeycomb core to maximize its strength under bending loads or a blast and involved four objective functions given by analytical models. Finally, the algorithm was again defined in details in [7] along with convergence proofs, and some summary of accomplished case studies. The latter publication is the most complete reference.

The case study considered presently is also one of multi-objective optimization of the flight performance of an aircraft. However here the aircraft is transonic, of A320-type, and the goals of the optimization are defined differently. The evaluation of the cost functions was accomplished by using the open-source software FAST-OAD developed by ONERA and ISAE-SUPAERO (see Section 4).

In the base algorithm proposed in `https://mgda.inria.fr`, the user provides procedures for computing the cost and constraint functions. Here the algorithm has been split to include a preliminary "Design of Experiment" permitting to execute a number of evaluations, those necessary to construct the initial meta-models, in a batch mode, and limiting the number of calls to the evaluation procedure to one per computed Nash equilibrium point. These modifications are detailed in Section 5. Lastly, the results of the numerical campaign are described in Section 7 where the efficiency gains are analyzed in details, before some concluding remarks are drawn in Section 8.

2 Base algorithm

From here and throughout, a hierarchy between the cost functions is introduced according to the running index j:

1. Primary cost functions: $\{f_j(\mathbf{x})\} \quad (j = 1, \ldots, m)$
2. Secondary cost functions: $\{f_j(\mathbf{x})\} \quad (j = m+1, \ldots, M)$

Objective and cost functions are all assumed to be smooth, say $C^2(\Omega_{ad})$.

A first phase of optimization (omitted here) has led to the election of a design-point $\mathbf{x}_A^\star$ in the open interior of the admissible domain Ω_{ad} that is Pareto-optimal for the minimization of the sole primary cost functions under the constraints.

The second phase of optimization is aimed to construct a continuum of Nash equilibria $\{\overline{\mathbf{x}}_\varepsilon\}$ parameterized by ε ($0 \leq \varepsilon \leq 1$) in way such that: firstly, $\overline{\mathbf{x}}_0 = \mathbf{x}_A^\star$ (compatibility), and secondly, for ε sufficiently small, the Pareto-optimality condition of the primary cost functions remains $O(\varepsilon^2)$, whereas the secondary cost functions decrease linearly with ε (or faster).

2.1 Preliminary calculations

The preliminary calculations have been defined in full in [8, 7] and only summarized here. They are made of three main steps:

- Definition of a convex "Primary Steering Function" $f_A^+(\mathbf{x})$
- Definition of a "Territory Splitting"
- Definition of a "Secondary Steering Function" $f_B(\mathbf{x})$.

The cost functions are all assumed to be strictly positive. This may require the preliminary application of an adequate exponential transform, which preserves regularity and sense of variation of the function. This permits us to consider gradients in the logarithmic form, and thus examine the variations of these functions in terms of proportions from their starting values.

The Pareto-stationarity condition satisfied at $\mathbf{x} = \mathbf{x}_A^\star$ is expressed as follows:

$$\sum_{j=1}^{m} \alpha_j^\star \frac{\mathbf{P}\nabla f_j^\star}{f_j^\star} = 0 \tag{4}$$

where the supercript * on any symbol denotes throughout an evaluation at $\mathbf{x} = \mathbf{x}_A^\star$, ∇ is the gradient operator w.r.t. $\mathbf{x}$, $\{\alpha_j^\star\}$ $(j = 1, \ldots, m)$ are the known coefficients of a convex combination (to be determined), $\mathbf{P}$ is the projection operator onto the linear subspace orthogonal to all constraint gradients, $\{\nabla c_k^\star\}$ $(k = 1, \ldots, K)$. Then the primary steering function is defined as follows:

$$f_A^+(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j^\star \frac{f_j(\mathbf{x})}{f_j^\star} + \frac{c}{2} \left\| \mathbf{x} - \mathbf{x}_A^\star \right\|^2 \tag{5}$$

where c is a nonnegative constant adjusted to enforce the local convexity of $f_A^+(\mathbf{x})$ and the associated Lagrangian. Then, by virtue of (4), the primary steering function is stationary and convex at $\mathbf{x}_A^\star$, and therefore achieves a local minimum.

The territory splitting is based on the orthogonal decomposition of the following $n \times n$ reduced Hessian matrix [10, 8]:

$$\mathbf{H}'_A = \mathbf{P}\mathbf{H}_A^{+,\star}\mathbf{P} = \mathbf{\Omega}\mathcal{H}\mathbf{\Omega}^t \tag{6}$$

where $\mathbf{H}_A^{+,\star} = (\nabla^2 f_A^+)^\star = \mathbf{H}_A^\star + c\mathbf{I}_n$ and $\mathbf{H}_A^\star = \nabla^2 f_A^\star$. Matrix $\mathbf{H}'_A$ is positive semi-definite, of rank $n - K$ exactly since matrix $\mathbf{H}_A^{+,\star}$ is strictly-positive definite by convexity fix. The eigenvectors, column-vectors of matrix $\mathbf{\Omega}$, are conventionally arranged in a special way: the first K are associated

with the directions of the constraint gradients (null space of $\mathbf{P}$), and the remaining ones are ordered by decreasing eigenvalue. In this way, the tail eigenmodes are associated with the smaller sensitivities of the primary steering function. The splitting is associated with the following change of variables:

$$\mathbf{x} = \mathbf{x}_A^\star + \mathbf{\Omega} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} := \mathbf{X}(\mathbf{u}, \mathbf{v}) \tag{7}$$

where $\mathbf{u} \in \mathbb{R}^{n-p}$, $\mathbf{v} \in \mathbb{R}^p$, and p is an integer free parameter to be chosen such that and $1 \leq p < n - K$. In this setting, the following notations are used:

$$\mathbf{\Omega} = (\mathbf{\Omega_u}\ \mathbf{\Omega_v}), \quad \mathbf{\Omega_u} = \begin{pmatrix} \mathbf{\Omega_{uu}} \\ \mathbf{\Omega_{vu}} \end{pmatrix}, \quad \mathbf{\Omega_v} = \begin{pmatrix} \mathbf{\Omega_{uv}} \\ \mathbf{\Omega_{vv}} \end{pmatrix} \tag{8}$$

and the following scaled logarithmic gradients are associated with the secondary cost functions:

$$\mathbf{g}_j = \mathbf{S}^{-\frac{1}{2}} \frac{\nabla_{\mathbf{v}} f_j^\star}{f_j^\star} = \mathbf{S}^{-\frac{1}{2}} \frac{\mathbf{\Omega}_{\mathbf{v}}^t \nabla f_j^\star}{f_j^\star} \qquad (j = m+1, \ldots, M) \tag{9}$$

where the scaling matrix $\mathbf{S} = (\nabla_{\mathbf{v}} \nabla_{\mathbf{v}}^t f_A^+)^\star$ is the lower $p \times p$ diagonal block of matrix $\mathcal{H}$ of (6), positive-definite by convexity fix. Hence these gradients reflect the sensitivities of the secondary cost functions when $\mathbf{v}$ varies, and a scaling by elements related to the primary steering function as well. Following the strategy employed in the Multiple Gradient Descent Algorithm (MGDA) [8], one identifies the minimum-norm element $\boldsymbol{\omega}_B^\star$ in the convex hull of the gradients $\mathbf{g}_j$:

$$\boldsymbol{\omega}_B^\star = \sum_{j=m+1}^{M} \alpha_j^\star \mathbf{g}_j \tag{10}$$

and the coefficients $\{\alpha_j^\star\}$ of this convex combination are used to define the secondary steering function:

$$f_B(\mathbf{x}) = \sum_{j=m+1}^{M} \alpha_j^\star \frac{f_j(\mathbf{x})}{f_j^\star} \tag{11}$$

and this completes the preliminaries.

2.2 Nash game formulation

One defines a continuation parameter ε ($0 \leq \varepsilon \leq 1$) and the auxiliary cost function:

$$f_{AB}(\mathbf{x}) = (1 - \varepsilon) f_A^+(\mathbf{x}) + \varepsilon f_B(\mathbf{x}) \tag{12}$$

For every fixed value of ε, two virtual players, *Player A* and *Player B*, respectively controling the sub-vectors $\mathbf{u}$ and $\mathbf{v}$ in (7), are involved in a Nash game defined by the following strategies:

– Strategy A: *Player A* attempts to minimize $f_A^+\big(\mathbf{X}(\mathbf{u}, \mathbf{v})\big)$ by the strategy $\mathbf{u}$, subject to the equality constraint $\mathbf{c}\big(\mathbf{X}(\mathbf{u}, \mathbf{v})\big) = 0$, and by accounting for *Player B*'s fixed strategy $\mathbf{v}$,

whereas:

– Strategy B: *Player B* attempts to minimize $f_{AB}(\mathbf{X}(\mathbf{u}, \mathbf{v}))$ by the strategy $\mathbf{v}$, subject to no constraints, but by accounting for *Player A*'s fixed strategy $\mathbf{u}$.

2.3 Theoretical results

The following results have been established [8, 7]:

- The Nash equilibrium point $\overline{\mathbf{x}}_\varepsilon$ exists for all sufficiently small ε, and

$$\overline{\mathbf{x}}_0 = \mathbf{x}_A^\star \tag{13}$$

  an essential property referred to as compatibility or consistency.

- If $\phi_A(\varepsilon) = f_A^+(\overline{\mathbf{x}}_\varepsilon)$, $\phi'_A(0) = 0$ and

$$f_A^+ = 1 + O(\varepsilon^2) \tag{14}$$

  which reflects the fact that, as ε varies, the Pareto-stationarity of the primary cost functions is degraded by $O(\varepsilon^2)$ only.

- If $\phi_B(\varepsilon) = f_B(\overline{\mathbf{x}}_\varepsilon)$, $\phi'_B(0) = -\sigma_B$, where $\sigma_B = \|\boldsymbol{\omega}_B^\star\|^2 \geq 0$. Hence, the secondary steering function $f_B(\mathbf{x})$ decreases linearly as ε increases, and by virtue of properties of the MGDA construction, all the secondary cost functions do as well.

