Author version

Published as:

Orndorff, N. C., Lupp, C., & Hwang, J. T. (2026). A distributed algorithm for large-scale multidisciplinary design optimization with global constraints. In AIAA AVIATION 2026 Forum (Paper No. AIAA 2026-4500). https://doi.org/10.2514/6.2026-4500

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/orndorff2026distributed/

```
@inproceedings{orndorff2026distributed,
  author = {Nicholas C. Orndorff and Christopher Lupp and John T. Hwang},
  title = {A Distributed Algorithm for Large-Scale Multidisciplinary Design Optimization
    With Global Constraints},
  booktitle = {AIAA AVIATION 2026 Forum},
  year = {2026},
  doi = {10.2514/6.2026-4500},
  url = {https://doi.org/10.2514/6.2026-4500}
}
```

A Distributed Algorithm for Large-Scale Multidisciplinary Design Optimization with Global Constraints

Nicholas C. Orndorff*
University of California San Diego, La Jolla, CA, 92093

Christopher A. Lupp†
Air Force Research Laboratory, Wright-Patterson AFB, OH, 45433

John T. Hwang‡
University of California San Diego, La Jolla, CA, 92093

Distributed multidisciplinary design optimization (MDO) formulations are essential when engineering problems must be decomposed along disciplinary boundaries. However, existing approaches often lack convergence guarantees and struggle to handle global constraints efficiently. Penalty-based methods are widely used but they tend to introduce ill-conditioning and exhibit slow convergence, which limits their applicability to large-scale problems. We propose a distributed MDO formulation based on an augmented Lagrangian merit function that achieves exact global constraint satisfaction without requiring large penalty coefficients, thereby avoiding ill-conditioning. Distributed subproblems are solved using a block coordinate descent algorithm that is guaranteed to converge to stationary points of the original MDO problem. The formulation is constructed such that each subproblem can be solved without evaluating any other disciplinary models, preserving the strict disciplinary partitioning that motivates this work. This is accomplished by introducing copies of some design variables and intermediate quantities, which are computed within specific subproblems and then cached for use in subsequent subproblems. The proposed method is validated on two MDO problems: a simple aero-structural optimization and a more comprehensive optimization of a blended-wing-body aircraft. Both problems converge to correct solutions without numerical difficulties, demonstrating the potential of this approach for reliable, large-scale distributed MDO.

I. Introduction

Distributed multidisciplinary design optimization (MDO) formulations are often required when engineering problems must be decomposed along disciplinary boundaries. Such decompositions can be unavoidable in practice when, for example, disciplines are managed by distinct organizations or teams, when some disciplines rely on proprietary models, or when certain models are implemented in legacy codes [1–5].

Distributed formulations decompose an MDO problem into several coupled subproblems, each operating in a reduced subspace [6–8]. Because the subproblems are solved independently, their solutions are coordinated through an iterative algorithm to recover a consistent solution [9, 10]. A fundamental requirement of any such distributed formulation is that it must be mathematically equivalent to the original monolithic problem so that both admit the same solutions [6]. However, this requirement alone is not sufficient. An equally important and often overlooked requirement is that the iterative solution algorithm must converge to a stationary point. This depends on the complex interaction between the algorithm and the distributed formulation and, as a result, many distributed MDO formulations are equivalent to the original monolithic problem, but they cannot be guaranteed to converge to the correct solutions without imposing additional restrictive assumptions. This limitation affects several widely used formulations, including collaborative optimization [11], concurrent subspace optimization [12], and analytical target cascading [13].

To further complicate matters, when MDO problems are decomposed by discipline, they tend to have large numbers of global constraints. These constraints depend upon variables and models from multiple subproblems, so it is not

*PhD Candidate, Department of Mechanical and Aerospace Engineering, AIAA Student Member.
†Research Aerospace Engineer, Air Warfare Directorate, AIAA Associate Fellow.
‡Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA Senior Member.
Distribution A: Approved for public release; distribution is unlimited. AFRL-2026-2045

immediately clear how or where they should be accounted for in the distributed formulation. Resolving these global constraints efficiently and accurately is one of the hardest challenges for distributed MDO formulations. Numerous legacy methods rely on penalty functions to enforce these constraints directly in the objective function of each subproblem; however, this often leads to ill-conditioning and slow convergence because large penalty coefficients are required to satisfy feasibility. Penalty methods often fail for anything other than the simplest distributed MDO problems [14–16].

These challenges motivate the need for a distributed MDO formulation that (a) has convergence guarantees and (b) efficiently enforces global constraints. In this context, an efficient method will accurately satisfy the global constraints without introducing numerical ill-conditioning or other issues that hinder reliable convergence, particularly for large-scale problems. These properties are required if distributed optimization is to become a practical and widely adopted tool for state-of-the-art MDO applications.

