On the Consequences of the "No Free Lunch" Theorem for Optimization on the Choice of MDO Architecture

Charlie Vanaret and François Gallard

Institut de Recherche Technologique Saint Exupéry, Toulouse, France

Joaquim R. R. A. Martins

Department of Aerospace Engineering, University of Michigan Ann Arbor, Michigan, USA

Multidisciplinary Design Optimization (MDO) based on high-fidelity models is challenging due to the high computational cost of evaluating the objective and constraints. To choose the best MDO architecture, a trial-and-error approach is not possible due to the high cost of the overall optimization and the complexity of each implementation. We propose to address this issue by developing a generic methodology that applies to any (potentially expensive) design optimization problem. The methodology consists in generating a scalable analytic replacement function that can be quickly computed, yet captures the structure and behavior of the original high-fidelity problem. One crucial characteristic of this replacement model is that the number of inputs and outputs can be set independently, making the problem scalable. This facilitates the evaluation of MDO architectures for the original MDO problem. The methodology is applied to two academic MDO test cases: a supersonic business jet design problem and a propane combustion problem. Well-known architectures (Multidisciplinary Feasible (MDF) and Individual Disciplinary Feasible (IDF)) are benchmarked on various instances to demonstrate the dependency between the performance of the architecture and the dimensions of the problem. The results show that the "no free lunch" theorem of optimization, which states that no particular optimization algorithm holds an advantage over another when considering all possible problems, also applies to the choice of the best suited MDO architecture for an MDO problem with scalable dimensions.

Nomenclature

N	Number of disciplines
x_i	Local design variables of discipline i
x_0	Shared (global) design variables
$y = (y_1, \ldots, y_N)$	Coupling variables
$y^t = (y_1^t, \ldots, y_N^t)$	IDF target coupling variables
$y_{j \neq i}$	Coupling variables computed by all disciplines but the ith
$y_i(x_i, x_0, y_{j \neq i})$	Disciplinary analysis of discipline i
f	Objective function
g	Physical inequality constraints
$[\underline{x}, \overline{x}]$	Bounds on a variable x
ϕ_i	i-th component of a vector-valued function ϕ
$\frac{\partial \phi}{\partial x}$	Partial derivative of a function ϕ with respect to a variable x
$\frac{d\phi}{dx}$	Total derivative of a function ϕ with respect to a variable x
n_x	Common dimension for all inputs of a scalable problem
n_y	Common dimension for all outputs of a scalable problem
N_x	Total input dimension of a scalable problem
N_y	Total output dimension of a scalable problem

1. Introduction

1.1. Motivation

MDO is the application of numerical optimization to the design of systems dependent on multiple disciplines. The multiple disciplines arise in a system that is intrinsically governed by multi-physics models, consists of multiple components, or both. MDO strives to account for the multiple disciplines at two distinct levels: the analysis level and the design level. At the analysis level, we need to ensure that all the interdisciplinary couplings are considered in a system analysis. At the design level, we need to make sure that the optimization problem is formulated such that there is enough information to perform optimal multidisciplinary design tradeoffs. Coupling variables capture the interactions between the various disciplines ; a given discipline has a subset of these coupling variables as inputs, and another subset as outputs. The sole coupled analysis however is not sufficient to achieve tradeoffs. First, the objective function must reflect the performance that we want to maximize, and the appropriate contributions to the objective from each discipline are correctly accounted for. Second, the design variables must include the variables that drive the design from each discipline. Design variables that directly affect at least two disciplines are called global variables, while design variables that directly affect only one discipline are local to that discipline. Finally, various constraints are required to obtain meaningful designs. Constraints can also be local or global, depending on whether they depend directly on variables from one or more disciplines, respectively.

The general Simultaneous Analysis and Design (SAND) optimization problem[1] is:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{subject to} \quad & g(x,y) \leq 0 \\ & R_i(x_i, x_0, y) = 0, \quad \forall i \in \{1, \ldots, N\} \end{aligned} \tag{1}$$

where N is the number of disciplines, f is the objective function, g are design constraints, x_i are the local design variables of discipline i, x_0 are global (shared) design variables, x denotes $(x_0, x_1, \ldots, x_N)$, y are coupling variables, $R_i(x_i, x_0, y) := y_i(x_i, x_0, y_{j\neq i}) - y_i$ denotes the i-th disciplinary residual and $y_i(x_i, x_0, y_{j\neq i})$ are disciplinary analyses in explicit form. For the sake of simplicity, equality constraints are rewritten as two inequalities constraints, and state variables (outputs of the disciplines) have not been written explicitly, since they can be considered as extensions of the coupling variables. In addition, a discipline that is formulated using residuals ($R_i(x_i, x_0, y) = 0$) can be equivalently expressed in explicit form ($y = y(x_i, x_0)$), thanks to the implicit function theorem. In the following, we formulate the disciplines in explicit form; however, the present methodology could also be applied to disciplines in residual form.

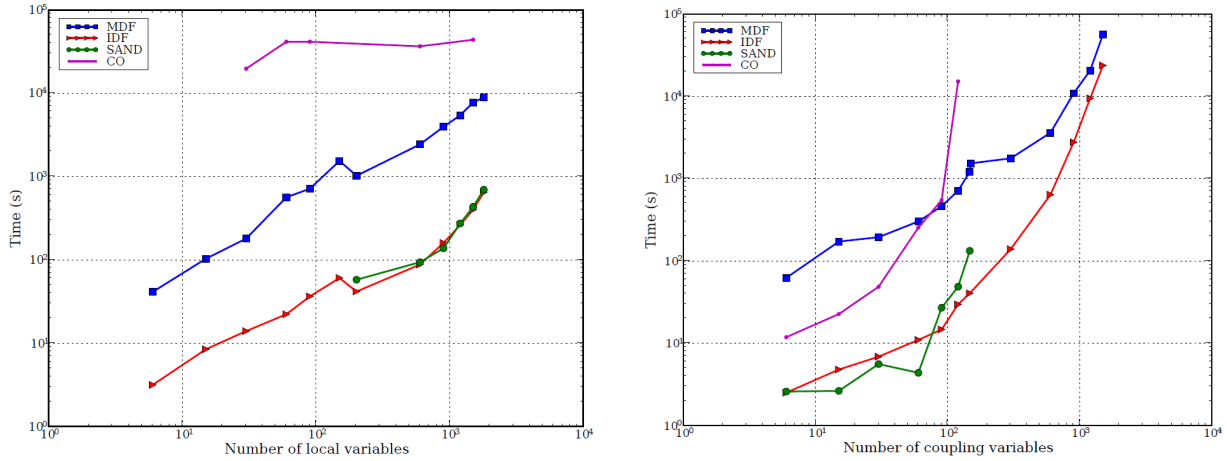
MDO architectures reformulate the original SAND problem by generating one or multiple MDO optimization problems, whose definitions may include multiple coupled simulations and additional constraints. MDO architectures aim at lowering the overall computational cost of the optimization, enabling the use of dedicated algorithms to the discipline subproblems, or both. Since all MDO architectures are derived from the original SAND problem, they should provide identical solutions.

For instance, architectures based on a Multidisciplinary Analysis (MDA), such as MDF[2] or Bilevel Integrated System Synthesis (BLISS),[3] solve the coupled simulation at each optimization iteration, while others, such as IDF,[2] decouple the analyses and thus only guarantee the consistency between the simulations at the convergence of the optimization.

When using high-fidelity models, the most efficient architecture is the one that requires the fewest disciplinary evaluations, since the high-fidelity model evaluations dominate the computational cost. The most efficient architecture for a particular MDO problem is not known a priori. Furthermore, an architecture that proves efficient for a particular problem may exhibit mediocre performance for a problem with different properties. This phenomenon was described in the context of optimization algorithms by Wolpert and Macready.[4] They proved that on all possible optimization problems, all optimization algorithms perform the same on average. The choice of an appropriate optimization algorithm is thus instance-dependent. This is a well known issue in applied optimization: the choice of the best algorithm depends on the application, making the development of optimization algorithm an active subject of research.

Since different MDO architectures generate different optimization problems for a given SAND problem, one can expect that the same optimization algorithm will have a different performance on these problems. Therefore, the performances of MDO architectures cannot be measured independently from the performances of optimization algorithms. This makes MDO architectures subject to the "no free lunch theorem".

An analytical scalable problem was introduced by Martins and Tedford[5,6] to highlight the influence of the problem dimensionality on the performance of architectures. This analytical problem has a quadratic objective function and linear inequality constraints, and the governing equations consist of a linear system. The user may tune the number of disciplines, the number of output coupling variables associated with each discipline, the number of local design variables associated with each discipline, the number of global design variables, and the strength of coupling between the disciplines. Figure 1 compares the performance of MDO architectures on various dimensions of the local design variables and the coupling variables. This scalable problem provides an insight into the performance of MDO architectures with respect to the number of design variables or coupling variables. However, the study is limited to a simple analytical problem with a fixed structure, and is not inspired from a physical problem.



(a) Effect of the number of local design variables

(b) Effect of the number of coupling variables

Figure 1: Analytic scalable problem[5]

In the context of high-fidelity MDO based on computational fluid dynamics (CFD) and computational structural mechanics (CSM), the disciplinary simulations typically run for a duration of several minutes to hours. An MDO problem may require hundreds of simulations, which could ultimately add up to several weeks for an optimization. A trial-and-error approach is therefore not suitable to find the best MDO architecture, or optimization algorithm, for a high-fidelity MDO problem.

Furthermore, the numerical simulations in CFD or CSM typically involve meshes with variable discretizations, depending on the desired refinement of the models. These models, when coupled, have various dimensions for the coupling variables (typically, the loads and displacement vectors in aeroelastic problems are computed from the simulation meshes). The geometry description may also vary from a study to another, impacting the number of design variables, as well as the constraints (for instance, stress limits depend on the number of structural elements in CSM models). This paper illustrates how the numbers of design and coupling variables impact the resolution of optimization problems generated by MDO.

The appropriate architecture is crucial for a particular high-fidelity based design optimization. We believe that generating a scalable analytic *replacement* of the physical problem that is easily computed is a step toward a better grasp of MDO architectures. Thus, we propose a methodology that transforms an expensive physical model into a model based on analytical replacements that is cheap to evaluate. The original problem is then replaced by the cheap problem, in order to benchmark MDO architectures.

The replacement function is not to be confused with a *surrogate* model, since it is not meant to provide an actual approximation of the original model. Instead, the replacement function is just meant to mimic the trends of the original problem. The replacement problem should therefore preserve the mathematical structure of the original one. Additionally, the replacement model is scalable, that is, the dimensions of the design variables and the coupling variables may be varied independently, which allows performance comparisons of the architectures with respect to the dimensions. The generated problems illustrate the variations in performance of the architectures and algorithms, showing that there is "no free lunch" for MDO architectures.

The proposed methodology, presented in Section 2, first builds an interpolated model for each discipline of the original problem with a limited number of evaluations, then extrapolates the interpolated model to an arbitrary dimension. Our methodology preserves the interface of the original problem, that is, the names of the inputs (design variables) and the outputs (coupling and state variables). Any high-fidelity discipline of the original problem may therefore be replaced by a cheap scalable component generated by our methodology. The interpolated models are not meant to be surrogate models, in that they do no accurately approximate the values of the original functions, they merely emulate the same mathematical structure and properties.

In Section 3, we demonstrate mathematical properties guaranteed by the methodology: the existence of solutions to the coupling problem and to the MDO problem, and the existence of bounds on the coupling variables. This last property is particularly important for IDF. Finally, in Section 4 we apply the methodology to two well-known benchmark problems in the MDO literature: the Supersonic Business Jet (SSBJ) problem of Sobieski et al.,[3] and the propane combustion problem.[6,7]

1.2. MDO architectures

MDO architectures reformulate the original SAND problem into one or multiple optimization problems, and possibly coupling problems.

1.2.1. MDF

MDF is an architecture that guarantees an equilibrium between all disciplines at each iterate x of the optimization process. Consequently, should the optimization process be interrupted, the best known solution has a physical meaning. MDF generates the smallest possible optimization problem, in which the coupling variables and the state variables are removed from the set of optimization variables. The residuals, which are associated with the state variables, are removed from the set of constraints. MDF can be stated as:

$$\begin{aligned} \min_{x} \quad & f(x, y(x)) \\ \text{subject to} \quad & g(x, y(x)) \leq 0 \end{aligned} \tag{2}$$

The coupling variables $y(x)$ are computed at equilibrium via an MDA. This amounts to solving a system of (possibly nonlinear) equations using fixed-point methods (Gauss–Seidel, Jacobi) or root-finding methods (Newton–Raphson, quasi-Newton). A prerequisite for invoking MDF is the existence of an equilibrium for any values of the design variables x encountered during the optimization process. The MDF workflow is shown in Figure 2.