Thus the objectives of our construction are achieved.

These theoretical results are illustrated in Figure 1 in the case of two primary convex cost functions $(f1, f2)$ whose associated Pareto front is represented by a convex red curve ("Primary Pareto Front, PPF), and an arbitrary number of secondary cost functions. The continuum of Nash equilibria is represented by the green curve, tangent to the PPF at the chosen starting point $\mathbf{x}_A^\star$.

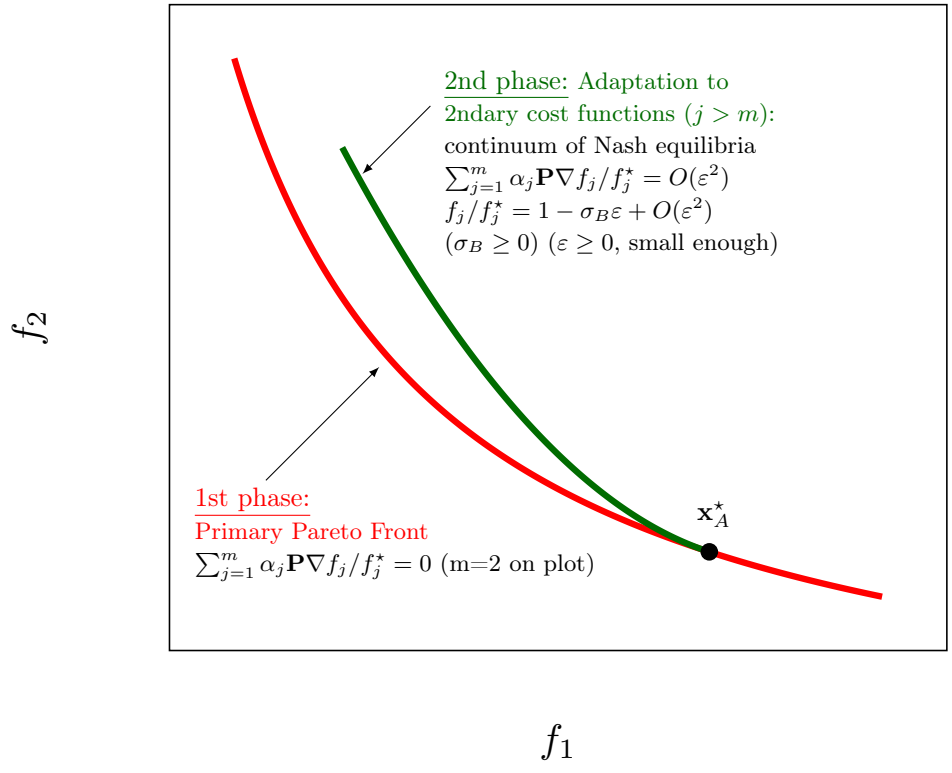


Figure 1: Primary Pareto Front (in red) and Continuum of Nash Equilibria (in green) in the case of $m = 2$ primary cost functions (f_1, f_2) and an arbitrary number of secondary cost functions

In practice, the algorithm is applied to local quadratic metamodels as described in [6]. Some implementation variations have been introduced to gain efficiency. These are defined in Section 5.

3 FAST-OAD simulation software and coupling with the MGDA software

3.1 The FAST-OAD software

FAST-OAD (Future Aircraft Sizing Tool – Overall Aircraft Design) is a framework for performing rapid Overall Aircraft Design. The software was developed jointly by ISAE-SUPAERO[2] and ONERA[3] for the preliminary design, analysis and optimization of aircraft. It is a dimensioning code that integrates models describing, with appropriate fidelity levels, the different physics involved in aircraft design: aerodynamics, propulsion, mass, geometry, performance, flight qualities, certification requirements, etc. FAST-OAD makes these different disciplines interact through multidisciplinary analysis and optimization techniques to design and evaluate an aircraft that meets the specifications defined by the user.

The software relies on the OpenMDAO [12] framework. It allows easy switching between models for a same discipline, and also adding/removing/developing models to match the user's needs. Details can be found in [3] and in the official documentation: `https://fast-oad.readthedocs.io/en/v1.3.5/` . See `https://github.com/fast-aircraft-design/FAST-OAD` for installation of the Python software. This question is revisited in Section 3.3.

3.2 The mgda.inria.fr software platform

The MGDA platform (`https://mgda.inria.fr`) is intended to provide the user with information or software for multi-objective differentiable optimization. The platform is made of four components of increasing complexity:

Basic MGDA tool for finding a search direction. This component allows for computing a descent direction common to an arbitrary set of cost functions whose gradients are provided in situations other than Pareto stationarity.

Directions for solving a constrained problem. This component refers to the research report Inria "Quasi-Riemannian Multiple Gradient Descent Algorithm for constrained multiobjective differential optimization" [5] for solving constrained problems by using the basic MGDA tool.

Tool for solving a prioritized optimization problem, or Nash-MGDA. This component permits to solve a multi-objective optimization problem in which the cost functions are given in two sets:

- a primary set of cost functions subject to constraints for which a Pareto optimal point is provided by the user (after using the previous tool or any other multiobjective method, possibly an evolutionary algorithm)
- a secondary set of cost functions to be reduced while maintaining quasi Pareto optimality of the first set.

In this report, we use a slightly modified version of the base code permitting to externalise the initial calls to FAST-OAD necessary to build up the local metamodels.

Stochastic MGDA This component provides references to SMGDA, an extension of MGDA applicable to certain stochastic formulations.

The platform was developed by the SED Inria Department (Service for software Experimentation and Development), in particular by T. Kloczko, N. Niclausse, M. Vesin and J.Wintz. The build-up of this platform has required a focus on two software entities, namely MGDA itself, and the server providing its online counterpart experience, hereafter referred to as mgda-www.

[2] `https://www.isae-supaero.fr/en/`
[3] `https://www.onera.fr/en`

3.2.1 Technical considerations on MGDA

The MGDA base code has been developed using fortran 77 (for the core) up to 2003 (see C++ additions below) using the blas and lapack libraries as its sole dependencies. Its uses cmake as a build system and is either packaged as a rpm package, suitable for the red-hat family of linux distributions, or as a self-contained binary archived, suitable for most linux distributions. It is also packaged for macOS and windows platforms.

What follows describes the original Fortran implementation of MGDA along with some C++ additions later helpful to use this codebase in the context of the online platform.

The main code and subroutines are written in Fortran 77, except for one procedure. The execution follows the preparation steps indicated in details in [6] permitting to define the initial local metamodels for constraints and cost functions, including an orthogonal decomposition of the working space on which relies the territory splitting between the virtual players of the Nash games. A procedure that permits to identify the minimum-norm element in the convex hull of given gradients was written in C to permit recursivity.

C++ additions. In order to handle the variety of use cases exposed by the online platform, some features have been added to the original code base, namely to:

- handle a heterogeneous set of options
- add some parsing facilities
- implement a pilot to handle them all and interface the original fortran procedures
- intertwine these procedures with the pilot using ISO C BINDING.

Fortran's ISO C Binding module provides named constants, types and procedures that are useful in a mixed-language (C and Fortran) program.

3.2.2 Technical considerations on mgda-www

The online user interface over the software stack described before is implemented as a web application leveraging the nodejs asynchronous event- driven JavaScript runtime, for both its frontend part, and its backend part.

Frontend. Presented as a conventional website, each component follow the same structure:

- Synopsis: a generic description of the problem at hand
- Run: a set of forms to retrieve inputs and options from the user, actually run the code and present its output
- Example: more in-depth example runs to help the power user, or collaborator with an access on the binaries for its own academic use, to implement or interface its own use case with the mgda procedures.

Backend. The backend is the component that connects the frontend of mgda-www together with the actual code.

However, in order to make these calls to the (server side) mgda binaries consistent with the the various step of the wizards implemented by the frontend, the platform implements and leverages a RESTful Application Programming Interface (API). Put simply, a REST API, allows a client (a web browser) to send messages to a server in order for it to perform specific actions and return their result asynchronously to be rendered with the web page.

The RESTful implementation in mgda-www is very basic. It consists in the following routes:

- upload: This allows the user to upload its own dataset, to be fed either to the MGDA or Nash-MGDA procedures.

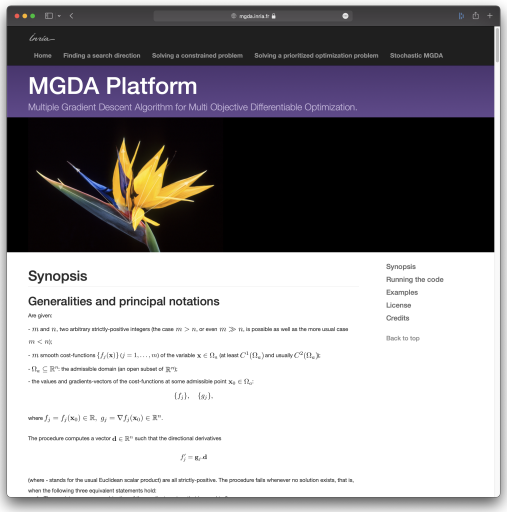


Figure 2: Index page for the first component of the platform

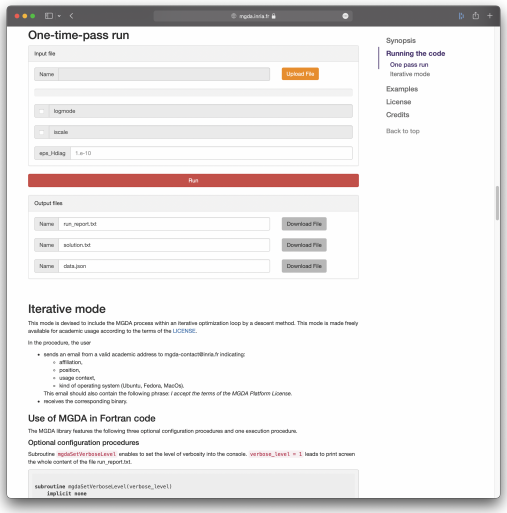


Figure 3: Forms to allow interacting with the server side binaries

- compute: This checks and validates the presence of all required files before spawning and executing the relevant procedure call at hand, finally retrieving its output and presenting them either in the frontend or as a download option.
- download: This allows the user to download the output file of the previous remote procedure call.