To address this need, we propose an approach in which each distributed subproblem minimizes an augmented Lagrangian merit function that directly penalizes any global constraints. This is still a penalty method, but crucially, the augmented Lagrangian framework achieves exact constraint satisfaction without requiring excessively large penalty parameters [17], thereby avoiding the root cause of ill-conditioning in other penalty-based approaches [18]. The distributed solution is obtained by sequentially solving each subproblem using a block coordinate descent (BCD) algorithm, which is guaranteed to converge to stationary points for a broad class of problems with bounded curvature [19]. This inner BCD loop is embedded within an outer loop that updates the Lagrange multipliers and penalty parameters according to the standard augmented Lagrangian method. The resulting algorithm has a primal-dual structure that is somewhat similar to distributed algorithms from other fields, such as the alternating direction method of multipliers [20], which has demonstrated success in other large-scale and distributed applications such as robotics [21] and machine learning [22].

To best satisfy the original motivation for this work, it should be possible for each subproblem to be solved without evaluating any disciplinary models from other subproblems. This is required if disciplinary models cannot be shared between subproblems, as might be the case if some models are proprietary. Achieving this strict partitioning can be complicated, because MDO problems often have a coupled structure in which disciplinary models and global constraints cannot be computed in isolation. We address this by suggesting several problem transformations that modify the original optimization problem to remove coupling by using copies of design variables and intermediate state variables. The equivalence of variable copies is enforced by adding consensus constraints to the augmented Lagrangian merit function. By removing the coupling, certain intermediate quantities can be computed once during a specific subproblem minimization, and then cached for use in subsequent subproblems. If the problem has been reformulated correctly, these cached quantities are invariant in all but one subspace, so they never need to be re-computed in any other subproblem. The result is a distributed formulation where disciplinary models maintain a sense of privacy because they are only ever evaluated in their respective subproblems. Any cached variables are stored in a common data structure that is updated after each subproblem solution. Rather than suggest a universal problem reformulation, we propose that coupling should be addressed on a discipline-by-discipline basis. This is because not all disciplines are coupled in ways that require reformulation, and in general, reformulations increase the problem size and complexity, which should be avoided if possible. In general, we observe that the most efficient problem reformulations are those that are tailored to the specific problem structure.

The privacy afforded by our distributed formulation is informal because it is a result of the partitioned structure of the distributed architecture. In a mathematical sense, the disciplines are not differentially private because it may be possible to reconstruct the disciplinary models from sufficient quantities of the deterministic subproblem solutions. The fundamental law of information recovery states that overly accurate answers to too many questions will inevitably destroy privacy [23]. This provable law has been demonstrated by algorithms for reconstruction attacks in distributed optimization (e.g., image reconstruction [24]). It is expected that any strategies for further improving the privacy of distributed optimization methods will also degrade the speed and accuracy of convergence. For example, a common method for improving the privacy of distributed optimization is to add random noise to the subproblem solutions, thereby statistically obscuring certain trends [25].

The principal contribution of our proposed approach is that (a) it is guaranteed to converge, and (b) it can efficiently resolve coupling due to global constraints. This results in a method that converges faster and more reliably on complicated and large-scale MDO problems, for which existing distributed MDO methods would be expected to struggle with numerical ill-conditioning and slow convergence. To this end, we demonstrate the performance of our method on two MDO problems that both converge to the correct solutions without any numerical difficulties. The first of these problems is a simple and low-fidelity aero-structural optimization problem, and the second is a more comprehensive and higher-fidelity optimization of a blended-wing-body aircraft. Convergence is not fast, so we present some additional

results showing how certain algorithmic hyper-parameters affect the time required to find a solution.

This paper is structured as follows. Section II introduces the mathematical problem formulation and provides background on block coordinate descent methods and their convergence guarantees. Section III introduces the augmented Lagrangian approach with which we enforce global constraints, as well as the complete algorithm for solving distributed MDO problems. Our two examples are presented in Section IV, with a discussion of their specific distributed implementations. This paper is summarized in Section V, including a discussion of limitations and future work.

II. Background

A. Mathematical preliminaries

We consider the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0, \end{aligned} \tag{1}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is a twice continuously differentiable objective function that is minimized with respect to the design variables $x \in \mathbf{R}^n$. The inequality and equality constraint functions are given by $g : \mathbf{R}^n \to \mathbf{R}^m$ and $h : \mathbf{R}^n \to \mathbf{R}^p$, respectively. The gradients of f, g, and h are assumed to be Lipschitz continuous; in other words, the curvature has an upper bound.