Gradient-based optimization algorithms require the computation of the total (coupled) derivatives $\frac{d\phi}{dx}$ of $\phi(x, y(x))$, where $\phi \in \{f, g\}$:

$$\frac{d\phi}{dx} = \frac{\partial \phi}{\partial x} + \frac{\partial \phi}{\partial y}\frac{dy}{dx} \tag{3}$$

Linearizing the MDA at equilibrium, we obtain

$$\frac{dR}{dx} = 0 = \frac{\partial R}{\partial x} + \frac{\partial R}{\partial y}\frac{dy}{dx}. \tag{4}$$

Inserting the solution for dy/dx into Equation (3) leads to the expression of the total derivatives:

$$\frac{d\phi}{dx} = \frac{\partial \phi}{\partial x} - \underbrace{\frac{\partial \phi}{\partial y}\overbrace{\left[\frac{\partial R}{\partial y}\right]^{-1}}^{-\frac{dy}{dx}}}_{-\psi_\phi^T}\frac{\partial R}{\partial x}. \tag{5}$$

The product of the three matrices in this equation can be obtained in two ways. Using the *direct method*, the term dy/dx is obtained by solving the linear system:

$$\frac{\partial R}{\partial y}\frac{dy}{dx} = -\frac{\partial R}{\partial x}, \tag{6}$$

which is independent of the function ϕ and must be solved for every design variable in the vector x. The total derivative is then given by Equation (3).

The *adjoint method* consists in solving Equation (5) for the adjoint vector ψ_ϕ^T, obtained by solving the linear system:

$$\frac{\partial R}{\partial y}^T \psi_\phi = -\frac{\partial \phi}{\partial y}^T, \tag{7}$$

which is independent of the design variables x and must be solved for each function ϕ. The total derivative is then given by:

$$\frac{d\phi}{dx} = \frac{\partial \phi}{\partial x} + \psi_\phi^T \frac{\partial R}{\partial x}. \tag{8}$$

Given the equations above, it is more efficient to compute coupled derivatives using the adjoint method when the number of variables is larger than the number of functions. Conversely, the direct method is more efficient when the number of functions is larger than the number of variables.

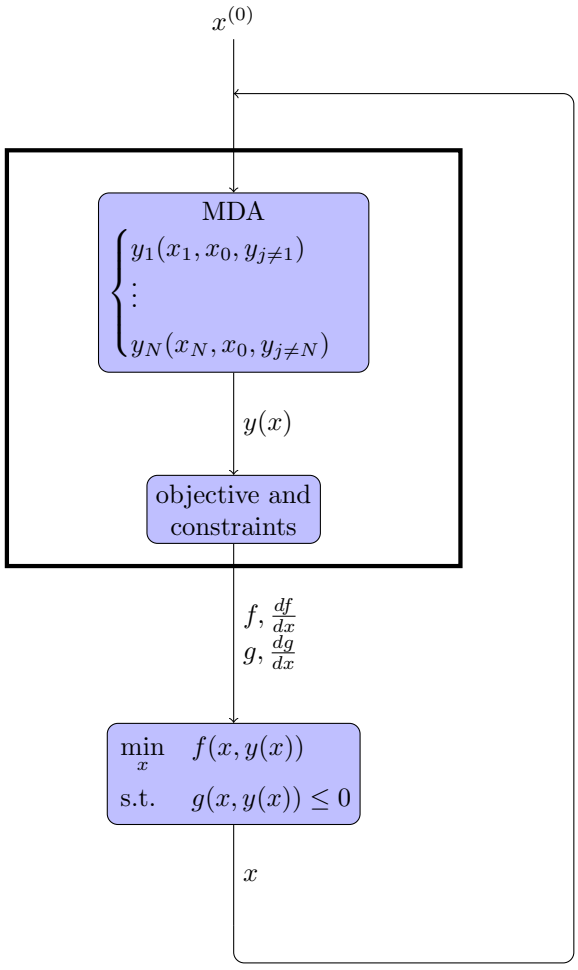


Figure 2: MDF

1.2.2. IDF

IDF handles the disciplines in a decoupled fashion: all disciplinary analyses are performed independently and possibly in parallel. The coupling variables y^t (called targets) are driven by the optimization algorithm and are inputs for all disciplinary analyses $y_i(x_i, x_0, y^t_{j\neq i}), \forall i \in \{1, \ldots, N\}$. By comparison, MDF handles the disciplines in a coupled manner: the inputs of the disciplines are outputs of the other disciplines. IDF can be stated as:

$$
\begin{aligned}
\min_{x,y^t} \quad & f(x,y^t) \\
\text{subject to} \quad & g(x,y^t) \leq 0 \\
& y_i(x_i,x_0,y^t_{j\neq i}) - y^t_i = 0, \quad \forall i \in \{1,\ldots,N\}
\end{aligned}
\tag{9}
$$

Additional consistency constraints $y_i(x_i,x_0,y^t_{j\neq i}) - y^t_i = 0, \forall i \in \{1,\ldots,N\}$ ensure that the coupling variables computed by the disciplinary analyses coincide with the corresponding inputs y^t of the other disciplines. This guarantees an equilibrium between all disciplines at convergence. The IDF workflow is shown in Figure 3.

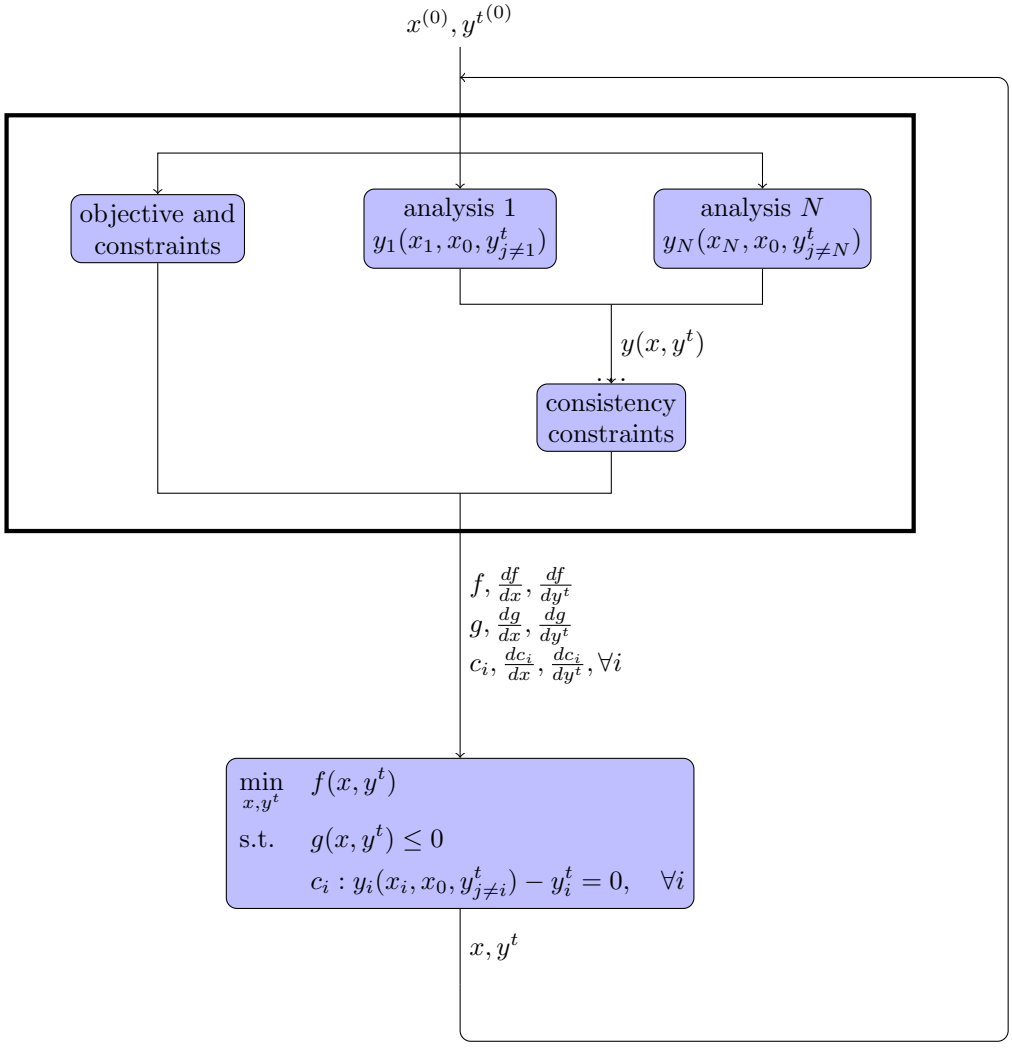


Figure 3: IDF

2. Scalable analytic replacement function

The proposed approach consists in automatically developing a cheap scalable analytic replacement of an expensive physical model. The function is meant to capture the trends of the original physical model (monotonicity, convexity) and preserves the interface as well (number and names of inputs and outputs), as shown in Figure 4. It is not meant to approximate the actual values of the original model.

Table 1 details the sequence of operations that transform the original model (a coupling or a constraint) $\phi : \mathbb{R}^n \to \mathbb{R}^m$ into the scalable analytic function. The second and third columns represent the functional notation of the function before and after the transformation, respectively. The evaluation cost of the resulting

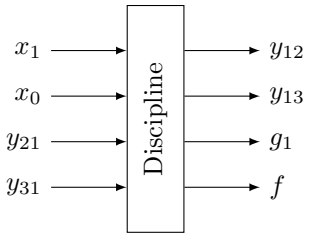


Figure 4: Example of disciplinary interface

function is listed in the fourth column. In the following sections, we detail each of these operations. n_x denotes the size of each design vector (such as x_0 in Figure 4), n_y denotes the size of each coupling vector (such as y_{12}) and N_x represents the total size of the design space.

Table 1: Methodology steps

Transformation	From	To	Resulting function
1D restriction	$\phi : \mathbb{R}^n \to \mathbb{R}^m$	$\phi^{(1d)} : [0,1] \to \mathbb{R}^m$	expensive
Interpolation	$\phi^{(1d)} : [0,1] \to \mathbb{R}^m$	$\phi^{(\text{int})} : [0,1] \to [0,1]^m$	cheap
Extrapolation	$\phi^{(\text{int})} : [0,1] \to [0,1]^m$	$\phi^{(\text{ext})} : [0,1]^{N_x} \to [0,1]^{n_y}$	cheap

2.1. One-dimensional restriction

The original model ϕ is restricted to a one-dimensional function $\phi^{(1d)} : [0,1] \to \mathbb{R}^m$ by evaluating it along a diagonal line in the domain $[\underline{x_1}, \overline{x_1}] \times \ldots \times [\underline{x_n}, \overline{x_n}]$:

$$\phi^{(1d)}(t) = \phi(\underline{x_1} + t(\overline{x_1} - \underline{x_1}), \ldots, \underline{x_n} + t(\overline{x_n} - \underline{x_n})) \tag{10}$$

As an example, consider the original physical model $\phi(x_1, x_2) = \sin(\sqrt{x_1^2 + x_2^2})$ over the domain $[-5, 5]^2$, show in Figure 5. The one-dimensional restriction of this function along the domain's diagonal is shown in Figure 6.

2.2. Interpolation

Let T be a set of samples in $[0,1]$ with cardinality $|T|$, on which $\phi^{(1d)} : [0,1] \to \mathbb{R}^m$ is evaluated. For any component $i \in \{1, \ldots, m\}$ of $\phi^{(1d)}$, we define the direct image:

$$\phi_i^{(1d)}(T) := \{\phi_i^{(1d)}(t) \mid t \in T\}. \tag{11}$$

The minimal and maximal elements of this function are given by:

$$\begin{aligned} m_i &:= \min \phi_i^{(1d)}(T) \\ M_i &:= \max \phi_i^{(1d)}(T), \end{aligned} \tag{12}$$

The set of scaled images is defined as:

$$\phi_i^{(s1d)}(T) := \left\{ \frac{\phi_i^{(1d)}(t) - m_i}{M_i - m_i} \mid t. \in T \right\} \tag{13}$$

Each component i of the function $\phi^{(1d)}$ is then approximated by a scaled polynomial interpolation $\phi_i^{(\text{int})}$ (e.g., a third-order spline) over the data $\left(T, \phi_i^{(s1d)}(T)\right)$. Note that the derivative of a one-dimensional interpolation is generally provided by interpolation libraries. Figure 7 shows a scaled third-order polynomial interpolation $\phi^{(\text{int})}$ of $\phi^{(1d)}$ for our example function.