In the standard usage of the platform, the user provides fortran-compatible procedures that permit the separate computation of *(i)* the primary cost functions, *(ii)* the secondary cost functions, and *(iii)* the constraints functions that are called during the computation of the continuum of Nash equilibria to upgrade the metamodels. However here, all three sets are computed together by one call to FAST-OAD, and this is done only once per equilibrium point. (See Section 6.)

3.3 Coupling the two platforms

This coupling has been realized in compliance with the instructions of the document [2] by using the associated library **call_py_fort**.

A FAST-OAD run wraps up an OpenMDAO problem, which is, in a nutshell, the assembly of components that each have input and output variables. Of course, the outputs of some component can be the inputs of some other ones, so that the whole system can be solved. FAST-OAD allows to define the problem to solve (or to optimize) through a configuration file that makes easy to add/remove/replace any component. By doing that, the input data of the problem can be very different from one problem to the other, but FAST-OAD comes with facilities to build the needed input data files. A FAST-OAD problem can be fully run from a command line interface or from the Python API.

Usage of Python API, including pre-processing and post-processing utilities are currently provided through Python notebooks.

A typical run of FAST-OAD uses following types of user files:

Configuration file (.yml): This file defines the OpenMDAO problem by defining:

- what components will be in the problem
- the files for input and output data

- the problem settings
- the definition of the optimization problem if needed (not used here)

Input/output data files (.xml): These files contain the information of the variables involved in the system model:

- The input file contains the global inputs values required to run all the model. The user is free to modify the values of the variables in order that these new values are considered during a model run.
- The output file contains all the variables (inputs + outputs) values obtained after a model run.

Mission definition file (.yml): This file defines the mission profile that is used for computing aircraft performances using a mass point formulation. It is used only if a flight mission is simulated and none of the bundled mission profiles is used, which is the current case.

To the end of coupling FAST-OAD together with MGDA in order to address the problem at hand, these files have been tailored as follows:

- The configuration file defines the overall aircraft design problem in a classical way: the aircraft is sized though an iterating process devised to achieve the targeted mission while respecting the specified values for input parameters. Simple, low-fidelity models are used for sake of computation speed.
- The input file (.xml) defines the values for all input parameters that are not modified in our process (e.g. passenger count, mission range, various models settings, etc).
- The mission definition file (.yml) defines the mission profile that is used for computing fuel consumption and climb duration. The targeted mission is a climb-cruise-descent sequence, followed by a diversion, a holding phase, and a final descent.
- FAST-OAD launcher (.py): This Python file allows MGDA to launch FAST-OAD. It provides a function that is called by the Fortran program, and launches a FAST-OAD computation. The function uses a dictionary as argument. This dictionary is populated by the Fortran program (using the `set_state` function from call_py_fort) prior to calling the Python function. The Fortran program retrieves the FAST-OAD results using the `set_state` function from call_py_fort.

4 Optimization test-case

We consider the optimization of six variables that are essential to the design of an aircraft of the Airbus A320-type, to maximize its flight performance measured by three cost functions two of which prioritized under several functional constraints calculated by the simulation software FAST-OAD.

4.1 Cost and constraint functions

Primary cost functions

- Fuel mass at take-off (by multiples of 10^4 kg): f_1;
- Operating empty weight, OEW (by multiples of 10^4 kg): f_2.

Secondary cost function

- Ascent time (by multiples of 10^3 s): f_3.

Functional constraints

- $0.05 \leq$ static margin ≤ 0.10
- wing span ≤ 36 (m).

4.2 Physical and dimensionless optimization variables

Six optimization variables subject to interval bound constraints are considered:

1. distance from nose to 25% of wing Mean Aerodynamic Chord (MAC), $\phi_1 = WMAC25 \in [16, 18]$, to be controlled by x_1,
2. wing aspect ratio, $\phi_2 = WAR \in [5, 11]$, to be controlled by x_2,
3. horizontal tail aspect ratio, $\phi_3 = HTAR \in [1.5, 6]$, to be controlled by x_3,
4. wing taper ratio, $\phi_4 = WTR \in [0, 1]$, to be controlled by x_4,
5. wing sweep angle at 25%MAC, $\phi_5 = WSWEEP25 \in [20, 30]$, to be controlled by x_5,
6. maximum take-off thrust, $\phi_6 = MTOT \in [100000, 140000]$, to be controlled by x_6.

These physical variables are all bound by interval constraints, namely:

$$a_i \leq \phi_i \leq b_i \quad (i = 1, \ldots, 6) \tag{15}$$

for user-specified $\{a_i\}$ and $\{b_i\}$. These constraints are handled via the parameterization chosen to define a simple one-to-one relationship between ϕ_i and x_i in a way such that:

- x_i varies in the interval $[-1, 1]$;
- ϕ_i is a monotone-increasing function of x_i from $\phi_i(-1) = a_i$ to $\phi_i(1) = b_i$.

The interval constraint on ϕ_i is equivalent to maintaining the function

$$\psi_i = \left(\phi_i - \frac{a_i + b_i}{2}\right) \Big/ \left(\frac{b_i - a_i}{2}\right) \tag{16}$$

in the interval $[-1, 1]$. This can be accomplished simply via a sine transform:

$$\psi_i = \psi_i(x_i) = \sin\left(\frac{\pi x_i}{2}\right), \quad x_i \in [-1, 1], \tag{17}$$

so we let:

$$\phi_i(\mathbf{x}) = \frac{a_i + b_i}{2} + \frac{b_i - a_i}{2} \sin\left(\frac{\pi x_i}{2}\right), \quad x_i \in [-1, 1]. \tag{18}$$

4.3 Slack variables and numerical formulation of constraints

Both constraints are formulated as equality constraints, say

$$\begin{cases} c_1(\mathbf{x}, x_7) = 0 & \text{for the interval constraint on static margin} \\ c_2(\mathbf{x}, x_8) = 0 & \text{for the inequality constraint on wing span} \end{cases}$$

where, for the moment, $\mathbf{x} \in \mathbb{R}^6$ is the vector of dimensionless optimization variables, whereas (x_7, x_8) are slack variables to be defined. Subsequently, (x_7, x_8) are integrated into the definition of the vector $\mathbf{x}$ thereafter handled as an 8-component vector in the numerical process.

The particular form given to (c_1, c_2) should reflect two objectives:

1. enforcing automatically the inequality constraints on state margin and wing span;
2. defining c_1 (resp. c_2) as a smooth, convex and monotone-increasing function of x_7 (resp. x_8) at least in the interval $[-1, 1]$.

Concerning the above second requirement, convexity is required to facilitate the back steering of these functions to 0 via a Newton-type iteration on the slack variables when a small violation of the constraint is observed. The functions are also expected to be convex in $\mathbf{x}$, or require only a small convexity fix. Monotonicity is not essential but convenient. It ensures that each constrained physical variable (static margin or wing span) varies in the same sense as its associated slack variable (resp. x_7 or x_8).

Interval constraint on static margin The second derivatives of the static margin w.r.t. the optimization variables have been approximated by central differences at the starting point $\mathbf{x}_A^\star$. All six axial derivatives were found negative:

$$\left\{\left(\frac{\partial^2 static:margin}{\partial x_i^2}\right)_0\right\} = (-0.100969, -32.5971, -32.6567, -57.5602, -71.5263, -74.8816)$$

where the subscript $_0$ refers to the starting point. Thus, the following dimensionless function that varies in the opposite sense is defined

$$g_1 = g_1(\mathbf{x}) = \frac{0.10 - static:margin}{0.05} = 2 - 20 \times static:margin$$

and the constraint is formulated as follows:

$$0 \leq g_1 \leq 1.$$

and handled simply via a sine function:

$$c_1(\mathbf{x}, x_7) = g_1(\mathbf{x}) - \sin\gamma_1 = 0, \quad \gamma_1 = \frac{1 - x_7}{K_1},$$

in which to ensure the positivity of the sine and its convexity in x_7 at least inside the macro lattice (that is here, $x_7 \in [-1, 1]$), the argument γ_1 should be maintained in $[0, \pi]$. Actually, the more stringent condition $\gamma_1 \in [0, \pi/2]$ will now be enforced by the choice of the constant K_1. For this, let

$$\theta_1 = \sin^{-1}(g_1)_0 \in [0, \pi/2]$$

and assume that $\theta_1 \leq \pi/4$; in the case $\theta_1 > \pi/4$, slightly different formulas would be used with identical efficiency.