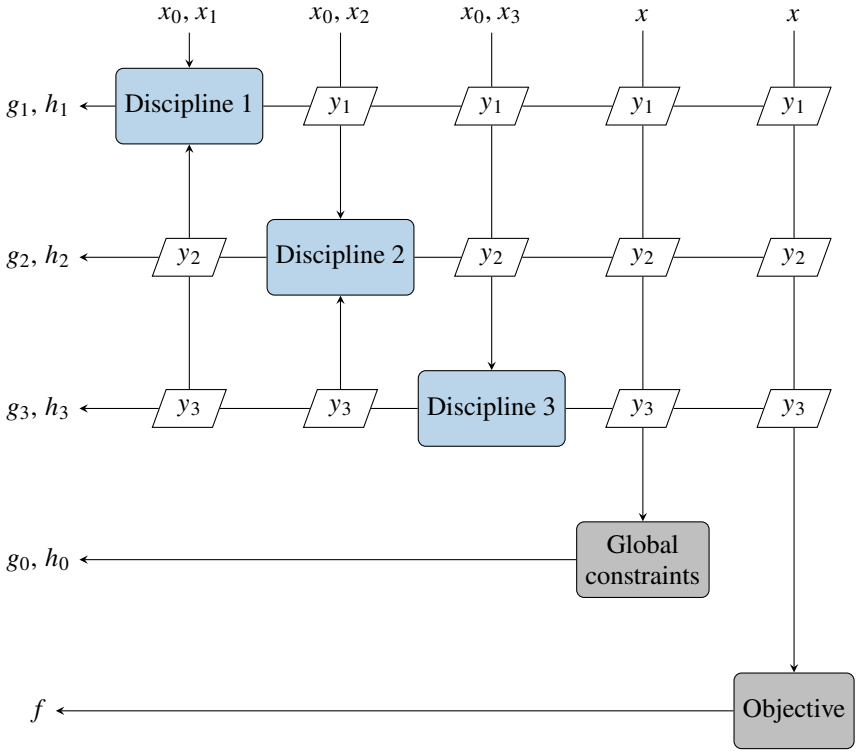

Fig. 1 The extended design structure matrix for an MDO problem with three disciplines [6].

In this work, the optimization problem is decomposed by discipline and solved using a distributed algorithm. This decomposition yields N subproblems as a direct consequence of the standard N^2 (or design structure matrix) representation (Fig. 1), where, for our purposes, disciplines are synonymous with subproblems. The N^2 diagram shows the data-dependency structure of the N disciplines, each of which has *local variables* $x_i \in \mathbf{R}^{n_i}$ which are unique inputs

to each subproblem, and *global variables* $x_0 \in \mathbf{R}^{n_0}$ which are shared across multiple subproblems. The full design variable vector is formed by concatenation: $x = [x_0, x_1, x_2, ..., x_N]^\top$.

The N^2 diagram can be interpreted as a computational graph describing the coupling between disciplines. Off-diagonal entries represent intermediate variables $y_i \in \mathbf{R}^{q_i}$ computed by evaluating the disciplinary models on the diagonal. Figure 1 illustrates the most general case, in which intermediate variables appear both above and below the diagonal. Following the conventions of Martins and Ning [6], entries above the diagonal feed forward, while entries below the diagonal feed backward. Any two disciplines are two-way coupled if they are linked by intermediate variables both above and below the diagonal, a scenario that results in a cyclic graph.

We distinguish between *local constraints* and *global constraints*. Local constraints depend only on a single block of variables x_i, and are denoted by $g_i(x_i) : \mathbf{R}^{n_i} \to \mathbf{R}^{m_i}$ and $h_i(x_i) : \mathbf{R}^{n_i} \to \mathbf{R}^{p_i}$. In contrast, global constraints depend on multiple variable blocks (e.g., x_i and x_j with $i \neq j$), and are denoted by $g_0(x) : \mathbf{R}^n \to \mathbf{R}^{m_0}$ and $h_0(x) : \mathbf{R}^n \to \mathbf{R}^{p_0}$. MDO problems often contain both local and global constraints, where global constraints are particularly problematic for distributed formulations.

It is assumed that all the design variables x and all the intermediate variables y are *public*, while the disciplines are *private*. In other words, every subproblem has unrestricted access to the current values of the design and intermediate variables for use in its own computations, but only the i-th subproblem can evaluate the models associated with discipline i. Likewise, the global constraints and the objective function are known to all subproblems.

B. Block coordinate descent methods

The monolithic optimization problem shown in Figure 1 is decomposed by discipline. This defines N subproblems, each of which minimizes the same objective function with respect to its local variables x_i, subject to local constraints $g_i(x_i)$ and $h_i(x_i)$ while treating all other variables (denoted x_{-i}) as fixed parameters. The i-th subproblem is:

$$\begin{aligned} \text{minimize} \quad & f(x_i; x_{-i}) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & g_i(x_i) \leq 0, \\ & h_i(x_i) = 0, \end{aligned} \tag{2}$$

where the monolithic problem must first be reformulated to remove any global variables (see Sec. III.B).

The N subproblems are solved iteratively using a block coordinate descent (BCD) algorithm. BCD is a Gauss–Seidel method in which each subproblem is solved sequentially using the most up-to-date values of all other variables—analogous to the Gauss–Seidel method for solving linear systems of equations. BCD is computationally decentralized in the sense that it does not have a hierarchical structure. This contrasts with many existing distributed MDO formulations, in which a higher-level optimization problem coordinates multiple lower-level subproblems [6–8]. Hierarchical formulations have weak convergence guarantees, often requiring convexity assumptions that are rarely satisfied in practice. In contrast, BCD admits stronger convergence guarantees for general non-convex problems. Convergence to stationary points of the original monolithic problem is guaranteed so long as:

1) each subproblem is solved exactly,
2) the gradients of the objective and any constraints are Lipschitz continuous,
3) the objective function is monotonically decreasing with respect to BCD iterations,

where some of these assumptions can be relaxed in practice [19]. Algorithm 1 shows a simple BCD implementation.