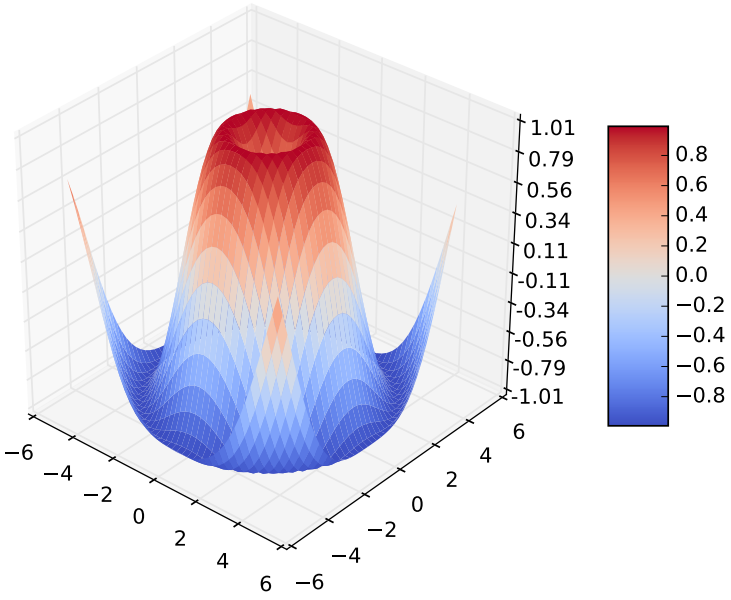


Figure 5: Surface of the function $\phi(x_1, x_2) = \sin(\sqrt{x_1^2 + x_2^2})$

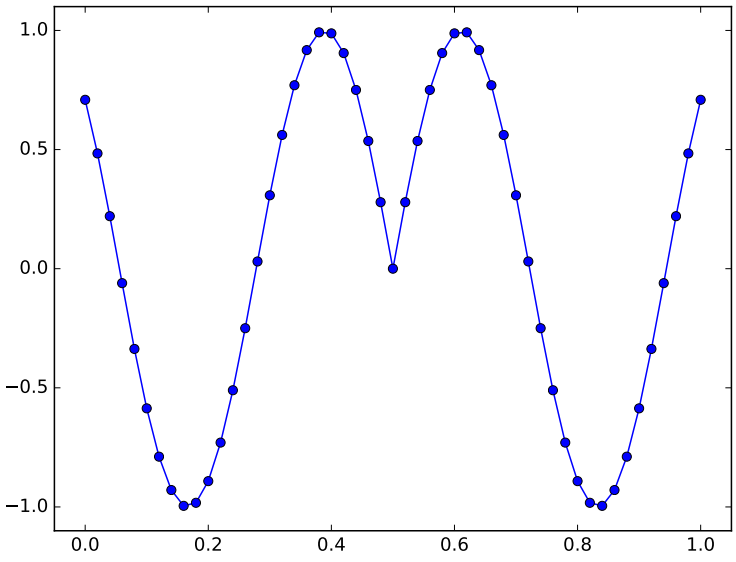


Figure 6: One-dimensional restriction $\phi^{(1d)}$ of $\phi(x_1, x_2) = \sin(\sqrt{x_1^2 + x_2^2})$

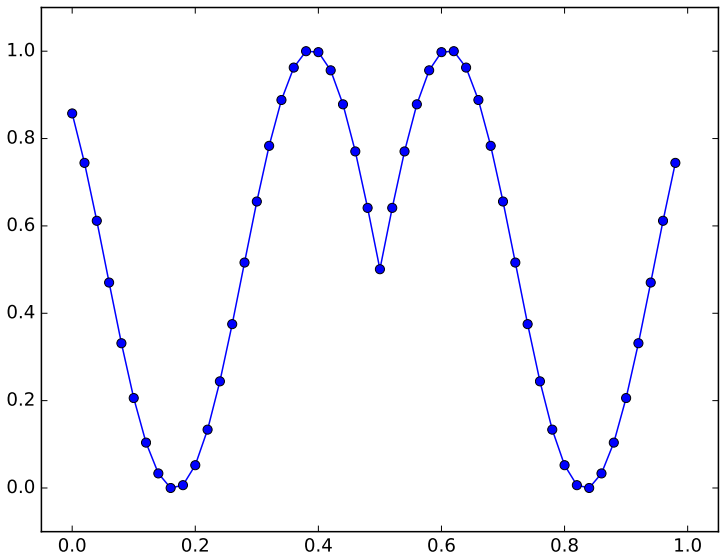


Figure 7: Scaled one-dimensional interpolation $\phi^{(\text{int})}$ of $\phi(x_1, x_2) = \sin(\sqrt{x_1^2 + x_2^2})$

2.3. Extrapolation

Based on a vector of one-dimensional interpolations $\phi_i^{(\text{int})}, i \in \{1, \ldots, m\}$ that are fast to compute, the extrapolation step generates a function $\phi^{(\text{ext})}$ whose input and output sizes vary independently. The way inputs and outputs of arbitrary sizes are handled is determined by two data structures: one that establishes the dependency of the outputs of the scalable function $\phi^{(\text{ext})}$ with respect to its inputs (described in Section 2.3.1), and the other that defines the correspondence between the interpolations $\phi_i^{(\text{int})}$ and the components of the scaled function $\phi^{(\text{ext})}$ (described in Section 2.3.2).

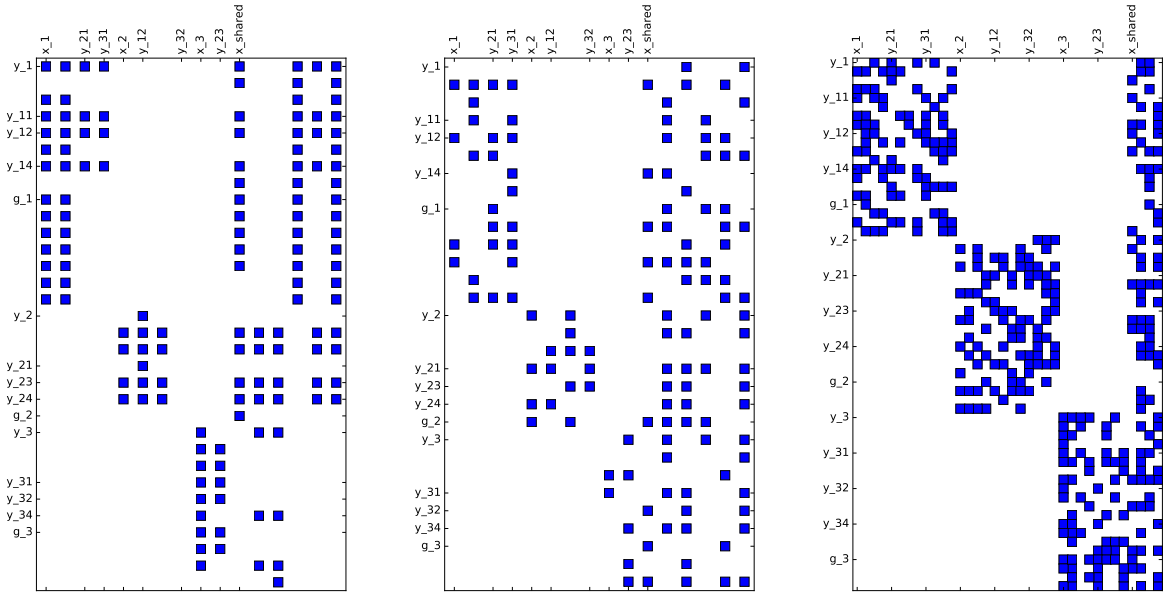
2.3.1. Input-output dependency

Within a discipline, the dependency of scalable functions (with n_y components each) with respect to their inputs (each of size n_x) may be fixed by constructing a sparse dependency matrix S. Each block row of S represents a function of the problem (constraint or coupling), and each block column represents an input (design variable or coupling). A nonzero element represents the dependency of a particular component of a function with respect to a particular component of an input.

The matrix S is randomly generated with a given density factor d, which determines the degree of dependency of outputs with respect to inputs for the original model. Figure 8a illustrates the dependency matrix of the SSBJ problem (see Section 4.4). The variables x_{shared} (block column on the right) are global design variables, shared by all disciplines. The three remaining blocks in the matrix correspond to the three disciplines (aerodynamics, structures, and propulsion). Figure 8b shows the dependency matrix, generated at random, of the scalable SSBJ problem with the same dimensions as the original problem, and density factor $d = 0.4$. Figure 8c shows the dependency matrix, generated at random, of the scalable SSBJ problem with dimensions $n_x = 4$ and $n_y = 4$, and density factor $d = 0.4$.

In order to generate a family of consistent problems, we first generate a large random dependency matrix S, which is then restricted to the desired dimensions (Figure 9). This approach guarantees that the dependency of a given output with respect to its inputs does not change when their sizes change. Figure 9a and Figure 9b represent restrictions of the large matrix in Figure 9c (for which the inputs and outputs are scaled independently), for $(n_x = 2, n_y = 2)$ and $(n_x = 3, n_y = 3)$, respectively. Note that the constraints g_i have the same size $n_x = 10$ as the x_i design variables.

Figure 10 illustrates various density factors of the dependency matrix. The density increases with this coefficient, from 0.2 in Figure 10a up to a full dependency matrix in Figure 10c. It is not desirable to use

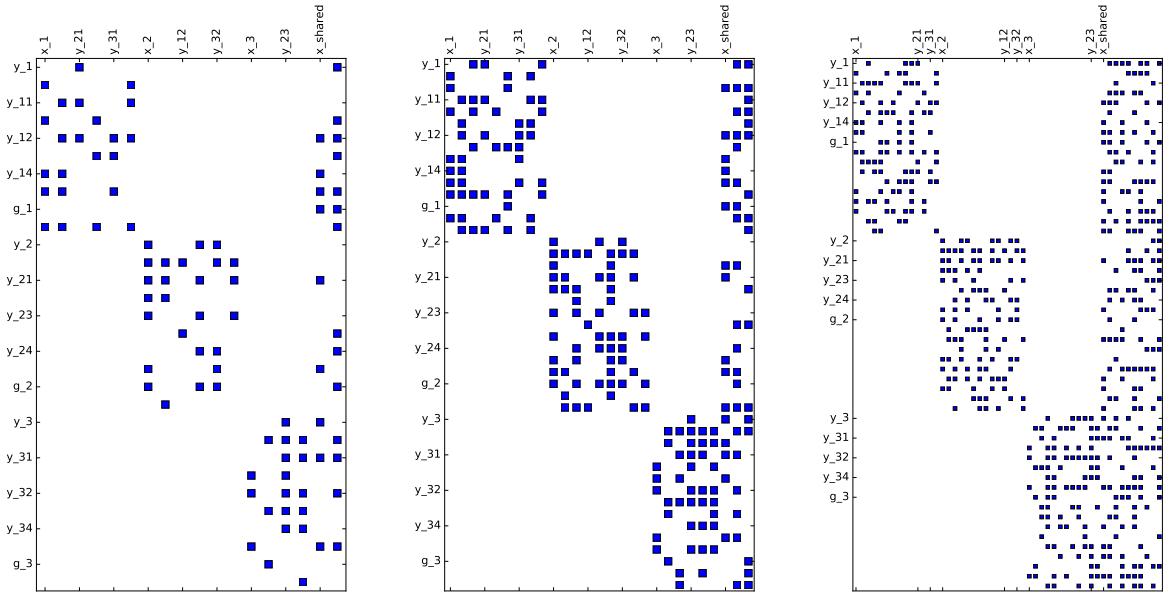


(a) Original model

(b) Scalable analytic replacement function (original sizes) with $d = 0.4$

(c) $(n_x, n_y, d) = (4, 4, 0.4)$

Figure 8: Dependency matrix for the SSBJ problem



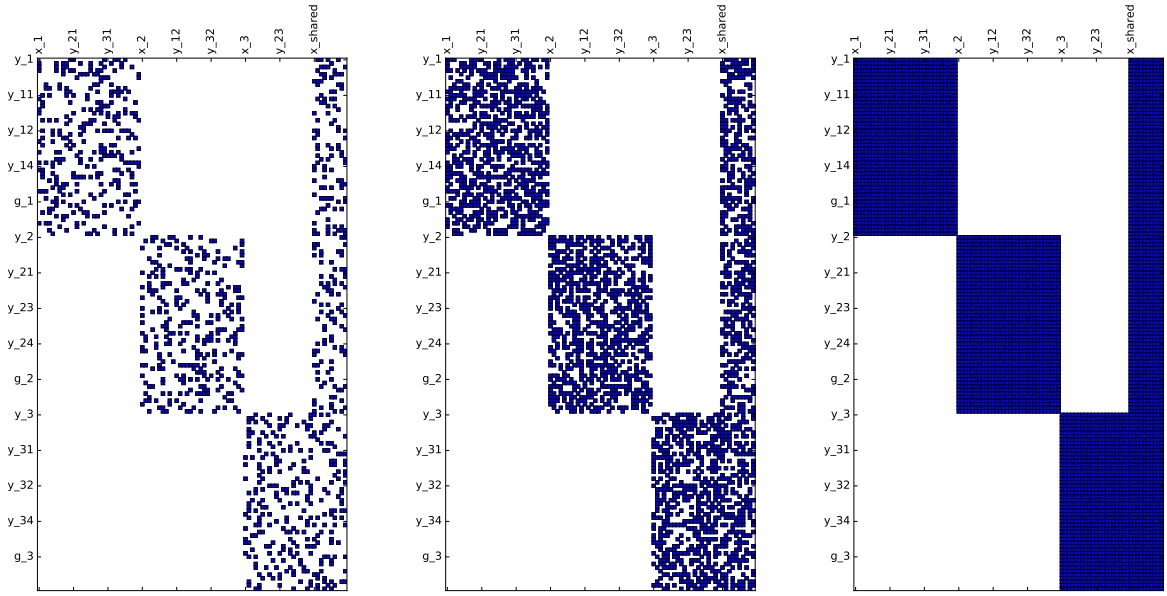
(a) $(n_x, n_y, d) = (2, 2, 0.4)$

(b) $(n_x, n_y, d) = (3, 3, 0.4)$

(c) $(n_x, n_y, d) = (10, 2, 0.4)$

Figure 9: Effect of the dimensions on the dependency matrix of the SSBJ problem

a density factor of 1.0, since this does not generate independent combinations for the components of the outputs ; as a result, many components would become equal.