At $\mathbf{x}_A^\star$, the condition $c_1 = 0$ must be enforced, and this requires that

$$\frac{1 - (x_7)_0}{K_1} = \theta_1 + 2k\pi, \text{ or } \pi - \theta_1 + 2k\pi.$$

Let us try

$$1 - (x_7)_0 = K_1\theta_1.$$

At the upper limit of the macro lattice, $x_7 = 1$, and $\gamma_1 = 0$. At the lower limit, $x_7 = -1$, and $\gamma_1 = 2/K_1 = \pi/2$ provided $K_1 = 4/\pi$; then

$$(x_7)_0 = 1 - K_1\theta_1 = 1 - \frac{4\theta_1}{\pi} \in [0, 1].$$

At intermediate points, $\gamma_1 \in [0, \pi/2]$ as required. Finally:

$$\gamma_1 = (1 - x_7)\frac{\pi}{4}.$$

Inequality constraint on wing span The axial second derivatives of wing span have been approximated by central differences at the starting point, and all six were found to be negative:

$$\left\{\left(\frac{\partial^2 wing:span}{\partial x_i^2}\right)_0\right\} = \left(-2.064\times 10^{-2}, -15.9242, -1311.66, -1311.92, -1424.07, -1522.41\right).$$

This suggests to define c_2 in a way such that it varies in the opposite sense. One lets

$$g_2 = 1 - \frac{wing:span}{36}$$

and the constraint is formulated as follows:

$$0 \le g_2 \le 1.$$

Mimicking the development of the previous subsection, one lets:

$$c_2(\mathbf{x}, x_8) = g_2(\mathbf{x}) - \sin\gamma_2, \quad \gamma_2 = (1-x_8)\frac{\pi}{4},$$

and

$$\theta_2 = \sin^{-1}(g_2)_0, \quad (x_8)_0 = 1 - K_2\theta_2 = 1 - \frac{4\theta_2}{\pi} \in [0,1],$$

in which the assumption $\theta_2 \le \pi/4$ was made.

5 Design of Experiment

A Design of Experiment is carried out in the preparation phase to calculate a database used to elaborate initial metamodels for the constraint and steering functions. Lattices of points in $\mathbb{R}^8$ are defined, and function values at these points are calculated. The resulting database serves to the construction of the initial quadratic metamodels. More specifically, the data are used to approximate the gradients and the diagonal elements of Hessians by central differencing, and to approximate the other second derivatives by least squares.

The lattices are first constructed in $\mathbb{R}^6$ by discretizing the 6 physical variables and five "physical functions" (the 3 cost functions, static margin and wing span) are evaluated at these points by calls to FAST-OAD. The corresponding database is then extended to dimension 8 by discretizing also the slack variables. The latter do not affect the physical functions and only affect the constraint functions (c_1, c_2). As a result, for each 8-component vector $\mathbf{x}$, we dispose of the values of the 3 cost functions (f_1, f_2, f_3) and 2 constraint functions (c_1, c_2).

The starting values of the physical variables $\{(\phi_i)_0\}$ are provided by the user, yielding the $\{(\psi_i)_0\}$, from which the components of $\mathbf{x}_A^\star$ are calculated:

$$(x_i)_0 = \frac{2}{\pi}\sin^{-1}(\psi_i)_0. \tag{19}$$

For each component, the distance to the limits is calculated:

$$d_i = \min((x_i)_0 + 1, 1 - (x_i)_0) \tag{20}$$

as well as $d = \min_i d_i$ (ℓ_1-distance to the boundary). A somewhat smaller distance $h_{\text{cut}} \le d$ is chosen.

Micro lattice in $\mathbb{R}^6$. This lattice is chosen to approximate at $\mathbf{x}_A^\star$ the first partial derivatives of various functions and their Hessian diagonal elements. The approximation is made by central differences with stepsize $h = 10^{-2}$. This lattice is thus centered at $\mathbf{x}_A^\star$ and contains 16 additional points.

Macro and medium-size lattice in $\mathbb{R}^6$. These lattices are used to compute the non-diagonal elements of Hessians by least-squares, and thus provide a more global information. The macro lattice is centered at $0 \in \mathbb{R}^6$. For each pair of coordinates (x_i, x_j) (15 in total), 8 other points correspond to a uniform discretization of $[-\frac{1}{2}, \frac{1}{2}]^2$, and 8 other points to a uniform discretization of $[-1, 1]^2$. This lattice contains $15 \times 16 + 1 = 241$ points. The medium-size lattice has the same structure but is centered at $\mathbf{x}_A^\star$ (restricted to $\mathbb{R}^6$) and its largest size is $h_{\rm cut}$ instead of 1. This lattice also contains 241 points.

Note that by discretizing x_i at the points $0,\ \pm\frac{1}{2},\ \pm 1$, the corresponding sine function values are $0,\ \pm 1/\sqrt{2},\ \pm 1$ and these correspond to a Chebyshev distribution of points which is the recommended interpolation support in the so-called "best approximation".

In total, micro, medium-size and macro lattices contain $13 + 2 \times 241 = 495$ points.

Lattices extended to $\mathbb{R}^8$. In their extensions to $\mathbb{R}^8$ the numbers of datapoints are:

– 17 points for the micro lattice

– $28 \times 16 + 1 = 449$ points for the macro and medium-size lattices.

This makes a total of $17 + 2 \times 449 = 915$ data-points, out of which only 495 are actually evaluated by calls to the simulation procedure, and the other ones extrapolated from the parameterization since the slack variables do not affect the physical functions, but only the somewhat artificial constraint functions (c_1, c_2) through explicit formulas.

Weights and least-squares matrix. The least-squares procedure to approximate off-diagonal elements of Hessians has been described for the base algorithm in [6]. Two revisions were brought to this procedure impacting the least-squares matrix and the inversion process.

Firstly, since the database of points is not completely centered at $\mathbf{x}_A^\star$, the least-squares matrix is no longer scalar and has been recomputed. Secondly, the functions values have been weighted to reduce the influence of the points of the macro lattice that are very remote from $\mathbf{x}_A^\star$. The weights were defined to be 1 inside the medium-size lattice and to have the decay of the error-function of the ℓ_1-distance from it. The adjustment of the parameters was not really optimized.

The matrix is calculated and factorized once, and then used several times, firstly for the constraint metamodels, and subsequently for the two steering functions metamodels.

6 Coordination loop

The coordination loop has been described for the base algorithm in [6]. The discussion is here limited to the handling of the slack variables that has been revised to maintain the constraint functions rigorously to 0 throughout the determination of the continuum of Nash equilibria.

At the start of the procedure to compute a new Nash equilibrium point, the FAST-OAD software is called to evaluate the physical functions, as well as the cost and constraint dimensionless functions. Then the slack variables are updated to drive the cost functions (c_1, c_2) to 0, and this has the effect of upgrading the constraint metamodels. Then begins the coordination between the constrained minimization of the primary steering function f_A^+ by using the subvector $\mathbf{u}$ (for fixed $\mathbf{v}$) as the strategy of the virtual player A competing with the unconstrained minimization of the functionf_{AB} by using the subvector $\mathbf{v}$ (for fixed $\mathbf{u}$). The convergence of this coordination results in the definition of a new Nash equilibrium point. The procedure is continued until the coordination process fails to find a limit after a prescribed number of iterations, or because the defect of convexity makes the coordination itself inapplicable. Then the plotting procedure is launched using the available computed data.

7 Results

7.1 First phase of optimization: (f_1, f_2) Pareto set

A numerical approximation of the Pareto set was obtained by the classical NSGA-II using the pymoo toolbox [1]. The symbols • on Figure 4 indicate the corresponding Pareto front obtained after 2500 calls to FAST-OAD v. 1.3.4 which took about 21.5 hrs.

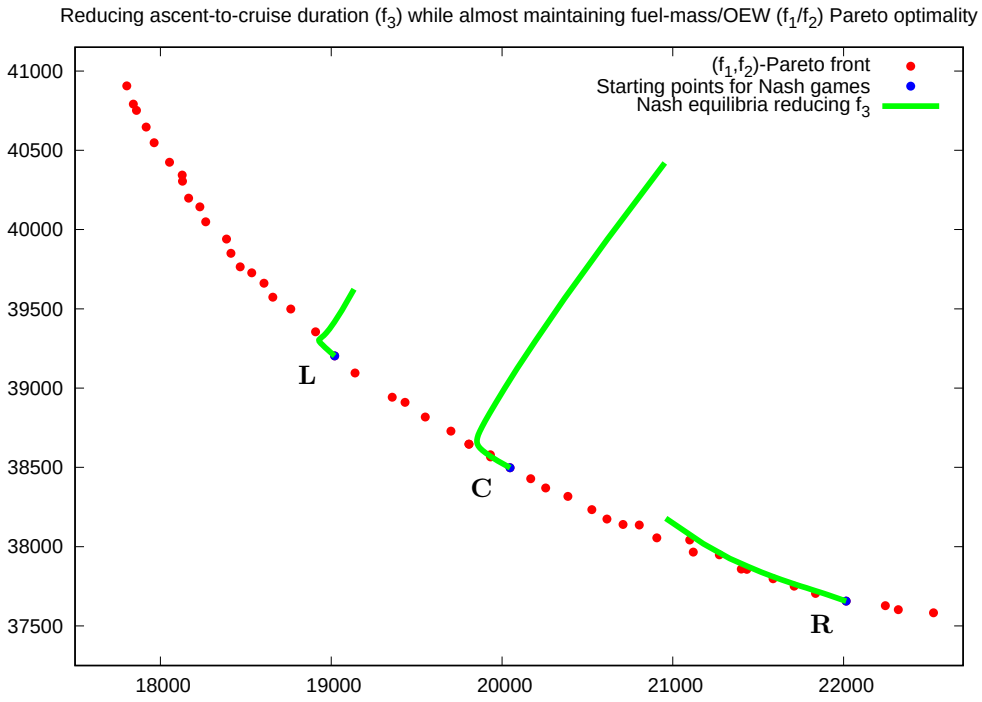


Figure 4: Pareto front associated with the primary cost functions (f_1, f_2) (red symbols); traces of three continua of Nash equilibria to reduce f_3 (green)

7.2 Design of Experiment

The continuum of Nash equilibria was calculated in three cases, indicated by the symbols L, C and R, on Figure 4 corresponding to three different starting points. To prepare each calculation, a database of 495 points about the starting point was computed each time providing values of (f_1, f_2, f_3) and $(c1, c_2)$ at 915 points according to the process detailed in Section 5. This calculation took about 4hrs on a laptop in the Case C; somewhat less in Case L, and more in Case R.