Algorithm 1 Block coordinate descent (BCD)

```
Initialize x ∈ R^n
while not converged do
    for i = 1, 2, ..., N do
        solve subproblem i
        update x
    end for
    check convergence
end while
```

The algorithm terminates when the relative step across one full BCD iteration becomes sufficiently small:

$$\max\left(|x^{k+1} - x^k|/|x^k|\right) \leq \eta, \tag{3}$$

where $\eta > 0$ is the BCD convergence tolerance and x^k is the value of x before the k-th BCD iteration. The step is normalized by the magnitude of the previous iterate to account for variations in the scale of variables. Any suitable variation of this convergence criterion can be substituted.

The BCD algorithm will not always converge to stationary points in the presence of global constraints [6] (e.g., Fig. 2). Therefore, it is necessary to reformulate problems so global constraints do not explicitly appear in the BCD subproblems.

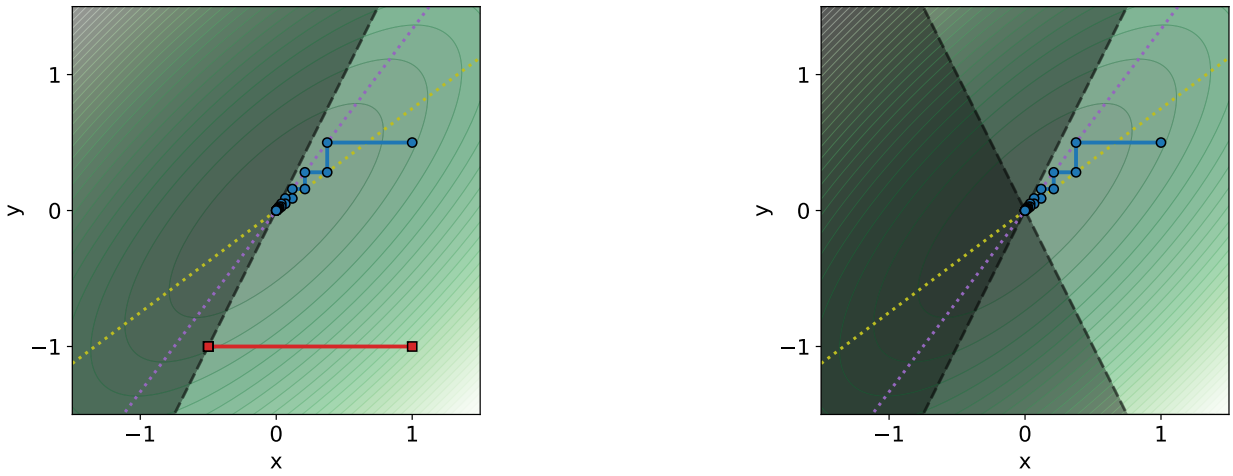

Fig. 2 Left: Block coordinate descent can get stuck at sub-optimal points in the presence of global constraints. Right: No feasible solution exists for the y minimization if initialized in the left-half plane.

III. Augmented Lagrangian methods for the distributed solution of MDO problems

A. An augmented Lagrangian relaxation of global constraints

Global constraints are notoriously difficult to resolve in a distributed formulation because each subproblem operates in a restricted subspace. One result of this is that subproblems might get stuck at sub-optimal points where there is no feasible solution (e.g., Fig. 2). Many distributed MDO formulations handle this issue by converting all global constraints into penalty terms that are added to the common objective function. However, the resulting optimization problems are extremely ill-conditioned and the global constraints are rarely exactly satisfied. For large-scale distributed MDO problems, these issues are exacerbated to the point that penalty methods rarely converge to correct solutions. Furthermore, convergence can be slow because the penalty functions create steep valleys in the design space that an optimizer must traverse to find a solution.

We propose to resolve global constraints using an augmented Lagrangian method. The augmented Lagrangian method uses penalty functions, but the critical difference is that it guarantees exact constraint satisfaction for some finite, positive value of the penalty parameter μ [17]. For more basic penalty methods, convergence to feasible solutions is only guaranteed as μ grows to infinity, which leads to Hessians that are arbitrarily ill-conditioned near stationary points [26]. Other penalty methods exist that guarantee exact constraint satisfaction, but they are commonly non-smooth variants of ℓ_1 norms (i.e., ReLU) [27, 28]. The augmented Lagrangian method preserves smoothness, so subproblems can be solved without violating any of the assumptions of gradient-based optimizers.