(a) $(n_x, n_y, d) = (10, 10, 0.2)$ (b) $(n_x, n_y, d) = (10, 10, 0.5)$ (c) $(n_x, n_y, d) = (10, 10, 1.0)$

Figure 10: Effect of the density factor on the dependency matrix of the SSBJ problem

2.3.2. Component dependency

Each component of the extrapolated function $\phi^{(\text{ext})}$ depends on a unique component of the one-dimensional interpolation $\phi^{(\text{int})}$. The correspondence between the m original components and the n_y extrapolated components may be established by picking for each extrapolated component i a random number k_i in $\{1, \ldots, m\}$ that represents the index of the selected original component.

Figure 11 illustrates a random mapping between $m = 3$ components of the interpolation and $n_y = 8$ components of the extrapolated function. The first component (in purple) is selected four times (components 2, 5, 6 and 8), the second component (in orange) is selected twice (components 1 and 7) and the third component (in green) is selected twice (components 3 and 4).

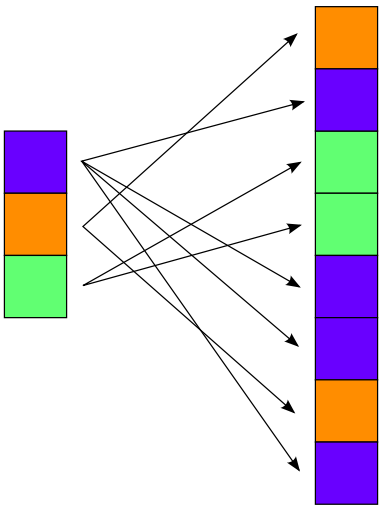


Figure 11: Component dependency between the interpolation and the extrapolation

2.3.3. *Extrapolated function*

The function $\phi^{(\text{ext})} : [0,1]^{N_x} \to [0,1]^{n_y}$ extrapolates $\phi^{(\text{int})}$ to n_y dimensions. The i-th component ($i \in \{1,\dots,n_y\}$) of $\phi^{(\text{ext})}$ is given by:

$$\phi_i^{(\text{ext})}(x) = \frac{1}{|S_{i\cdot}|} \sum_{j \in S_{i\cdot}} \phi_{k_i}^{(\text{int})}(x_j). \tag{14}$$

where $S_{i\cdot}$ represents the nonzero elements of the i-th row of the dependency matrix S, and $|S_{i\cdot}|$ their number. The function $\phi_{k_i}^{(\text{int})}$ is detailed in Section 2.3.2 and its arguments are explained in Section 2.3.1.

2.4. Inequality constraints

By construction, the scaled constraints take their values in $[0,1]$, which is incompatible with the original constraints $g_i \leq 0$. We propose to translate a scaled constraint $\tilde{g}_i$ by a threshold $\tau_i \in [0,1]$ as follows

$$\tilde{g}_i - \tau_i \leq 0. \tag{15}$$

The thresholds $(\tau_i)_i$ are computed such that the initial point is always feasible. It activates a given percentage $p \in [0,1]$ of the constraints, while the rest of the constraints are initially satisfied but inactive.

A threshold τ_i is given by:

$$\tau_i = \begin{cases} \tilde{g}_i^{(0)} & \text{if} \quad \mathcal{U}(0,1) \leq p \quad \text{(active)} \\ \alpha_i + (1-\alpha_i)\tilde{g}_i^{(0)} & \text{otherwise} \qquad \text{(satisfied)} \end{cases} \tag{16}$$

where $\mathcal{U}(0,1)$ is a random number picked in $[0,1]$ with uniform probability, $\tilde{g}_i^{(0)}$ is the constraint evaluated at the initial point, and $\alpha_i \in [0,1]$ determines to what extent inactive constraints are satisfied. Figure 12 illustrates the effect of the threshold τ_i on the feasibility of the constraint $\tilde{g}_i - \tau_i \leq 0$ at the initial point.

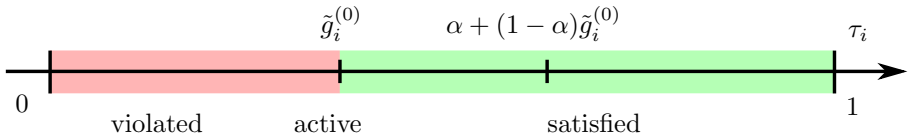


Figure 12: Effect of τ_i on the feasibility of the constraint $\tilde{g}_i - \tau_i \leq 0$ at the initial point

3. Theoretical properties

As previously mentioned, the proposed methodology aims at creating an analytical replacement function that preserves the functional characteristics of the physical model, and thus captures the properties that drive the optimization performance. For instance, if the objective function of the problem is linear or convex, the objective function of the scalable problem should also be linear or convex.

3.1. Assumptions on the interpolation

The interpolation $\phi_i^{(\text{int})}$ should be continuous for any value of the generated samples. This is the case of many standard one-dimensional interpolation methods, such as cubic splines.

3.2. Existence of a solution to the coupling problem

We show that the explicit definition of the coupling variables in the methodology guarantees that a solution to the coupling problem between all disciplines always exists, for any value of the design variables. This result is based on Brouwer's fixed point theorem.[8,9]

Theorem 1 (Brouwer's Fixed Point (1912))**.** *Every continuous function from a convex compact subset K of an Euclidean space to K itself has a fixed point.*

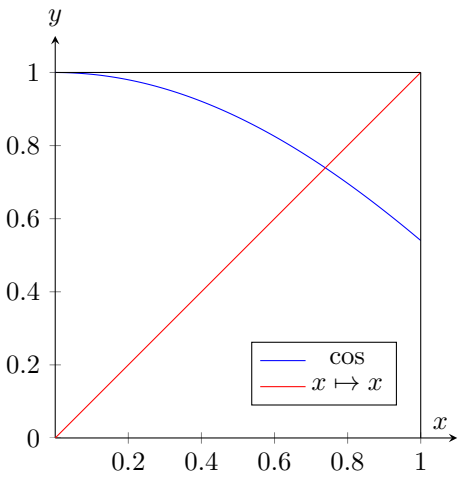


Figure 13: One-dimensional example illustrating Brouwer's fixed point theorem

Figure 13 illustrates Theorem 1 in one dimension. The function $x \mapsto \cos(x)$ is continuous and maps $[0,1]$ to itself. Therefore, cos has a fixed point $\tilde{x}$ such that $\cos(\tilde{x}) = \tilde{x}$. Graphically, this implies that the curves of cos and $x \mapsto x$ necessarily intersect.

Theorem 2 states the existence of a solution to the coupling problem for any value of the design variables, which is a prerequisite for benchmarking MDF. However, the theorem does not guarantee that a unique solution exists, or that any numerical method (fixed-point method or root-finding algorithm) will converge.

Theorem 2 (Existence of a solution to the coupling problem)**.** *An equilibrium between all disciplines exists for any value of the design variables x, that is there exists couplings at equilibrium $y^{(eq)}$ such that:*

$$y_i(x_i, x_0, y_{j\neq i}^{(eq)}) = y_i^{(eq)}, \quad \forall i \in \{1, \dots, N\} \tag{17}$$

Proof. Let us define the function μ that takes coupling variables as arguments and computes the disciplinary analyses in explicit form for fixed values of the design variables x:

$$\begin{array}{rcccl} \mu & : & [0,1]^N & \to & [0,1]^N \\ & & y & \mapsto & \mu(y) = \begin{pmatrix} y_1(x_1, x_0, y_{j\neq 1}) \\ \vdots \\ y_N(x_N, x_0, y_{j\neq N}) \end{pmatrix} \end{array} \tag{18}$$

By construction of the one-dimensional restriction whose inputs and outputs are scaled in $[0,1]$ (see Table 1), μ is continuous and maps $[0,1]^N$ into itself. From Theorem 1, μ has a fixed point, and therefore there exists a solution to the coupling problem for any value of x. □

3.3. Preservation of the ratio of original components

We now show that the ratio of components of the interpolation $\phi^{(\text{int})}$ is preserved in the extrapolation $\phi^{(\text{ext})}$.

Theorem 3 (Preservation of ratio)**.** *When n_y approaches $+\infty$, the ratio of components of the original functions is preserved.*

Proof. The n_y components of the extrapolation $\phi^{(\text{ext})}$ are randomly drawn with uniform probability from the m original components of the interpolation $\phi^{(\text{int})}$. $\phi_i^{(\text{ext})}$ is linear with respect to the interpolation function $\phi_{k_i}^{(\text{int})}$. As n_y approaches $+\infty$, all draws become equally possible with probability $\frac{1}{m}$, which preserves the proportions. □

3.4. Existence of a feasible solution to the scalable problem

Theorem 4 (Existence of a minimum)**.** *There exists a feasible solution to the scalable problem, for any dimension of inputs and outputs.*

Proof. By construction:

- The initial point is feasible with respect to the bound constraints and the nonlinear constraints. A translation is applied to the nonlinear inequality constraints, so that their value on the initial point is nonpositive (see Section 2.4). The feasible set is therefore nonempty ;
- By construction of the one-dimensional restriction and because the interpolations are continuous, the objective function is continuous and bounded from below on a closed set.

□

3.5. Existence of derivatives

Since the interpolation method generates continuously differentiable ($\mathcal{C}^1$) one-dimensional interpolations, which is the case of standard methods such as splines, the scalable replacement functions are also $\mathcal{C}^1$. This is required for gradient-based optimization.

Theorem 5 (Existence of derivatives)**.** *The scalable extrapolations are continuously differentiable with respect to their inputs.*

Proof. The extrapolation $\phi_i^{(\text{ext})}$ is linear with respect to the interpolation functions, which are assumed to be $\mathcal{C}^1$ (see Section 2.3.3). Therefore $\phi_i^{(\text{ext})}$ is also $\mathcal{C}^1$. □

3.6. Existence of bounds on the target coupling variables

All inputs and outputs belong to $[0, 1]$, which ensures that all optimization variables are bounded, in particular coupling variables in IDF.

4. Numerical results

We apply the methodology presented in Section 2 to two benchmark problems: the Supersonic Business Jet problem proposed by Sobieski et al.,[3] and a propane combustion problem.[7] The aim is to produce a benchmark plot of architectures performances, similar to Figure 1.[5] For our numerical experiments, we use the best practices for comparing methods on random problems recommended by Johnson.[10]

4.1. Comparison criterion

The computational time alone is not relevant to compare the variety of architectures on a particular instance of the scalable problem: since the original (expensive) disciplines have been replaced by analytic functions that are quickly evaluated, the total time gives no indication on the execution time of the initial problem.

We define a criterion equivalent to an effort in terms of disciplinary evaluations. This criterion aggregates the following factors:

1. n_{eval_i}: number of disciplinary analyses performed (possibly in parallel) in IDF over all disciplines i ;
2. n_{lin_i}: number of linearizations (partial derivatives) over all disciplines i ;
3. n_{LU}: number of LU factorizations during the computations of the total derivatives in MDF ;
4. n_{MDA}: number of MDAs in MDF.