7.3 Second phase of optimization: Nash equilibria

7.3.1 Trace of continua in the (f_1, f_2) plane

In all three cases, we observe on Figure 4 that the trace of the continuum (in green) is tangent to the Pareto front at the starting point, as it should by theory. After a segment very close to the

front, the trace experiences an abrupt change in curvature. The performance achieved by these continua must now be assessed. The numerical process fails at some ε inferior but close to is the limit of convexity $\varepsilon_{\max}$ calculated by the numerical method.

7.3.2 Climb duration f_3 vs fuel mass f_1 along the continuum

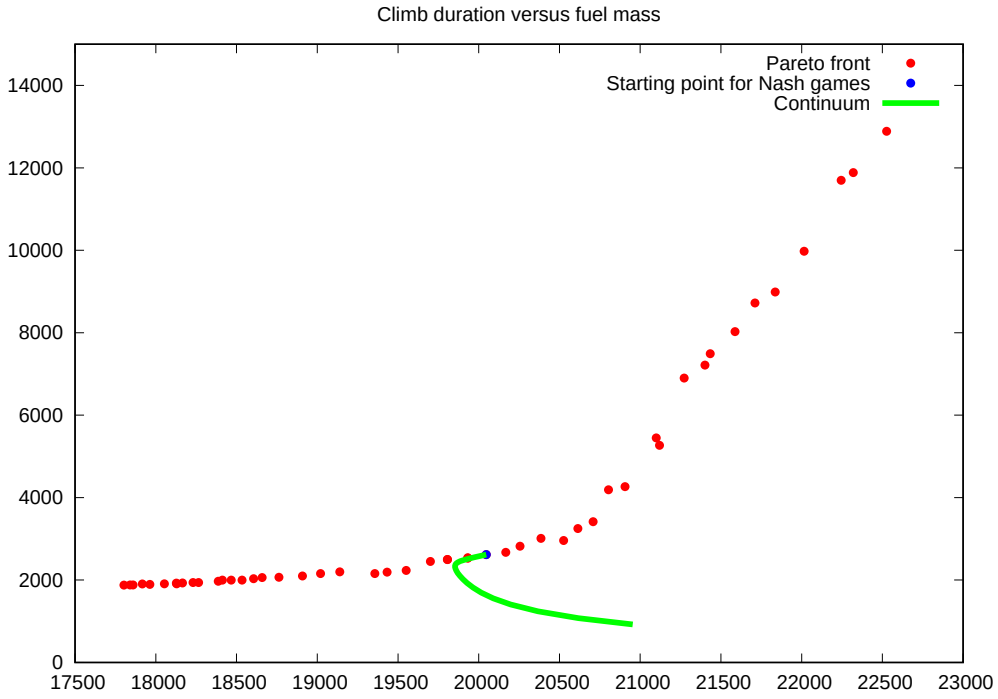


Figure 5: Climb duration f_3 vs fuel mass f_1 along the continuum - Case C

Not only the process has permitted to reduce the climb duration from the starting value, but it has also produced design-points that are more efficient in terms of this variable than all the design-points of the (f_1, f_2) Pareto front.

7.3.3 Three-dimensional visualizations

These visualizations depict the continua of Nash equilibria as curves in the (f_1, f_2, f_3) space (in green) in Cases L (upper one), C (center) and R (lower one).

In Figure 6, the continua are compared with the values of the triplet (f_1, f_2, f_3) attained at the points of the Pareto set w.r.t. (f_1, f_2) alone. In Figure 7, the continua are compared with the Pareto front associated with (f_1, f_2, f_3).

As already mentioned, the Pareto sets were numerically approximated by applying the classical NSGA-II algorithm [4] using the pymoo toolbox [1]. In the first case, 50 individuals were used, 50 generations computed, corresponding to 2500 calls to FAST-OAD, and the run time on a lap-top was 22 hrs. In the second case, 100 individuals were used, 100 generations computed, corresponding to 10000 calls to FAST-OAD, and the run time on a lap-top was 77 hrs.

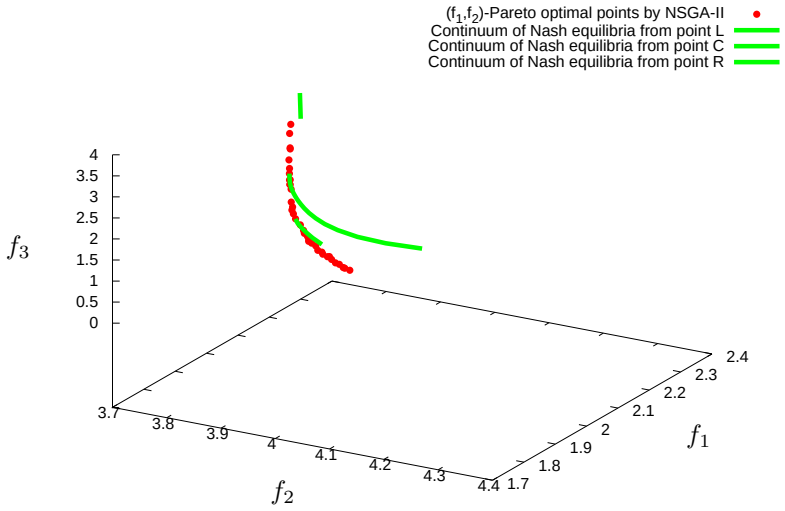


Figure 6: Values of (f_1, f_2, f_3) attained by the points of discrete Pareto set associated with (f_1, f_2) alone (red symbols), and the three continua of Nash equilibria initiated from it in Cases L (upper one), C (center) and R (lower one) (green curves)

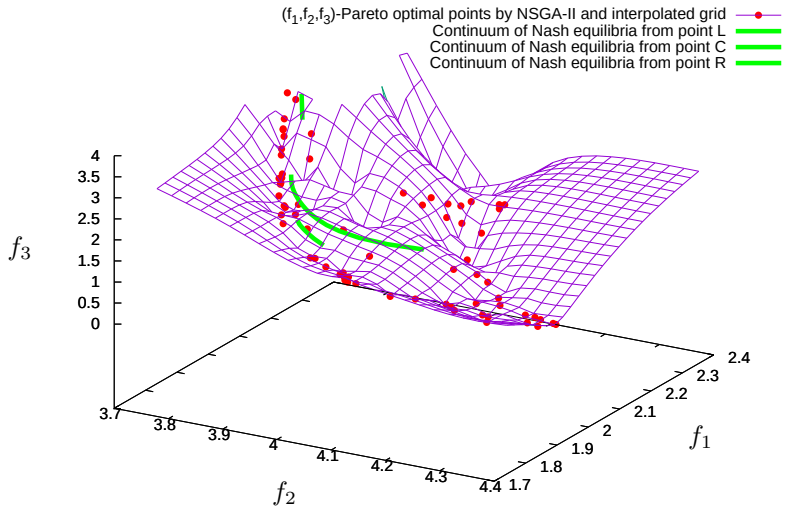


Figure 7: Pareto front associated with (f_1, f_2, f_3) (red symbols and interpolated grid), and the three continua of Nash equilibria initiated from it in Cases L (upper one), C (center) and R (lower one) (green curves)

In the Introduction, it was observed that when the cost functions are convex, the continuous Pareto set can only be enlarged by considering three cost functions instead of two. Here, discrete Pareto sets are considered for which a shift of the Pareto optimal points is possible; additionally, convexity is not guaranteed. Nevertheless the enrichment in red symbols from the first to the second figure is indicative of the enlargement.

The Nash equilibria are observed on Figure 7 to be above but close to the Pareto front associated with the whole set of cost functions, without prioritization. The paths appear directed towards local sinks of the secondary cost function f_3 in the (f_1, f_2, f_3) Pareto front visualized by the interpolated grid.

7.3.4 Cost Functions along continuum

The variation with ε of the three cost functions $f_1(\varepsilon)/f_1^\star$, $f_2(\varepsilon)/f_2^\star$ and $f_3(\varepsilon)/f_3^\star$ and the two steering functions $f_A^+(\varepsilon)$ and $\widetilde{f_B}(\varepsilon)$ is represented in Figures 8, 9 and 10 in Cases L, C and R respectively. In all three cases, the method is effective up to the end of the computed path which is only a part of the continuum, including the portion after the abrupt change in curvature. These plots illustrate the usual "funnel-shaped" pattern, systematically observed in other cases [8, 6, 9, 11, 7]: the primary steering function admits an initial horizontal tangent and increases moderately with ε, and so do the primary cost functions (f_1, f_2), which remain nearly Pareto-stationary; the secondary steering function admits a tangent with a strictly negative slope, and the secondary cost function f_3 remains very close to it, and decreases importantly.