The standard augmented Lagrangian merit function [29] is:

$$\mathscr{L}(x; \lambda, \mu) := f(x) + \lambda^\top h_0(x) + \frac{\mu}{2}\|h_0(x)\|^2, \tag{4}$$

where for now we only consider the global equality constraints $h_0(x)$ with the corresponding Lagrange multipliers $\lambda \in \mathbf{R}^{p_0}$. In its standard form, the augmented Lagrangian merit function only supports a single scalar penalty parameter μ that is applied to all global constraints. This means that μ will continue to grow until the most difficult constraint is satisfied, which sometimes leads to solution error and slow convergence because other constraints are unnecessarily over-penalized. We address this by modifying the augmented Lagrangian merit function to support individual penalty parameters per scalar constraint using a positive-definite diagonal matrix $\mathbf{M} \in \mathbf{R}^{p_0 \times p_0}$:

$$\mathbf{M} = \begin{bmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_{p_0} \end{bmatrix}, \tag{5}$$

which yields a new augmented Lagrangian merit function:

$$\mathscr{L}(x; \lambda, \mathbf{M}) := f(x) + \lambda^\top h_0(x) + \frac{1}{2} h_0(x)^\top \mathbf{M} h_0(x), \tag{6}$$

which has the same quadratic structure as the original function, but is expected to further improve the problem's conditioning and to speed up convergence by keeping the quadratic term to a minimum.

Each distributed subproblem now minimizes the new augmented Lagrangian merit function, with respect to its own local variables x_i, and subject to only local constraints $g_i(x_i)$ and $h_i(x_i)$. The i-th subproblem is:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(x_i; x_{-i}, \lambda, \mathbf{M}) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & g_i(x_i) \leq 0 \\ & h_i(x_i) = 0, \end{aligned} \tag{7}$$

where x_{-i}, λ, and $\mathbf{M}$ are treated as parameters. The local constraints are permissible because, unlike the conventional augmented Lagrangian framework, we are not trying to create entirely unconstrained subproblems. BCD will converge with local constraints, so they are left in the subproblems to reduce the complexity of computing $\mathscr{L}$. Common examples of local constraints in MDO problems are upper and lower variable bounds, because they will always depend solely on local variables by definition. Local constraints remain private along with the disciplinary models.

The distributed solution that minimizes $\mathscr{L}$ with respect to x for any fixed parameters is found using the BCD algorithm (Alg. 1). We add a new outer loop to the BCD algorithm that updates the Lagrange multipliers and the penalty parameters using the standard approximation for the dual ascent step where $\lambda^* \approx \lambda^k + \mathbf{M}^k h_0(x^k)$ for any outer-loop iteration k. To hasten convergence, the penalty parameter μ_i for every scalar constraint h_{0_i} is multiplied by some $\rho > 1$ if feasibility is not improving at a sufficient rate and the current feasibility exceeds the feasibility tolerance ϵ. The two outer-loop update equations [30] are:

$$\begin{aligned} \lambda^{k+1} &= \lambda^k + \mathbf{M}^k h_0(x^k) \\ \mu_i^{k+1} &= \begin{cases} \rho \mu_i^k & \text{if } |h_{0_i}(x^k)| > \tau |h_{0_i}(x^{k-1})| \text{ and } |h_{0_i}(x^k)| > \epsilon \\ \mu_i^k & \text{else} \end{cases}, \text{ for } i = 1, ..., p_0, \end{aligned} \tag{8}$$

where $\tau \in (0, 1)$ is used to check whether sufficient improvements in feasibility have been made across the previous outer-loop iteration. A typical value for τ is 0.5.

Algorithm 2 has a double-loop structure, where the inner loop solves the primal problem to minimize $\mathscr{L}$ with respect to x for fixed dual parameters λ and $\mathbf{M}$, which are subsequently updated in the outer loop. The algorithm is guaranteed to converge to stationary points so long as the inner loop is fully converged. This follows from standard augmented Lagrangian convergence proofs (e.g., [29]). The algorithm terminates when feasibility falls below a certain threshold:

$$\|h_0(x^k)\|_\infty \leq \epsilon. \tag{9}$$

A challenge with the ALBCD-G algorithm is that each subproblem must be well-posed for any attainable values of the parameters $\mathbf{M}$, λ, and x_{-i}. In other words, each subproblem must reliably converge to solutions given parameters that may be far from optimal. Some subproblems will struggle with this, and to some extent issues can be mitigated by starting closer to a feasible solution. Two-sided variable bounds can also help to keep intermediate solutions reasonable, thereby avoiding erroneous results from evaluating disciplinary models at extremely sub-optimal points.

Algorithm 2 Augmented Lagrangian block coordinate descent for global constraints (ALBCD-G)

while outer loop not converged **do** ▷ Augmented Lagrangian outer loop
  while inner loop not converged **do** ▷ Block coordinate descent inner loop
    for $i = 1, 2, ..., N$ **do**
      solve subproblem i
      update x
    end for
    check inner loop convergence
  end while
  if feasible **then**
    break
  end if
  update λ using Eq. 8
  update $\mathbf{M}$ using Eq. 8
end while

Each subproblem can be solved with any choice of optimizer, so long as sufficiently accurate solutions can be obtained. We prefer SQP methods because they perform well when warm-started from previous solutions, which is exactly what occurs in the ALBCD-G algorithm. Interior point methods are notoriously difficult to warm-start [31], so convergence typically takes longer. Gradient-free optimizers could be used, but without any guarantee of convergence.