The comparison criterion is given by:

$$C = \underbrace{\sum_{i \in \text{disciplines}} n_{\text{eval}_i}}_{\text{analyses}} + \overbrace{c_{\text{lin}} \sum_{i \in \text{disciplines}} n_{\text{lin}_i}}^{\text{linearizations}} + \underbrace{c_{\text{LU}} n_{\text{LU}}}_{\text{LU factorizations}} + \overbrace{c_{\text{MDA}} n_{\text{MDA}}}^{\text{MDA}} \tag{19}$$

where c_{lin} represents the time ratio between a linearization and an analysis, c_{LU} represents the time ratio between an LU factorization and an analysis, and c_{MDA} represents the time ratio between an MDA and an analysis. These time ratios can be estimated on the original problem.

4.2. Convergence criterion of the optimization algorithm

According to Johnson,[10] the optimum value (within a given tolerance) known a posteriori is not a valid termination criterion for the optimization algorithm, since it cannot be applied in practice on real problems.

Figure 14 portrays the evolution of the residual on the iterates of MDF and IDF for a scalable instance $(n_x, n_y, d) = (2, 8, 0.4)$ of the SSBJ problem (see Section 4.4) with the iterations. It highlights the presence of oscillations around the optimal solution for IDF, probably due to the difficulty to numerically satisfy equality constraints at convergence. The history of the objective function may therefore exhibit undesired plateaus at the end of convergence, which biases the comparison between architectures.

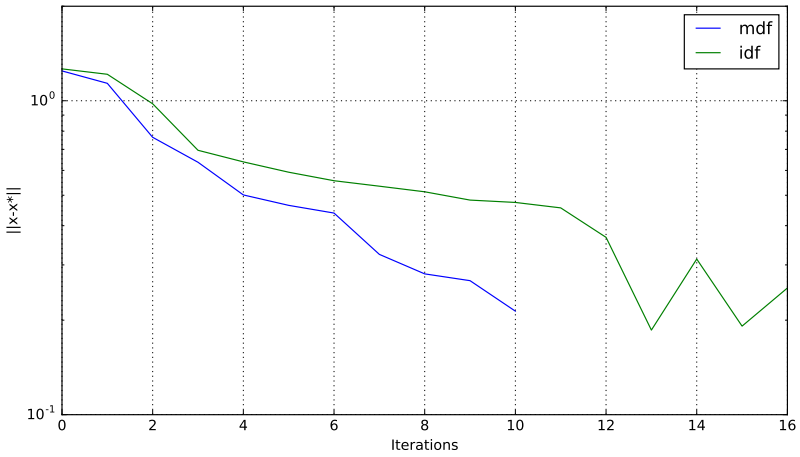


Figure 14: Evolution of the residual on the optimization iterates of MDF and IDF with the iterations

Our approach to address this is twofold. First, we impose strict relative and absolute tolerances (equal to 10^{-12} on both the solution and its objective value), so that the optimization algorithm converges towards identical values for both architectures on a given instance. Then, we perform an a posteriori filtering of the optimization histories by removing any later iterations that do not improve the solution by at least 10^{-6}. This eliminates undesired plateaus at the end of the convergence. The corresponding number of evaluations, linearizations, and MDAs are truncated accordingly.

4.3. Implementation of the architectures

The MDO architectures and test cases are implemented in the MDO platform of the MDA-MDO project at IRT Saint Exupéry.[11] In particular, GEMS (Generic Engine for MDO Scenarios), an in-house Python framework, is used to generate the optimization processes and host the present methodology.

4.3.1. MDF

The strongly and weakly coupled disciplines are computed via analysis of the coupling graph (see Figure 15 illustrating the SSBJ problem). Tarjan's algorithm[12] identifies the strongly connected components of the coupling graph, that is the subgraphs in which every vertex is reachable from every other vertex. A coupling problem is then created for each strongly connected component. The processed graph (see Figure 16 illustrating the SSBJ problem) represents the sequential evaluation of an MDA (equilibrium between the strongly coupled disciplines) and a weakly coupled discipline. The weakly coupled discipline is therefore not evaluated within the MDA.

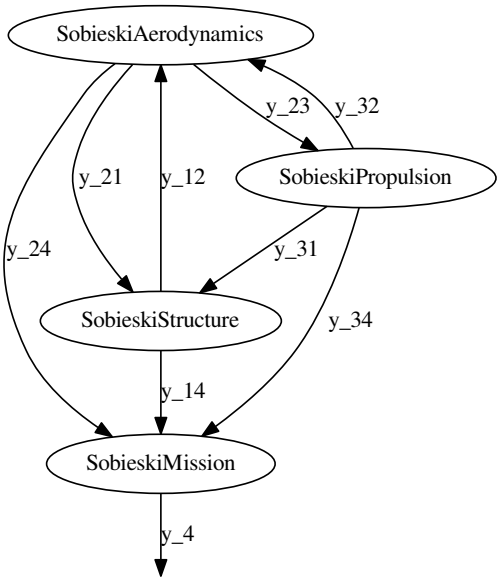


Figure 15: Initial graph of the SSBJ problem

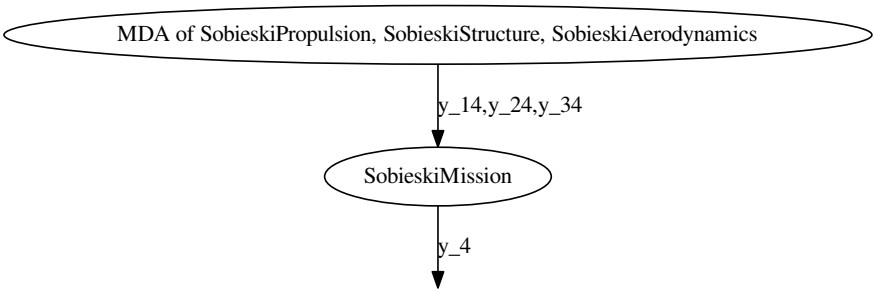


Figure 16: Processed graph of the SSBJ problem

4.3.2. IDF

The initial values of the coupling variables y^t that are driven by the optimization algorithm have a considerable influence on the convergence. We initialize y^t at equilibrium by computing an MDA for the initial design variable values, $y^{t(0)} = y(x^{(0)})$. Consequently, the equality consistency constraints (Section 1.2.2) are satisfied at the initial point.

4.4. Supersonic Business Jet

4.4.1. Problem description

The Supersonic Business Jet (SSBJ) MDO problem was introduced in the first publication on the BLISS architecture,[3] and is now a benchmark for MDO methods. The problem consists in maximizing the range of an SSBJ subject to various constraints. It is composed of three disciplines (structure, aerodynamics, and propulsion) based on semi-empirical and analytical models.[13,14] All disciplines depend on local design

variables, shared design variables and coupling variables, and compute disciplinary constraints. A fourth discipline, which is weakly coupled to the others, computes the range using the Breguet–Leduc formula. The dimensions of the inputs and outputs of the disciplines are fixed, and lower than 15. Table 2 provides the initial design and the solution of the SSBJ problem.

Table 2: Reference results of BLISS architecture applied to the SSBJ problem

Variable	Initial value	Optimum value
Range (nm)	**535.79**	**3,963.88**
λ	0.25	0.38757
x	1	0.75
C_f	1	0.75
Th	0.5	0.15624
t/c	0.05	0.06
h (ft)	45,000	60,000
M	1.6	1.4
AR	5.5	2.5
Λ (deg)	55	70
S_W (ft^2)	1,000	1,500

4.4.2. One-dimensional restrictions

Assisted by the methodology described in Section 2, we generated scalable analytic replacement functions based on the SSBJ problem. Figure 17 and Figure 18 display the one-dimensional interpolations of the objective function and the outputs (constraints and couplings) of the problem.

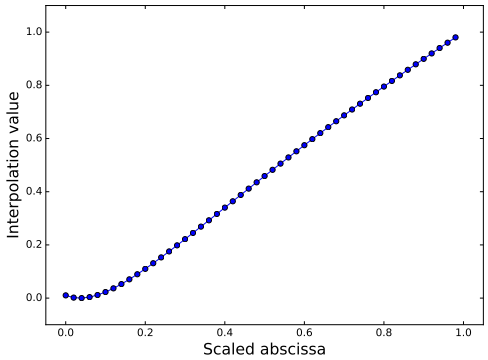
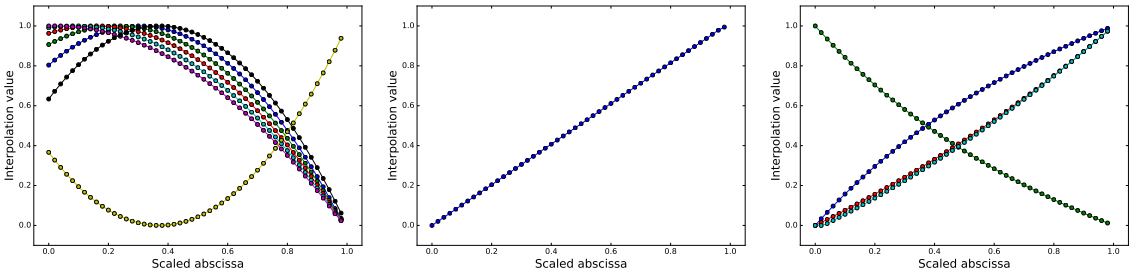
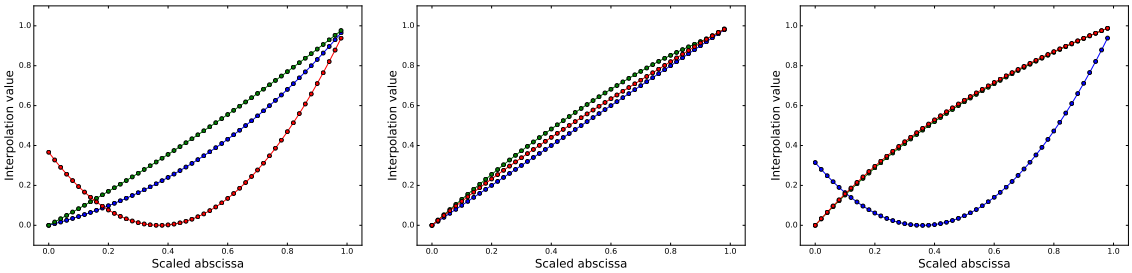


Figure 17: Interpolation of SSBJ objective function

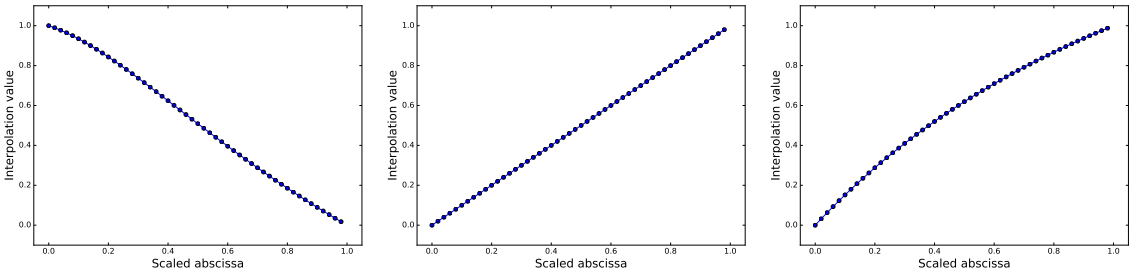
For a comparison and qualitative validation of the methodology, Figure 19 represents samplings of the original SSBJ functions when varying a single input in its full range of variation. A subset of these numerous (150) figures has been selected, among the representative ones in terms of variations, linearity, etc. Besides, all the constant outputs have been removed. Figure 18 and Figure 19 are qualitatively similar, demonstrating that the methodology conserves the functional characteristics: magnitude of variations, linearity, smoothness, and multimodality. Two specific functions in Figure 19 can cause problems to gradient-based algorithms: $y_{23} = f(y_{12})$ and $y_{24} = f(y_{12})$, since they are constant on a certain range, and then quadratic. Since we want to use gradient-based algorithms in the next experiments in order to address optimization problems with nearly 1000 design variables, we reduced the input range of these functions, to crop the ranges where the functions are constant. This preserves the original solution, which is not in these ranges. The present methodology could be used with the original intervals, but then, suitable optimization algorithms should be used, which may not scale up to such high dimension. And finally, the aim of this work is not to benchmark optimization algorithms but to compare MDO architectures, which justifies this choice.