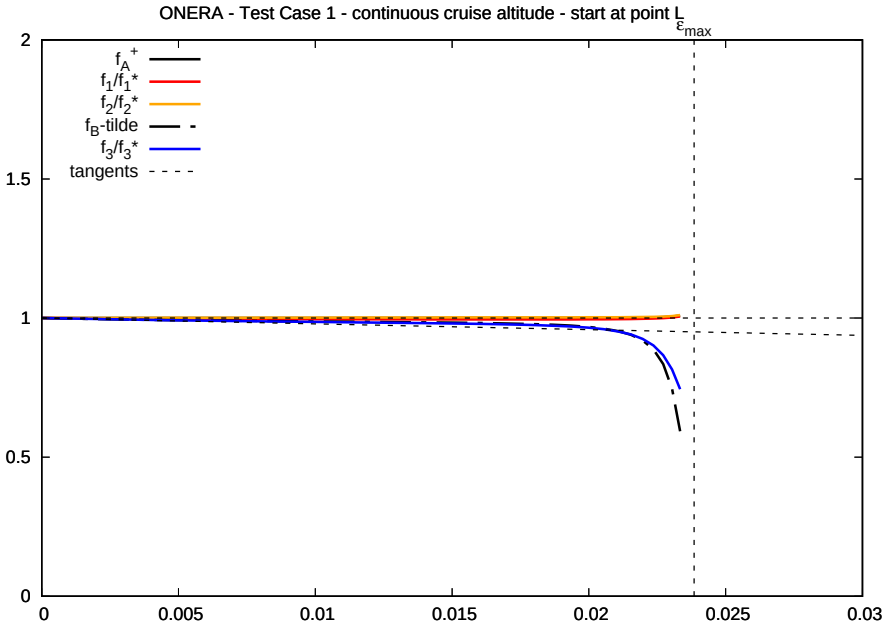


Figure 8: Cost functions along continuum as functions of ε - Case L

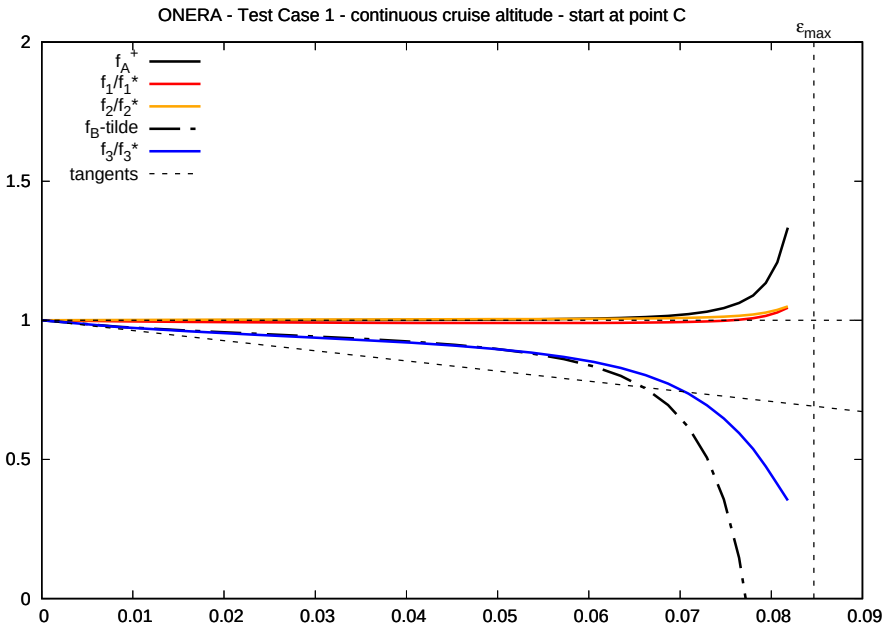


Figure 9: Cost functions along continuum as functions of ε - Case C

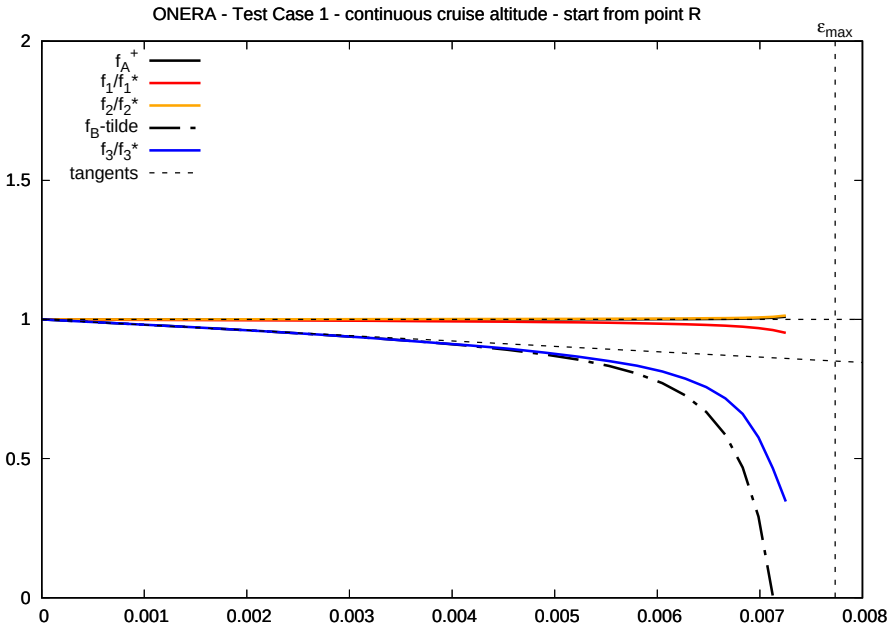


Figure 10: Cost functions along continuum as functions of ε - Case R

In each case, the designer could favor a point prior to the last computed Nash equilibrium point. Nevertheless, as an indication, Table 1 provides the percentage variation of the three cost functions realized at this point.

Case	f_1	f_2	f_3
L	+0.6	+1.1	-25.5
C	+3.9	+4.6	-62.9
R	-4.8	+1.4	-65.4

Table 1: Percentage variation of the three cost functions (fuel mass, OEW and climb duration) from starting point to lastly-computed point

7.3.5 Constraints along continuum

As mentioned in Section 6, the slack variables are upgraded after each Nash equilibrium point without alteration of the physical functions and this permits to maintain the constraint functions (c_1, c_2) to exactly 0. As a result, for the computed Nash equilibrium points the interval constraint on static margin and bound on wing span are satisfied throughout. This is evident from Table 2 which provides the first five and last five sets of values in Case C.

Nash equil. point	ϕ_1 (fuel) ($\times 10^4$ kg)	ϕ_2 (OEW) ($\times 10^4$ kg)	ϕ_3 (duration) ($\times 10^3$ s)	Static Margin ($\times 10^{-2}$)	Wing Span (m)
0	2.0047	3.8497	2.6175	5.7333	27.6631
1	1.9975	3.8541	2.5542	5.7322	27.8288
2	1.9933	3.8570	2.5147	5.7337	27.9297
3	1.9904	3.8591	2.4830	5.7347	27.9971
4	1.9885	3.8607	2.4570	5.7356	28.0447
5	1.9871	3.8622	2.4308	5.7365	28.0792
⋮	⋮	⋮	⋮	⋮	⋮
18	2.0090	3.9126	1.5560	5.8797	28.0927
19	2.0197	3.9300	1.4082	5.8301	28.0798
20	2.0367	3.9567	1.2423	5.8767	28.0647
21	2.0618	3.9940	1.0757	5.9505	28.0492
22	2.0953	4.0420	0.9233	6.0008	28.0247
$\Delta_{\max}$	+3.9%	+4.6%	-62.9%		

Table 2: Physical cost functions and constraints along the continuum in Case C

The interval constraint on the static margin and the bound constraint on the wing span are satisfied throughout.

7.3.6 Dimensionless and physical optimization variables along the continuum

These variables are observed to remain smooth all along, except for the slack variables at the approach of the convexity break point.

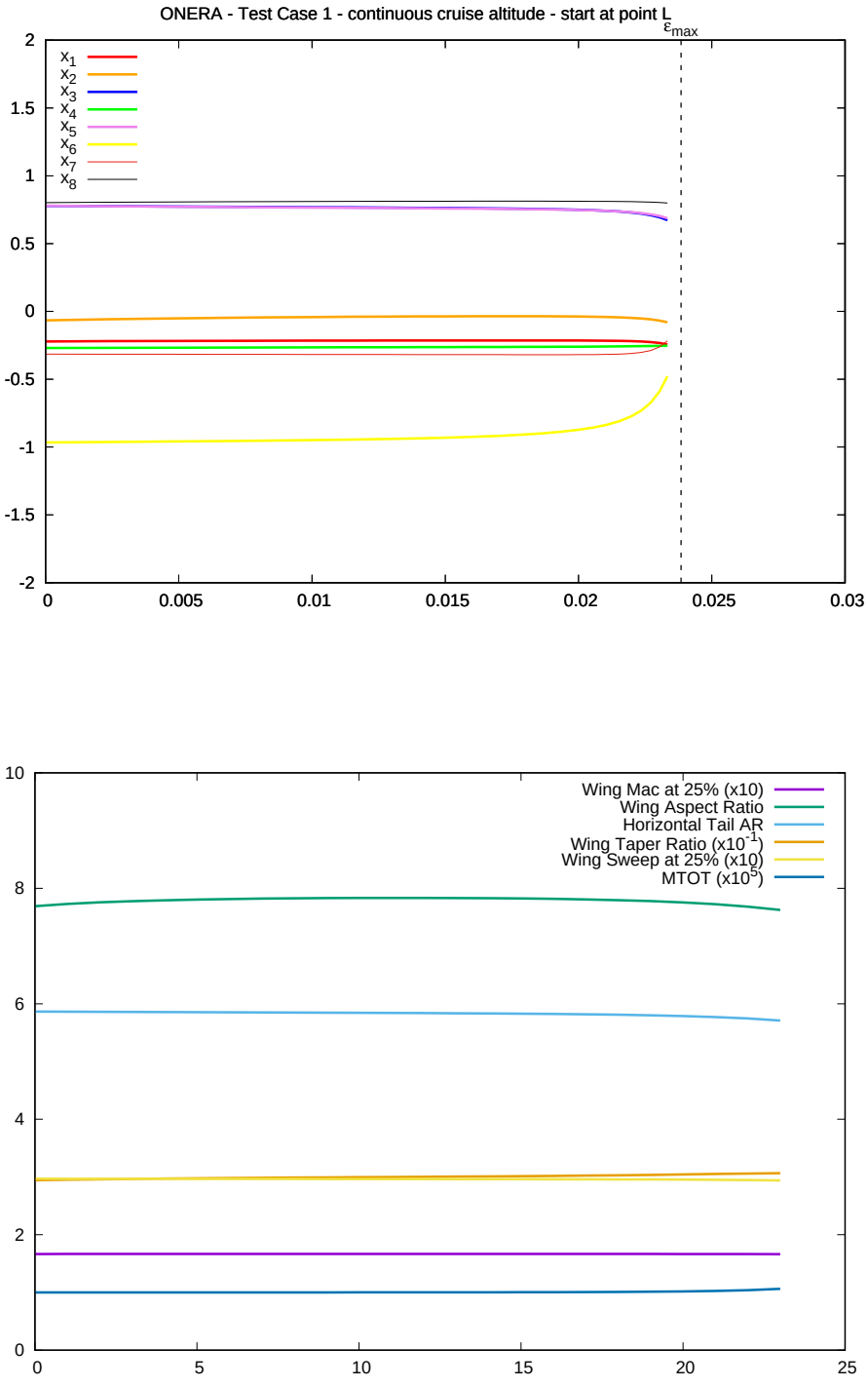


Figure 11: Dimensionless and physical optimization variables along the continuum - Case L
Mostly influential: Horizontal Tail Aspect Ratio (to be reduced); Wing Taper ratio and Maximum Take-Off Thrust (to be increased)