Now, suppose that in addition to equality constraints, there exist some global inequality constraints g_0 that we also wish to satisfy using the augmented Lagrangian method. Numerous methods have been proposed for adding inequality constraints to an augmented Lagrangian framework. The most commonly used approach is the Powell–Hestenes–Rockafellar (PHR) method [32]; however, the PHR method yields objective functions that are not twice continuously differentiable [33], which can cause convergence issues for subproblem optimizers. We therefore propose an alternative method that exploits the distributed problem structure.

We introduce a modified augmented Lagrangian merit function, where positive slack variables $s \in \mathbf{R}^m$ are used to convert the global inequality constraints into equality constraints:

$$\mathscr{L}(x, s; \lambda, \sigma, \mathbf{M}) := f(x) + \begin{bmatrix} \lambda \\ \sigma \end{bmatrix}^\top \begin{bmatrix} h_0(x) \\ g_0(x) + s \end{bmatrix} + \frac{1}{2} \begin{bmatrix} h_0(x) \\ g_0(x) + s \end{bmatrix}^\top \mathbf{M} \begin{bmatrix} h_0(x) \\ g_0(x) + s \end{bmatrix}, \tag{10}$$

where the Lagrange multipliers for the inequality constraints are denoted separately as $\sigma \in \mathbf{R}^{m_0}$, and the dimension of $\mathbf{M}$ is now $m_0 + p_0$. Then the slack variable is added to any single subproblem. For instance, the slack variable is added to subproblem one:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(x_1, s; x_{-1}, \lambda, \sigma, \mathbf{M}) \\ \text{with respect to} \quad & x_1,\ s \\ \text{subject to} \quad & g_1(x_1) \leq 0 \\ & h_1(x_1) = 0 \\ & s \geq 0, \end{aligned} \tag{11}$$

where the additional constraint $s \geq 0$ is permissible because it is now a local constraint.

B. Reformulating MDO problems to maintain disciplinary partitions

This work considers MDO problems that are decomposed by discipline, primarily because there are many scenarios where this type of decomposition is strictly required in industry applications—for example, when disciplinary models cannot be shared between organizations. In these scenarios, each subproblem might only have access to its own models, and it is therefore a requirement that each subproblem can be solved without evaluating disciplines or models from other subproblems. This can be difficult, because in the most general case (e.g., Fig. 1) there is interdisciplinary coupling through both state variables and global constraints. Therefore, before we use Algorithm 2, the optimization problem must be reformulated to remove certain coupling.

Disciplines may be one-way or two-way coupled. One-way coupling is not usually an issue, because any intermediate variable y_i can be cached for use in subsequent subproblems provided it remains invariant in all but the i-th subspace. Two-way coupling is more troublesome because it is usually resolved using an iterative process. The simplest way of handling two-way coupling in a distributed formulation is to consider the coupled disciplines as a single subproblem that must always be evaluated together. If this is not possible, then the coupling can be entirely removed from the problem using an individual-discipline-feasible (IDF) approach [34]. Figure 3 shows the IDF reformulation for two-way coupled disciplines using copies of intermediate variables. This reformulation requires the addition of a consensus constraint in the augmented Lagrangian merit function that enforces the equivalence of these copies in the outer loop.

Global variables affect disciplines across multiple subproblems, so they cannot be arbitrarily assigned to any single subproblem in a distributed formulation. To resolve this issue, Figure 4 shows how unique copies of global variables can be introduced to each relevant discipline such that changes to any global variable copy do not affect other disciplines. As in the IDF approach, a consensus constraint is added to the augmented Lagrangian merit function that enforces the equivalence of global variable copies in the outer loop.

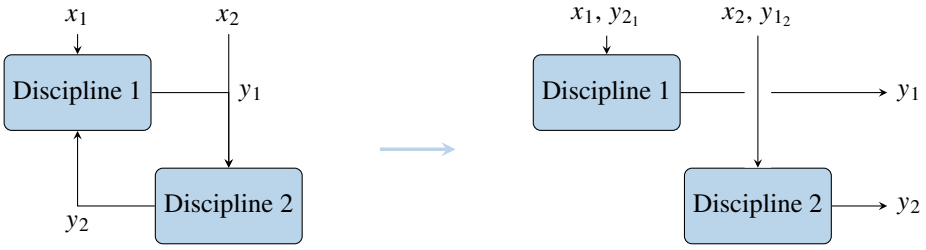

Fig. 3 Removing two-way coupling with intermediate-variable copies. A consensus constraint enforces that $y_1 = y_{1_2}$ and $y_2 = y_{2_1}$.