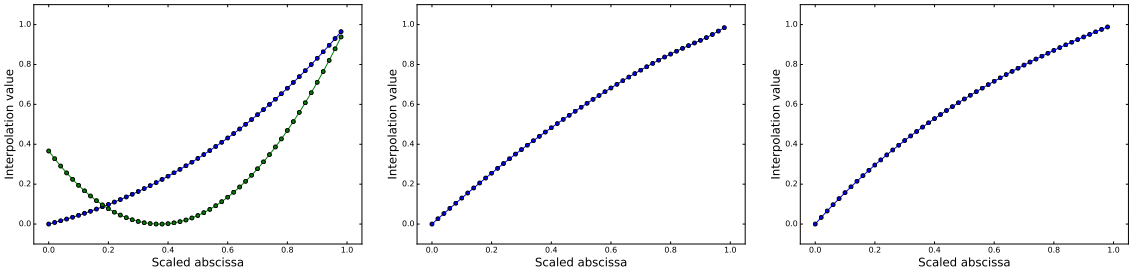
(a) g_1 (b) g_2 (c) g_3



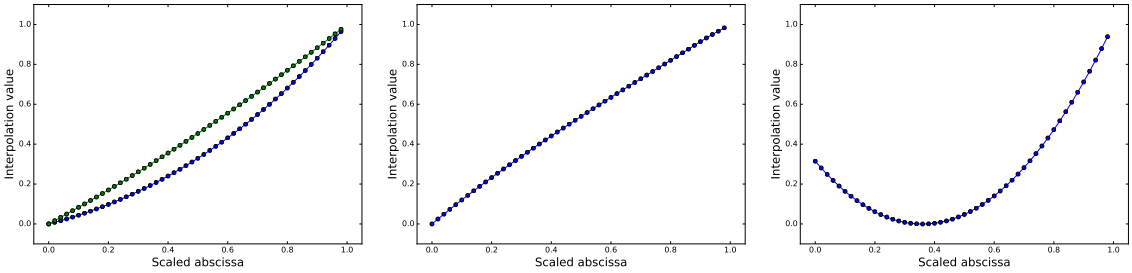
(d) y_1 (e) y_2 (f) y_3



(g) y_{11} (h) y_{21} (i) y_{31}



(j) y_{12} (k) y_{23} (l) y_{32}



(m) y_{14} (n) y_{24} (o) y_{34}

Figure 18: Interpolations of the SSBJ functions

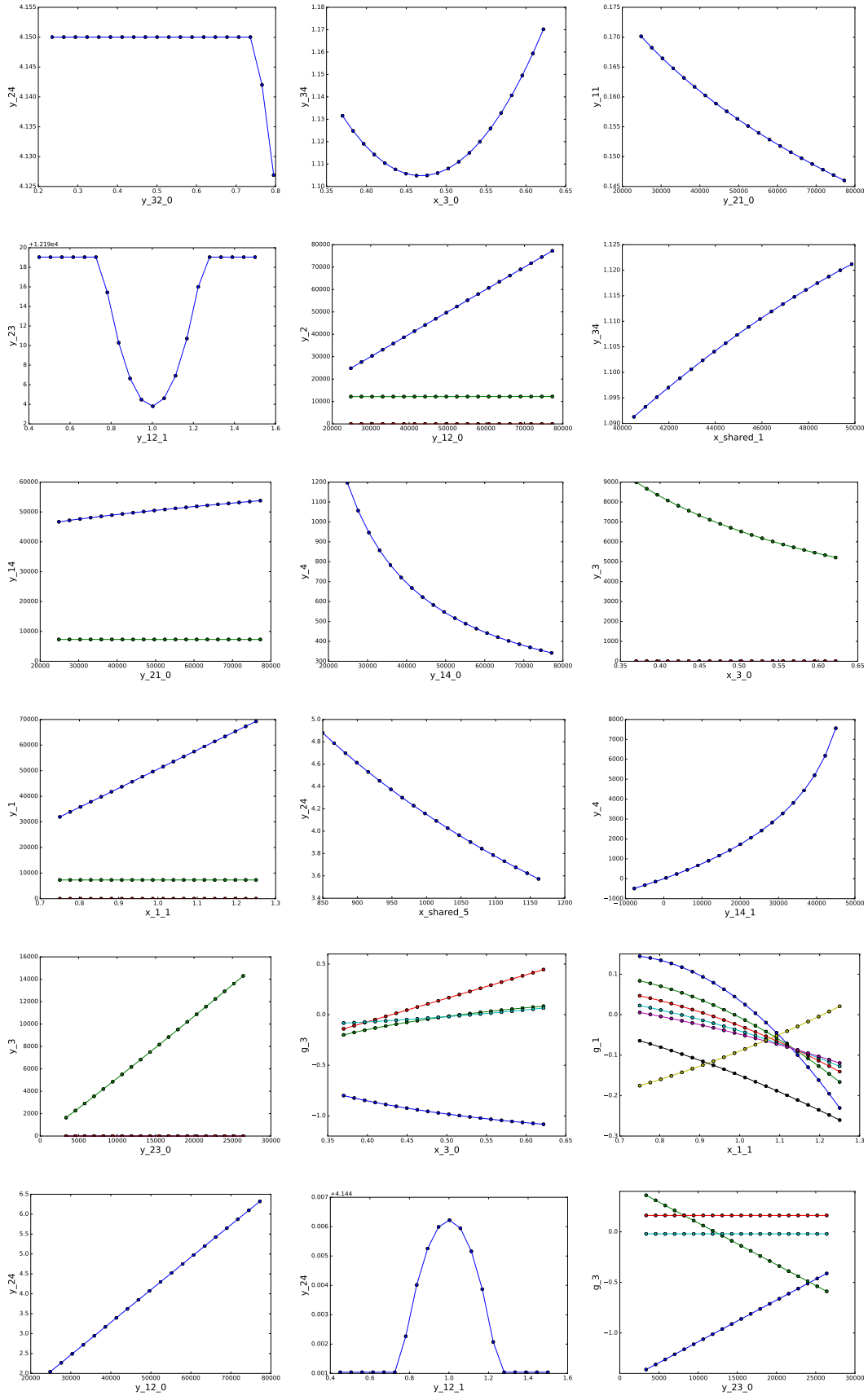


Figure 19: Some representative cuts of the original SSBJ functions

4.4.3. Parametric study of the dimensions of design variables and couplings

Figure 20 shows the evolution of the objective history with the optimization iterations for IDF and MDF, for a particular instance $(n_x, n_y) = (6, 16)$. The density factor d is set to 0.4, which is similar to that of the Jacobian matrix of the original SSBJ problem. The fixed-point algorithm invoked to converge the MDA in MDF is the Gauss-Seidel method.

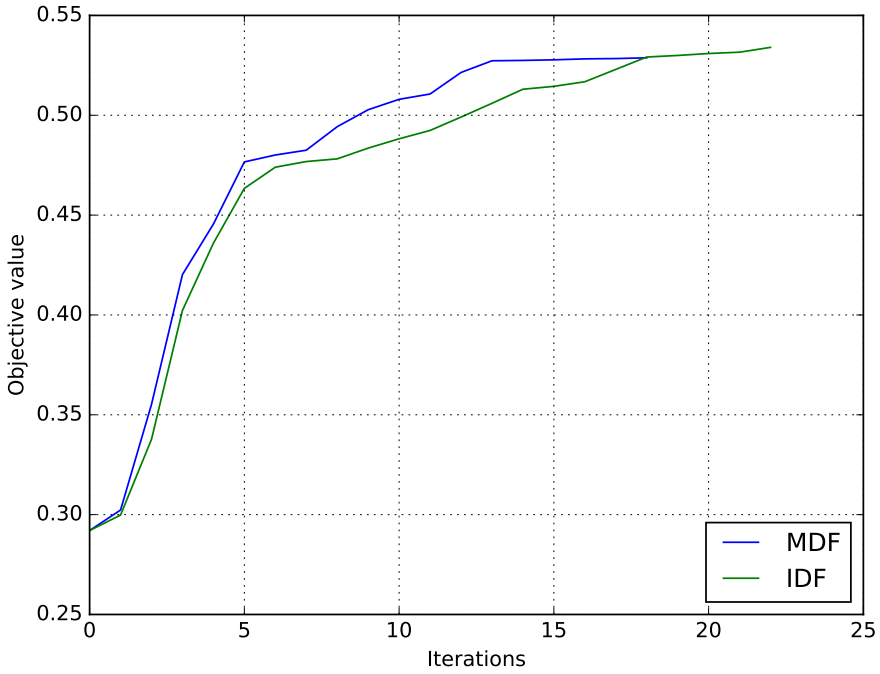


Figure 20: Objective function history for MDF and IDF optimizations with $(n_x, n_y, d) = (6, 16, 0.4)$

Figure 21 and Figure 22 display the constraint histories on the same instance for IDF and MDF, respectively. They represent the evolution of the values of the physical inequality constraints with the optimization iterations. Red indicates that a constraint is violated, green indicates that a constraint is satisfied and inactive, and white indicates that a constraint is active.

Figure 23 and Figure 24 show the cost for both architectures with a wide range of dimensions. The cost is estimated using Equation (19), with coefficients $c_{\text{lin}} = 0.5$, $c_{\text{MDA}} = 2$, $c_{\text{LU}} = 2$ that represent a typical aerostructural MDO problem. The abscissa represents the total size N_y of the coupling variables, and the multiple curves represent the two architectures over various numbers N_x of design variables. Similarly to the results obtained by Martins et al.,[5] IDF performs better for this problem, for the same dimension values. Table 3 shows the breakdown of the cost estimate.

Based on these results, we observe that on average, the cost increases with N_x and N_y, as expected. On average, IDF is more efficient than MDF, which is probably specific to the SSBJ problem. Particular instances, such as $(N_x = 20,\ N_y = 160)$ and $(N_x = 80,\ N_y = 120)$, seem harder than others, and do not satisfy the general trend. They prove that there is *no free lunch for MDO architectures*: a particular MDO architecture does not surpass other architectures on all instances. However, the performance of an MDO architecture depends on the dimensions of the design or coupling variables.

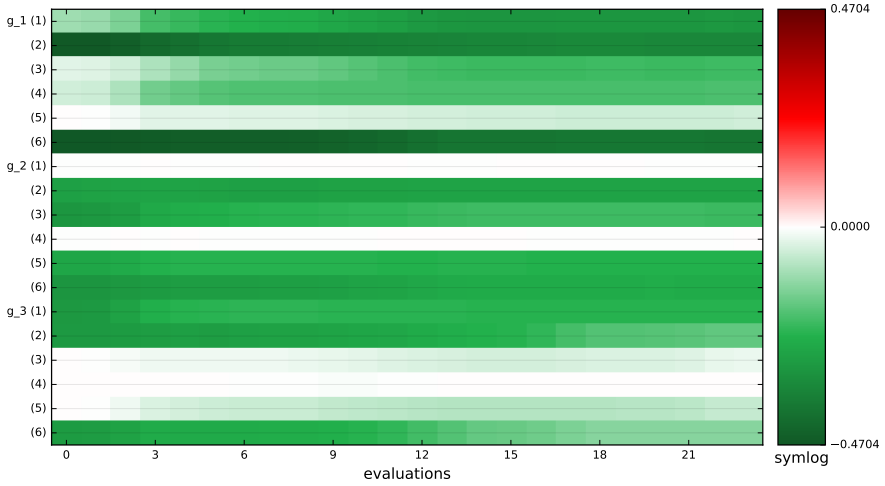


Figure 21: History of the constraints for IDF with $(n_x, n_y, d) = (6, 16, 0.4)$

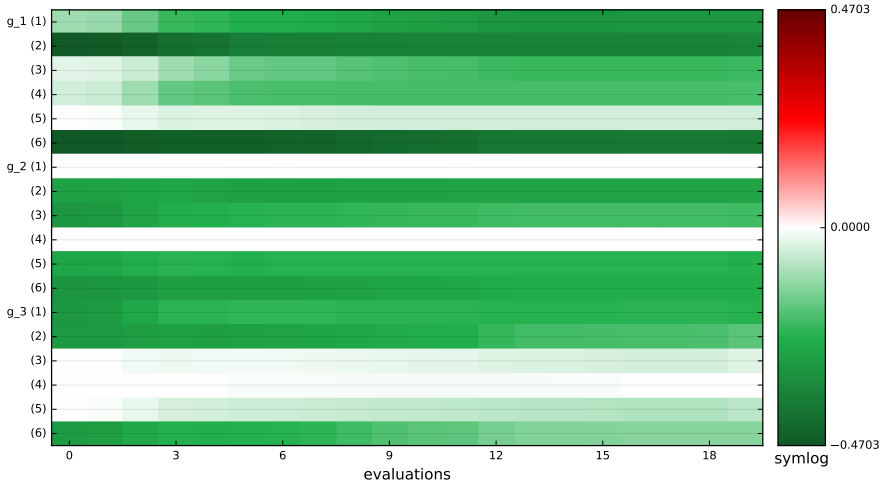


Figure 22: History of the constraints for MDF with $(n_x, n_y, d) = (6, 16, 0.4)$