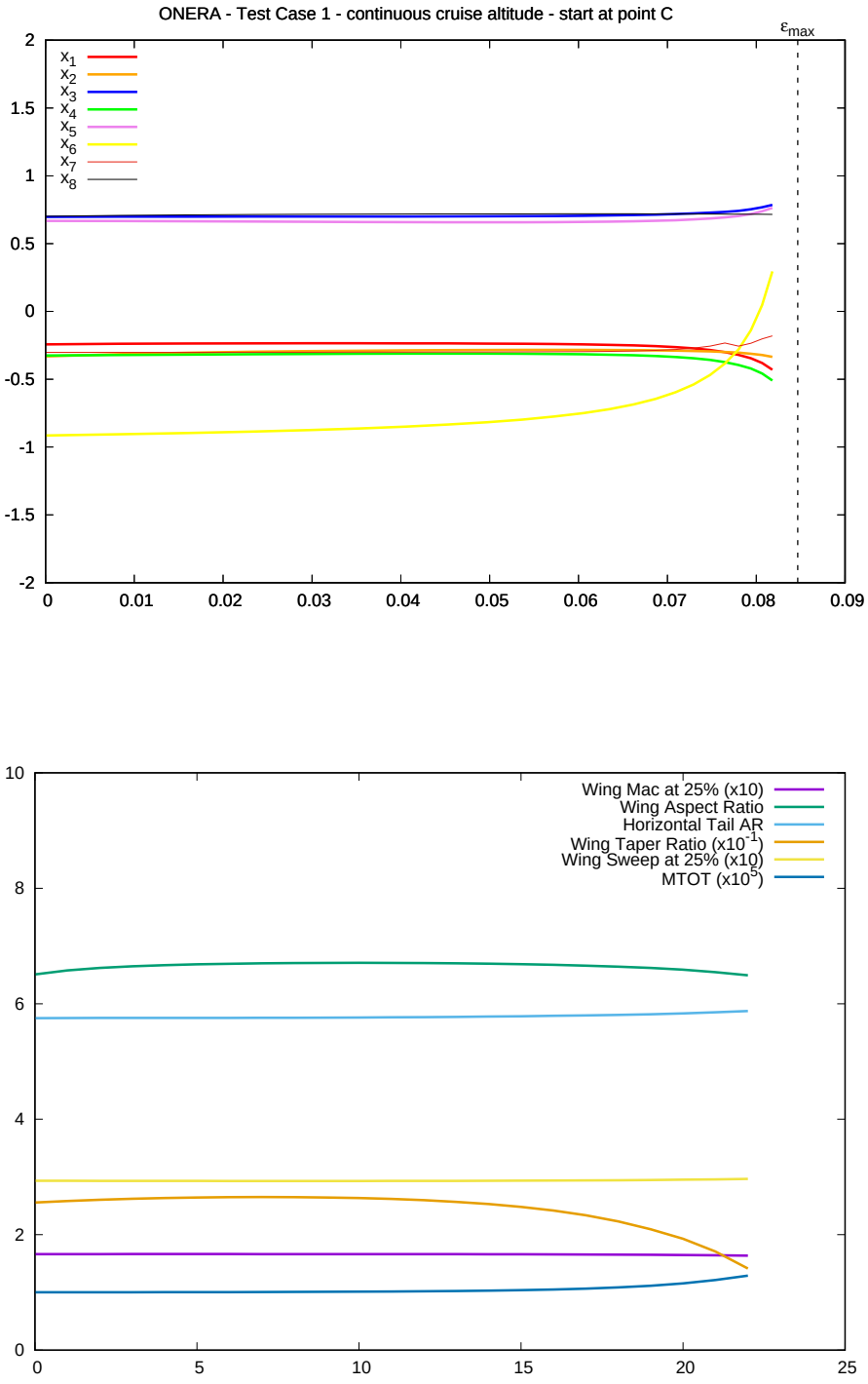


Figure 12: Dimensionless and physical optimization variables along the continuum - Case C
Mostly influential: Wing Taper Ratio (to be reduced); Maximum Take-Off Thrust (to be increased)

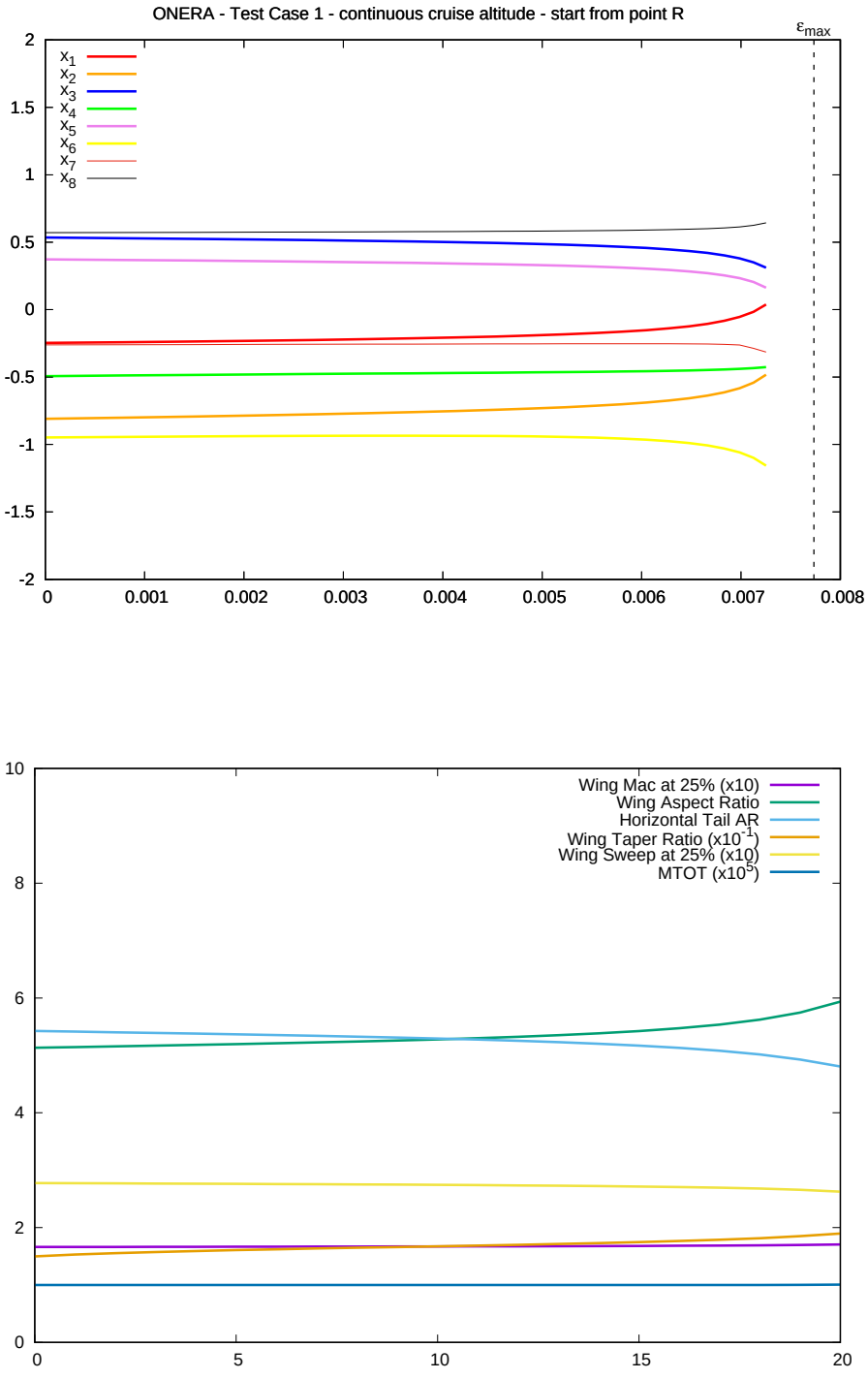


Figure 13: Dimensionless and physical optimization variables along the continuum - Case R
Mostly influential: Wing Aspect Ratio and Taper Ratio (to be increased); Horizontal Tail Aspect Ratio (to be reduced)

8 Conclusions

We have considered a simplified multi-objective optimization of the flight performance of an aircraft, of Airbus A320 type. The problem involved:

- six optimization variables affecting wing geometry, mean aerodynamic chord, and maximum take-off thrust;
- two cost functions (f_1, f_2) (fuel mass at take-off, Operating Empty Weight) considered primary;
- one cost function f_3 (climb duration), considered secondary;
- an interval constraint on the static margin;
- a bound constraint on the wing span.

The notion of prioritization relied on the split made between primary and secondary cost functions. The performance was calculated by a call to the open-source code FAST-OAD, and the optimization conducted using a customized version of the Nash-MGDA code of the platform `mgda.inria.fr`.

In a first optimization phase, the Pareto set and front associated with the sole primary cost functions were identified by using the classical NSGA-II algorithm. Three design-points were chosen in the Pareto set, two extreme (L and R), and one central (C). For each design-point, a Design of Experiment (DoE) was conducted externally to support the construction of local meta-models.