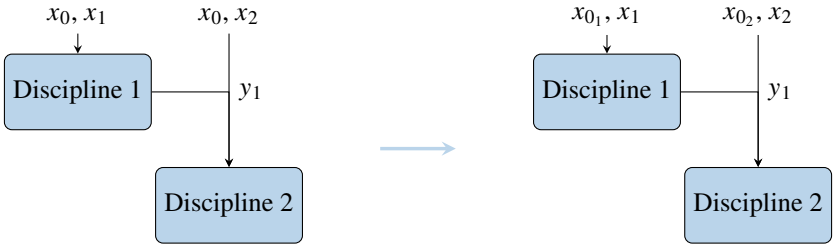

Fig. 4 Removing global variables with design-variable copies. A consensus constraint enforces $x_{0_1} = x_{0_2}$.

Every subproblem must evaluate the global constraints in order to compute the augmented Lagrangian merit function. This can add coupling between subproblems because evaluating the constraints requires solving models from multiple disciplines, which breaks the disciplinary partitioning. This type of coupling can be challenging to identify, because it is so dependent on the unique structure of the global constraint functions.

Some global constraints do not require reformulation. These uncoupled global constraints (Fig. 5) depend only on design variables and intermediate quantities that are invariant in all but one subspace, so they can be evaluated in every subproblem using cached values. For coupled global constraints, the IDF formulation can be applied to decouple the disciplines that provide inputs to the specific constraint functions. This process introduces copies of specific intermediate variables, which are added as unique design variables in every relevant subproblem (Fig. 6). The equivalence of the intermediate variable copies is enforced in the augmented Lagrangian merit function.

In practice, the structure of an MDO problem varies significantly. Therefore, the problem reformulations suggested here should be tailored to the specific problem structure. This is because problem reformulations enlarge the subproblem dimension and slow down convergence, so they should be avoided if they are not absolutely necessary. Wherever the reformulations are used, the variable copies are stored in a common data structure. After each subproblem solution, the data structure is updated with new values for any variable that has changed. This way, all subproblems are able to reference the common data structure, which always contains the most up-to-date values.

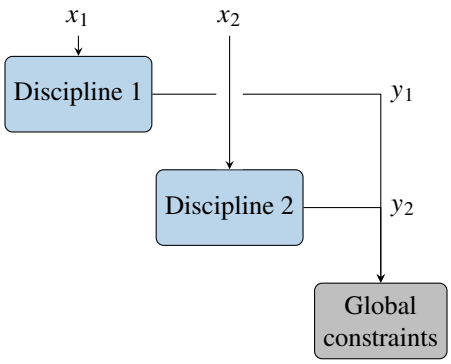

Fig. 5 Uncoupled global constraints do not require reformulation. The variable y_1 depends solely on x_1, and y_2 depends solely on x_2, so the constraints can be computed using cached values for both y_1 and y_2.

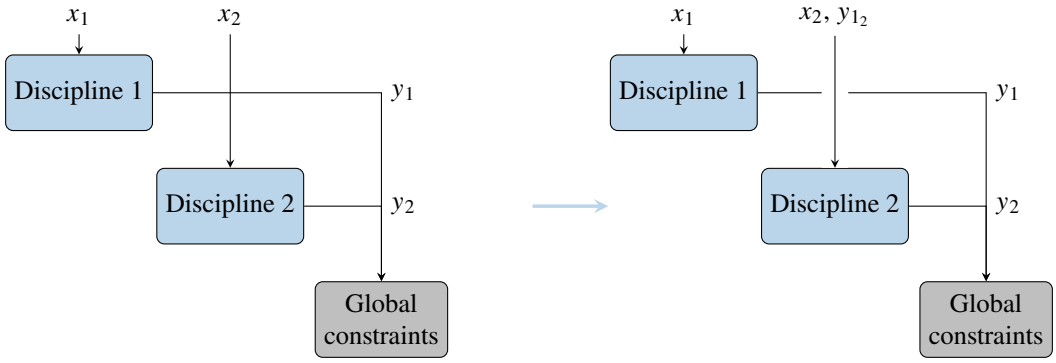

Fig. 6 Coupled global constraints (left) require an IDF reformulation (right).

IV. Results

A. Simple aero-structural optimization using ALBCD-G

Consider the following simple aero-structural optimization problem in which an unmanned aircraft is optimized for minimum drag coefficient:

$$\begin{aligned} \text{minimize} \quad & C_D \\ \text{with respect to} \quad & \theta,\ t \\ \text{subject to} \quad & L = W \\ & u_{\text{tip}} = u_{\max} \\ \text{while solving} \quad & A\Gamma = v \\ & Ku = f(\Gamma), \end{aligned} \tag{12}$$

where the circulation strength Γ is computed using a lifting-line model. The structural loads F are computed as a function of Γ and used to compute the displacement of the wing using an Euler-Bernoulli beam model. The aerodynamic design variables are the twist θ at each spanwise station, and the structural design variables are the thicknesses t defined at the same locations. This problem is one-way coupled; that is, the aerodynamic loads are used to compute structural displacements directly, with no iteration. Figure 7 shows the aerodynamic and structural disciplines, with exaggerated deflections.