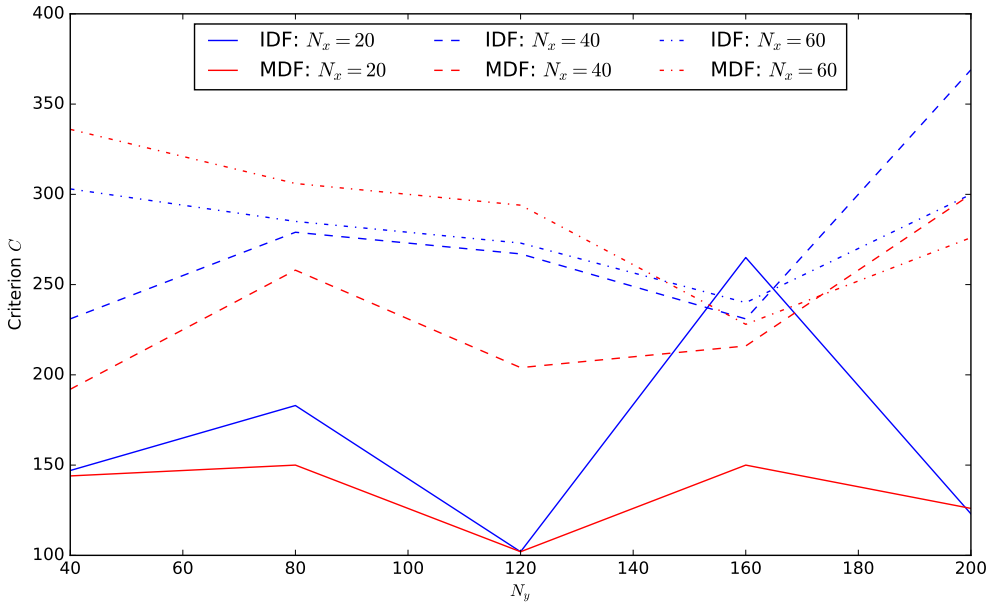


Figure 23: Estimated cost for MDF and IDF optimizations on the SSBJ test case with $N_x \in [20, 60]$ and $N_y \in [40, 200]$

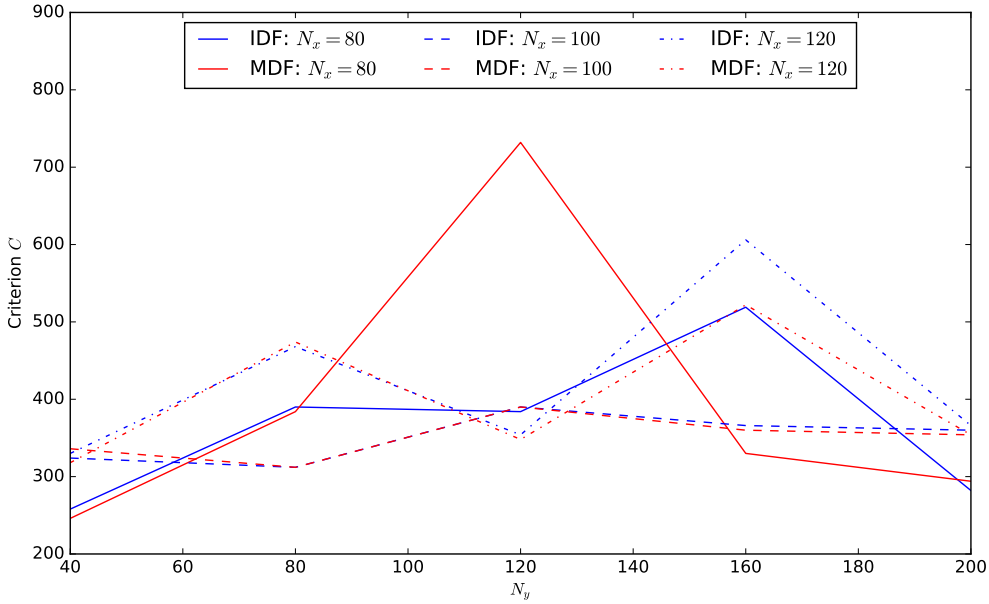


Figure 24: Estimated cost for MDF and IDF optimizations on the SSBJ test case with $N_x \in [80, 120]$ and $N_y \in [40, 200]$

Table 3: Statistics for MDF and IDF with respect to the dimensions for the SSBJ problem

		MDF				IDF		
N_x	N_y	n_{MDA}	$\sum_i n_{\text{lin}_i}$	n_{LU}	C	$\sum_i n_{\text{eval}_i}$	$\sum_i n_{\text{lin}_i}$	C
8	16	10	40	10	140	65	68	133
8	40	8	32	8	112	53	56	109
8	64	24	96	24	336	64	64	128
8	88	10	40	10	140	41	44	85
8	112	13	36	9	146	40	40	80
8	136	12	48	12	168	53	56	109
20	16	18	72	18	252	104	104	208
20	40	17	64	16	229	65	68	133
20	64	23	92	23	322	121	124	245
20	88	28	112	28	392	217	216	433
20	112	23	72	18	277	117	120	237
20	136	28	112	28	392	161	164	325
32	16	46	184	46	644	193	196	389
32	40	34	136	34	476	125	128	253
32	64	27	108	27	378	105	108	213
32	88	26	104	26	364	108	108	216
32	112	47	188	47	658	125	128	253
32	136	29	116	29	406	120	120	240
44	16	32	128	32	448	172	172	344
44	40	42	168	42	588	221	224	445
44	64	48	192	48	672	185	188	373
44	88	26	104	26	364	108	108	216
44	112	37	148	37	518	133	136	269
44	136	46	184	46	644	229	232	461
56	16	41	164	41	574	209	212	421
56	40	49	176	44	641	217	220	437
56	64	50	200	50	700	225	228	453
56	88	31	124	31	434	140	140	280
56	112	39	156	39	546	156	156	312
56	136	37	148	37	518	145	148	293
68	16	50	200	50	700	209	212	421
68	40	44	176	44	616	189	192	381
68	64	41	164	41	574	153	156	309
68	88	50	200	50	700	200	200	400
68	112	66	264	66	924	220	220	440
68	136	63	252	63	882	248	248	496

4.5. Propane combustion problem

4.5.1. Problem description

The propane combustion problem was transcribed from the NASA MDO test suite.[7,15] It describes the chemical equilibrium reached during the combustion of propane in air. The variables represent each of the ten combustion products as well as the sum of the products.

In the original problem, a system of eleven nonlinear equations must be solved. To obtain an MDO problem, the equations were divided into three components such that:

- coupling exists between the disciplines ;
- the evaluation of the objective function and constraints requires the solution of the coupled system of equations.

The problem has a single shared design variable ($x_0 \in \mathbb{R}$) and invokes three disciplines, governed by 2, 2, and 3 residual equations, respectively:

$$\begin{aligned}
&\text{Discipline 1} \begin{cases} x_0 + y_1 - 3 = 0 \\ y_0 y_1 - x_0 y_2 = 0 \end{cases} \\
&\text{Discipline 2} \begin{cases} 0.1 x_0 - y_1 y_3 \frac{40}{y_6} = 0 \\ 0.1 x_0^2 - y_1^2 y_5 \frac{40}{y_6} = 0 \end{cases} \\
&\text{Discipline 3} \begin{cases} 2(y_0 + y_2) + x_2 + x_3 - 8 = 0 \\ 2x_1 + y_4 - 40 = 0 \\ y_6 - x_0 - x_1 - x_2 - x_3 - y_0 - y_1 - y_2 - y_3 - y_4 - y_5 = 0 \end{cases}
\end{aligned} \tag{20}$$

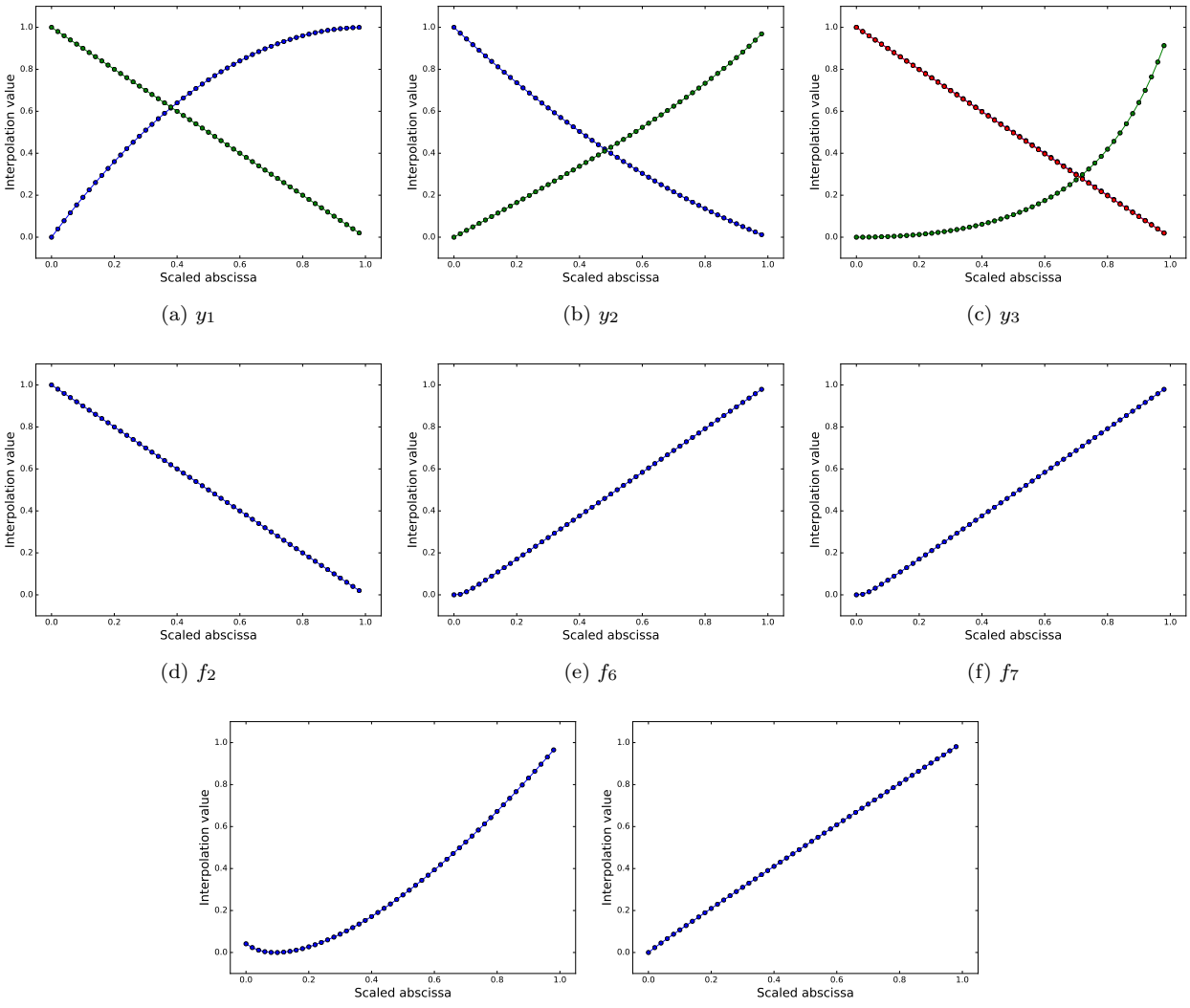
where $y = (y_0, y_1, y_2, y_3, y_4, y_5, y_6) \in \mathbb{R}^7$ are coupling variables. The local design variables $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ are used only in the third discipline. The propane combustion optimization problem is defined as follows:

$$\begin{aligned}
\underset{x,y}{\text{minimize}} \quad & f_2(x,y) + f_6(x,y) + f_7(x,y) + f_9(x,y) \\
\text{where} \quad & f_2(x,y) = 2x_0 + x_3 + y_0 + y_1 + y_3 + y_4 + 2y_5 - 10 \\
& f_6(x,y) = \sqrt{y_0 y_1} - x_2 \sqrt{\tfrac{40 x_0}{y_6}} \\
& f_7(x,y) = \sqrt{x_0 y_0} - x_3 \sqrt{\tfrac{40 y_1}{y_6}} \\
& f_9(x,y) = x_0 \sqrt{x_1} - y_1 y_4 \sqrt{\tfrac{40}{y_6}} \\
\text{subject to} \quad & 0 \le f_2(x,y), 0 \le f_6(x,y), 0 \le f_7(x,y), 0 \le f_9(x,y) \\
& 0 \le x \\
\text{residuals} \quad & R_1 : x_0 + y_1 - 3 = 0 \\
& R_2 : y_0 y_1 - x_0 y_2 = 0 \\
& R_3 : 0.1 x_0 - y_1 y_3 \tfrac{40}{y_6} = 0 \\
& R_4 : 0.1 x_0^2 - y_1^2 y_5 \tfrac{40}{y_6} = 0 \\
& R_5 : 2(y_0 + y_2) + x_2 + x_3 - 8 = 0 \\
& R_6 : 2x_1 + y_4 - 40 = 0 \\
& R_7 : y_6 - x_0 - x_1 - x_2 - x_3 - y_0 - y_1 - y_2 - y_3 - y_4 - y_5 = 0
\end{aligned}$$

The optimal solution is $x^* = (1.378887, 18.426810, 1.094798, 0.931214)$, and the minimum objective value is 0. All inequality constraints are active at this solution.