The second optimization phase was then initiated from either L, C or R. In the customized version of the Nash-MGDA code that relies on the pre-calculated data from the DoE, only one call to the simulation procedure, here FAST-OAD, was executed per computed Nash equilibrium point. Additionally, an improved treatment of the slack variables permitted to satisfy the functional constraints rigorously throughout the entire process.

In all three cases (L, C and R), the method was found effective to reduce substantially the secondary cost function f_3. In the central case C, the reduction could be of more than 60% while f_1 was diminished, an improvement, and f_2 was increased of less than 5%, a small degradation; both effects partially compensating each other resulted in an even smaller degradation of the Pareto-stationarity condition.

The focus in this report was on the demonstration of the potential of the prioritized optimization approach. Nevertheless, these preliminary numerical experiments permit to draw some sensible conclusions. In particular, it was observed that the mostly influential variables are different depending on the location on the Pareto front of the point chosen to initiate the calculation of the Nash equilibria:

- Case L (large OEW, small fuel mass): Horizontal Tail Aspect Ratio (to be reduced); Wing Taper ratio and Maximum Take-Off Thrust (to be increased);
- Case C (lighter sum at take-off OEW+mass fuel): Wing Taper Ratio (to be reduced); Maximum Take-Off Thrust (to be increased);
- Case R (small OEW, large fuel mass): Wing Aspect Ratio and Taper Ratio (to be increased); Horizontal Tail Aspect Ratio (to be reduced).

Concerning cost efficiency, the following estimated runtimes provide only approximate indications since the computations were not all performed on the same computer.

- **One call to FAST-OAD:**
  from 15 s to 45 s in general, depending on the point of evaluation which affects the inner iteration

- **1st Phase: Two-objective** (f_1, f_2) **Pareto front:**
  about 21.5 hrs by NSGA-II (2500 calls to FAST-OAD v. 1.3.4)
- **Design of Experiment about** $\mathbf{x}_A^\star$ **(495 points):**
  about 4 hrs (about 30 s per call), in Case C
- **2nd Phase: Continuum alone (22 equilibrium points):**
  about 7 mn runtime, in Case C
- **Three-objective** $(f1, f2, f3)$ **Pareto front:**
  about 77 hrs by NSGA-II.

Note that the present optimization test-case was indeed multi-objective, but not multi-disciplinary or multi-point. Thus a unique call to the performance evaluation procedure (FAST-OAD) was sufficient to compute all three cost functions. Nevertheless, the computation of the Pareto front associated with the three cost functions was substantially more costly than with two of them. In multi-disciplinary applications that we target in the long run, the penalty would be even more severe. That future context could thus reveal even more favorable to the mixed approach Pareto-front/Nash-games in contexts where the prioritization does make sense.

The computations relied on the gradient-free simulation procedure FAST-OAD, thus demonstrating a form of robustness of the prioritized approach. However applications in which the functional gradient is at hand could likely be handled even more efficiently by virtue of a more robust construction of the local metamodels.

Hence, in summary, when a prioritization of the cost functions in primary and secondary cost functions makes sense, it is computationally significantly more economical and effective to limit the computation of the Pareto front to the primary cost functions and to compute rapidly one or several continua of Nash equilibria that reduce the secondary cost functions, while best preserving the Pareto-optimality of the essential, primary cost functions. The Nash equilibria have been found close to the Pareto front associated with the whole set of cost functions, without prioritization, thus providing excellent designs.

In the future, we plan to realize a more complete and realistic experiment involving more than one secondary cost functions.

References

[1] J. Blank and K. Deb, *Pymoo: Multi-objective optimization in Python*, IEEE Access, 8 (2020), pp. 89497–89509.

[2] N. Brenowitz, *Calling Python from Fortran (not the other way around)*, Feb. 11 2022. https://www.noahbrenowitz.com/post/calling-fortran-from-python/.

[3] C. David, S. Delbecq, S. Defoort, P. Schmollgruber, E. Benard, and V. Pommier-Budinger, *From FAST to FAST-OAD: An open source framework for rapid overall aircraft design*, IOP Conference Series: Materials Science and Engineering, 1024 (2021), p. 012062.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 182–197.

[5] J.-A. Désidéri, *Quasi-Riemannian multiple gradient descent algorithm for constrained multiobjective differential optimization*, Research Report 9159, Inria, 21 March 2018. https://hal.inria.fr/hal-01740075v1.

[6] ——, *Platform for prioritized multi-objective optimization by metamodel-assisted Nash games*, Research Report 9290, Inria, September 2019. https://hal.inria.fr/hal-02285197.

[7] ——, *Adaptation by Nash games in gradient-based multi-objective/multi-disciplinary optimization*, in JANO13, Mathematical Control and Numerical Applications, Khouribga, Morocco, vol. 372, Springer Proceedings in Mathematics and Statistics, 2021. https://hal.inria.fr/hal-03430972.

[8] J.-A. Désidéri and R. Duvigneau, *Direct and adpative approaches to multi-objective optimization*, Research Report 9291, Inria, September 2019. https://hal.inria.fr/hal-02285899.

[9] ——, *Prioritized optimization by Nash games: towards an adaptive multi-objective strategy*, in ESAIM: Proceedings and Surveys, vol. 71, EDP Sciences, 2021, pp. 54–63. https://hal.inria.fr/hal-03430912.

[10] J.-A. Désidéri, R. Duvigneau, and A. Habbal, *Computational Intelligence in Aerospace Sciences, V. M. Becerra and M. Vassile Eds.*, vol. 244 of Progress in Astronautics and Aeronautics, T. C. Lieuwen Ed.-in-Chief, American Institute for Aeronautics and Astronautics Inc., Reston, Virginia, 2014, ch. Multi-Objective Design Optimization Using Nash Games.

[11] J.-A. Désidéri, P. Leite, and Q. Mercier, *Prioritized multi-objective optimization of a sandwich panel*, Research Report 9362, Inria, September 2020. https://hal.inria.fr/hal-02931770.

[12] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor, *OpenMDAO: An Open-Source Framework for Multidisciplinary Design, Analysis, and Optimization*, Structural and Multidisciplinary Optimization, 59 (2019), pp. 1075–1104.

[13] K. Miettinen, *Nonlinear Multiobjective Optimization*, Springer, 1998.

A Nested Pareto sets

To prove (3) in the context of differentiable, unimodal and convex cost functions $f_1,\ f_2,\ \ldots$.

Given the integer $k \geq 2$, let $(\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_k)$ be an element of the Pareto front associated with the first k cost functions. There exists (at least) one element $\overline{\mathbf{x}}$ of the Pareto set S_k such that:

$$f_1(\overline{\mathbf{x}}) = \bar{f}_1, \quad f_2(\overline{\mathbf{x}}) = \bar{f}_2, \quad \ldots, \quad f_k(\overline{\mathbf{x}}) = \bar{f}_k, \tag{21}$$

and

$$\sum_{j=1}^{k} \alpha_j \nabla f_j(\overline{\mathbf{x}}) = 0, \quad \alpha_j \geq 0 \ \forall j, \quad \sum_{j=1}^{k} \alpha_j = 1 \tag{22}$$

(Pareto stationarity). Consequently, the function

$$\phi(\mathbf{x}) = \sum_{j=1}^{k} \alpha_j f_j(\mathbf{x}) \tag{23}$$

is stationary at $\mathbf{x} = \overline{\mathbf{x}}$, and since it is convex, it achieves at $\overline{\mathbf{x}}$ a local minimum and this minimizing point is unique. We conclude that $\overline{\mathbf{x}}$ is unique.

Secondly, let $\mathbf{x}$ be an arbitrary element of S_k and let us make the additional hypothesis that $\mathbf{x} \notin S_{k+1}$. Then, there exists a design-point $\mathbf{y}$ that dominates $\mathbf{x}$ in efficiency w.r.t. $\{f_1, f_2, \ldots, f_{k+1}\}$, that is:

$$f_1(\mathbf{y}) \leq f_1(\mathbf{x}),\ f_2(\mathbf{y}) \leq f_2(\mathbf{x}),\ \ldots,\ f_k(\mathbf{y}) \leq f_k(\mathbf{x}),\ f_{k+1}(\mathbf{y}) \leq f_{k+1}(\mathbf{x}), \tag{24}$$

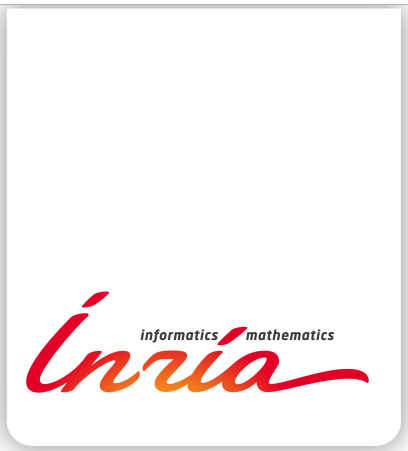
and at least one of these inequalities is strict. That inequality cannot be one of the above first k since otherwise $\mathbf{y}$ would also dominate $\mathbf{x}$ in efficiency w.r.t. $\{f_1, f_2, \ldots, f_k\}$ and this is excluded since $\mathbf{x} \in S_k$. Hence:

$$f_1(\mathbf{y}) = f_1(\mathbf{x}),\ f_2(\mathbf{y}) = f_2(\mathbf{x}),\ \ldots,\ f_k(\mathbf{y}) = f_k(\mathbf{x}), \tag{25}$$

and $f_{k+1}(\mathbf{y}) < f_{k+1}(\mathbf{x})$. But, we have shown that a point on the Pareto front associated with the functions $\{f_1, f_2, \ldots, f_k\}$ is the image of only one design-point in the Pareto set, and since $\mathbf{y} \neq \mathbf{x}$, we arrive at a contradiction. The contradiction is waived by rejecting the hypothesis that $\mathbf{x} \notin S_{k+1}$. Since $\mathbf{x}$ is arbitrary in S_k, this conclusion implies that $S_k \subseteq S_{k+1}$, and since k is arbitrary, the proof is complete.

□

Contents



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399