4.5.2. One dimensional restrictions

Similarly to Figure 18, the interpolated functions of the propane combustion problem are plotted in Figure 25. Again, for comparison, some samplings of the original propane combustion problem functions are plotted in Figure 26.



(a) y_1 (b) y_2 (c) y_3

(d) f_2 (e) f_6 (f) f_7

(g) f_9 (h) obj

Figure 25: Interpolations of the propane problem functions

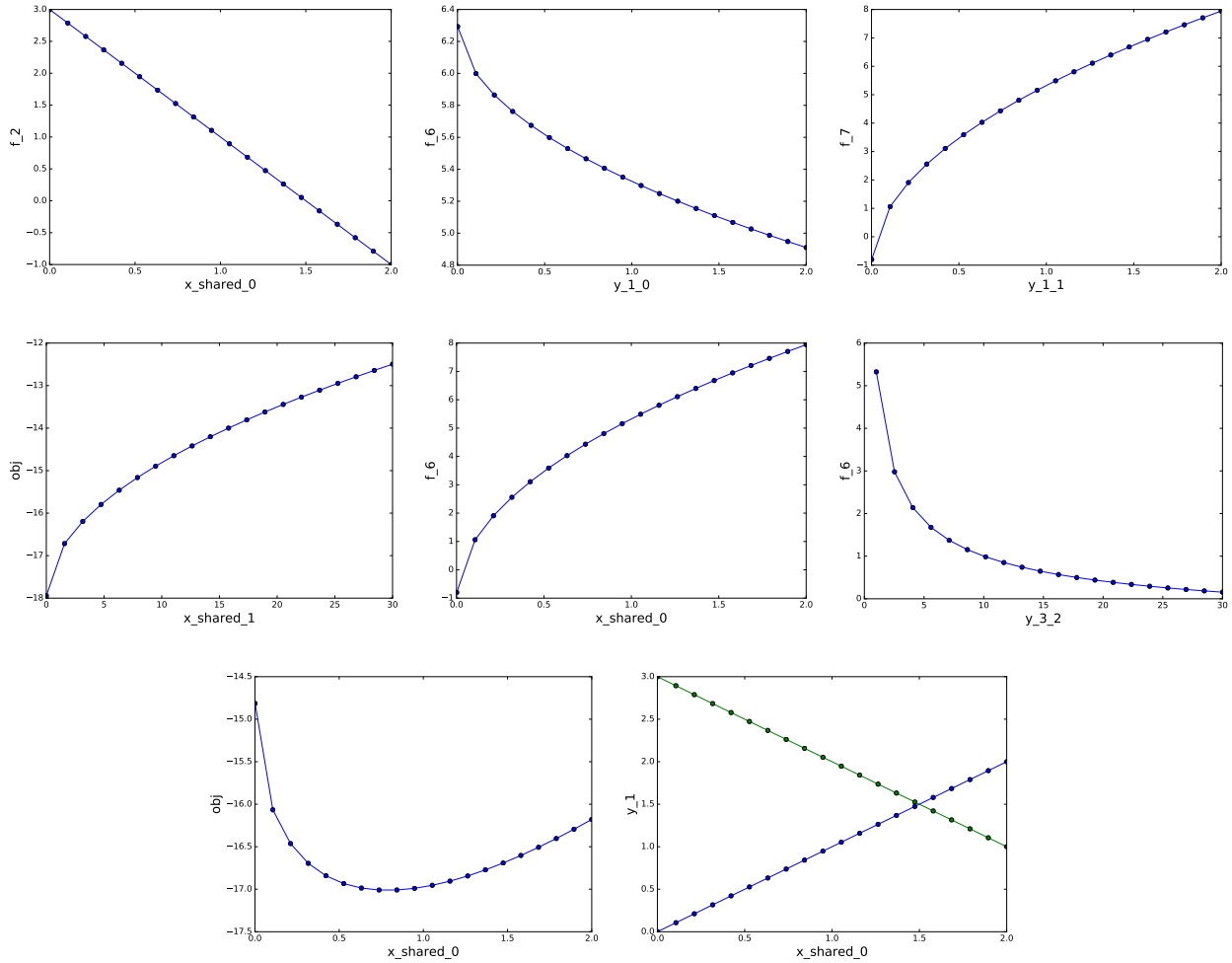


Figure 26: Representative cuts of the original propane combustion functions

4.5.3. Parametric study of the dimensions of design variables and couplings

A scalable problem based on the original propane combustion problem is generated. Figure 27 shows the cost estimates for IDF and MDF architectures for a wide range of dimensions for design and coupling variables. The density factor d is set to 0.7. The fixed-point algorithm invoked to converge the MDA in MDF is the Gauss-Seidel method. The cost coefficients are set to $c_{\text{lin}} = 0.5$, $c_{\text{MDA}} = 4$, $c_{\text{LU}} = 2$, which approximate the relative costs of the original problem. We set $c_{\text{MDA}} = 4$ since the disciplines are analytical and weakly coupled, so the MDA consists in the sequential execution of the four disciplines, which costs four times that of the execution of one discipline.

On average, the cost increases with N_x and (linearly with) N_y, as expected. IDF proves more efficient than MDF for $N_y < 100$, while MDF outperforms IDF for $N_y > 100$. This attests that the performance of the architectures is use case dependent, since we obtained a different trend for the SSBJ test case. Similarly to the SSBJ test case, particular instances, such as ($N_x = 180$, $N_y = 225$), seem computationally harder than others and do not follow the general trend. Again, this proves that there is no free lunch for MDO architectures, and that a particular MDO architecture is not more appropriate than other architectures on all instances.

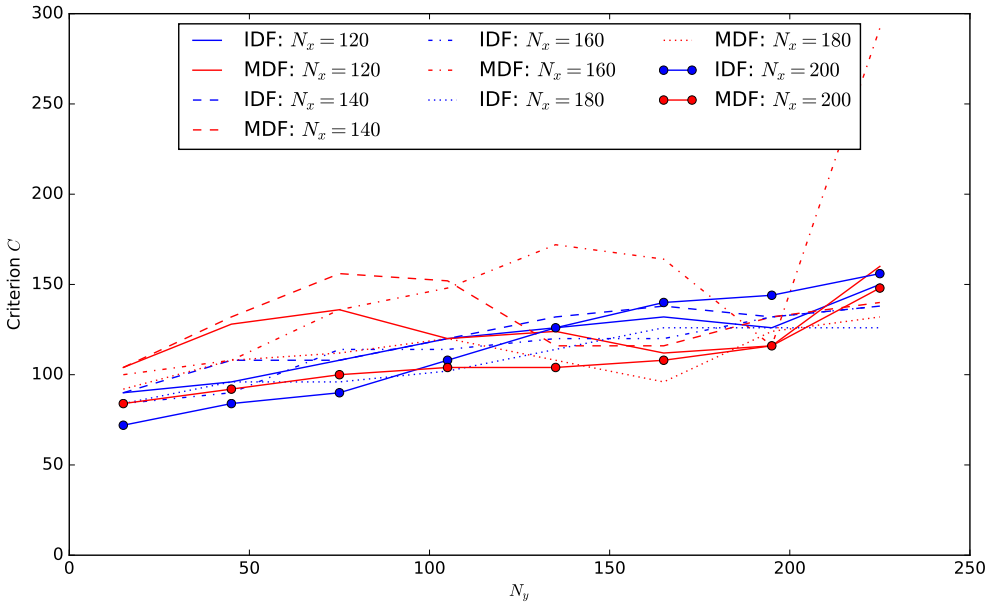


Figure 27: Estimated cost for MDF and IDF optimizations for the propane test case with $N_x \in [120, 200]$ and $N_y \in [15, 225]$

5. Conclusion and perspectives

A new method for deriving scalable analytic problems based on a given physical problem is proposed. The analytic replacement problems preserve the functional characteristics of the original problem and they proved useful in performing a rapid benchmarking of MDO architectures.

This approach is then used to construct two scalable problems: one based on the SSBJ problem, and another based on a propane combustion problem. Numerical experiments on these two problems showed that the choice of an MDO architecture for solving a particular problem, based on performance criteria, proved to be dependent on multiple parameters, such as the dimensions of the design variables and coupling variables. A modification of these parameters may alter the performance of the architectures, thus changing their relative ranks in a benchmark. Therefore, any comparison between MDO architectures should take into account multiple design variables dimensions, coupling variables dimensions, and multiple MDO problems.

For a particular MDO problem, the most appropriate MDO architecture may be chosen using our methodology. The set of available MDO architectures was limited to MDF and IDF in this paper, but may be extended to other architectures such as BLISS and CO, or even in-house architectures.

The proposed methodology provides insights on the scalability of MDO architectures with respect to the dimensions of the problem. This may be achieved without having to execute the MDO processes with the original models. Our methodology thus requires a limited number of evaluations of the original models that is independent of the desired dimensions of the design and the coupling variables of the scalable problem.

Glossary

BLISS Bilevel Integrated System Synthesis. 2

IDF Individual Disciplinary Feasible. 1, 2, 4, 5, 14, 15, 20–22, 27

MDA Multidisciplinary Analysis. 2, 4, 14–16, 20, 27

MDF Multidisciplinary Feasible. 1, 2, 4, 5, 13–15, 20–22, 27

SAND Simultaneous Analysis and Design. 2, 4

References

[1]Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, Vol. 51, No. 9, September 2013, pp. 2049–2075.

[2]Cramer, E. J., Dennis, Jr, J., Frank, P. D., Lewis, R. M., and Shubin, G. R., "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, 1994, pp. 754–776.

[3]Sobieszczanski-Sobieski, J., Agte, J. S., and Jr., R. R. S., "Bi-Level Integrated System Synthesis (BLISS)," Tech. Rep. TM-1998-208715, NASA, Langley Research Center, 1998.

[4]Wolpert, D. H. and Macready, W. G., "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, Vol. 1, No. 1, 1997, pp. 67–82.

[5]Martins, J. R. R. A. and Marriage, C., "An object-oriented framework for multidisciplinary design optimization," *3rd AIAA Multidisciplinary Design Optimization Specialist Conference*, 2007.

[6]Tedford, N. P. and Martins, J. R. R. A., "Benchmarking Multidisciplinary Design Optimization Algorithms," *Optimization and Engineering*, Vol. 11, No. 1, 2010, pp. 159–183.

[7]Padula, S. L., Alexandrov, N., and Green, L. L., "MDO Test Suite at NASA Langley Research Center," *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 1996, AIAA 1996-4028.

[8]Brouwer, L. E. J., "ber abbildung von mannigfaltigkeiten," *Mathematische Annalen*, Vol. 71, pp. 97–115.

[9]Park, S., "Ninety years of the Brouwer fixed point theorem," *Vietnam J. Math*, Vol. 27, No. 3, 1999, pp. 187–222.

[10]Johnson, D. S., "A theoretician's guide to the experimental analysis of algorithms," *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges*, Vol. 59, 2002, pp. 215–250.

[11]Gazaix, A., Gallard, F., and Gachelin, V., "Towards the Industrialization of New MDO Methodologies and Tools for Aircraft Design," *AIAA Aviation Conference*, June 2017.

[12]Tarjan, R., "Depth-first search and linear graph algorithms," *SIAM journal on computing*, Vol. 1, No. 2, 1972, pp. 146–160.

[13]Anderson, R., *Determination of the Characteristics of Tapered Wings*, NACA report, NACA, 1936.

[14]Raymer, D. P., *Aircraft Design: A Conceptual Approach (Education Series)*, AIAA (American Institute of Aeronautics & Ast, 2006.

[15]Tedford, N. P. and Martins, J. R., "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, Vol. 11, No. 1, February 2010, pp. 159–183.