The problem is decomposed into two subproblems: aerodynamics and structures. Other than the variable bounds, all of the constraints are global because they depend on variables from both subproblems. The lift-equals-weight constraint is uncoupled because lift is only a function of twist, and weight is only a function of thickness, so it can be resolved

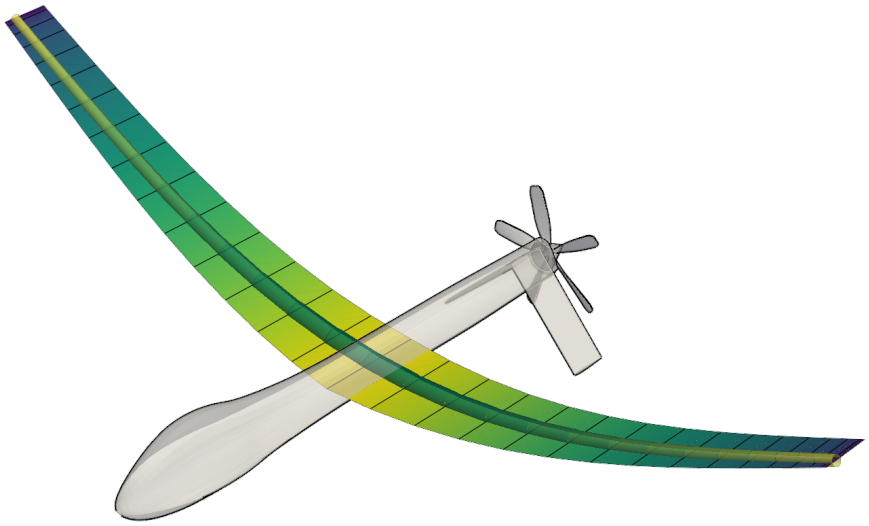

Fig. 7 The simple aero-structural test problem.

directly in the augmented Lagrangian merit function using cached values for L and W. The displacement constraint is coupled because the displacement depends on the aerodynamic loads, which depend on twist. Therefore, we reformulate the problem using a copy of the aerodynamic loads $\hat{F}$ that is added as a variable in the structural subproblem. This allows us to enforce the displacement constraint using cached values for u and a new consensus constraint $F = \hat{F}$. The augmented Lagrangian merit function for this problem is:

$$\mathscr{L}(\theta, t, \hat{F}; \lambda, \mathbf{M}) := C_D + \lambda^\top \begin{bmatrix} L - W \\ u_{\text{tip}} - u_{\max} \\ F - \hat{F} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} L - W \\ u_{\text{tip}} - u_{\max} \\ F - \hat{F} \end{bmatrix}^\top \mathbf{M} \begin{bmatrix} L - W \\ u_{\text{tip}} - u_{\max} \\ F - \hat{F} \end{bmatrix}, \tag{13}$$

where the constraints are scaled to roughly the same order of magnitude before being added to $\mathscr{L}$.

The aerodynamic subproblem is:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(\theta; t, \hat{F}, \lambda, \mathbf{M}) \\ \text{with respect to} \quad & \theta \\ \text{while solving} \quad & A\Gamma = v. \end{aligned} \tag{14}$$

The structural subproblem is:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(t, \hat{F}; \theta, \lambda, \mathbf{M}) \\ \text{with respect to} \quad & t,\ \hat{F} \\ \text{subject to} \quad & t \geq t_{\min} \\ \text{while solving} \quad & Ku = \hat{F}. \end{aligned} \tag{15}$$

The BCD inner loop that results from this decomposition is shown in Figure 8, where each subproblem is implemented with automatic derivatives using Jax [35] and solved using SciPy's SLSQP optimizer. The BCD loop minimizes $\mathscr{L}$ with respect to x given fixed values for the Lagrange multipliers and the penalty parameters. Figure 8 shows the specific intermediate quantities that are output and cached after each subproblem solution in order to maintain the strict disciplinary partitioning of the problem.

In the following results, we define the solution error as a normalized distance between the current iterate x and the known solution x^*, found by solving the monolithic problem to a tight tolerance: error $= \|(x - x^*)/x^*\|_2$. Figure 9 shows the convergence of the simple aero-structural problem using Algorithm 2. A near-exact solution is found in just over one hour on a laptop with an Intel Core Ultra 7 255H processor. For these results, a penalty parameter increment factor $\rho = 1.5$ is used to speed up convergence. Note the finite values of the penalty parameters upon convergence in

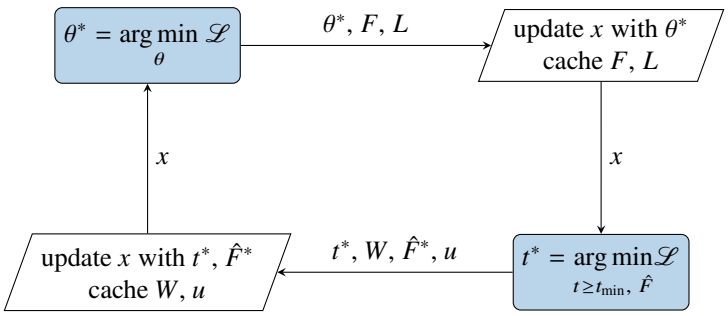

Fig. 8 The block coordinate descent loop for the simple aero-structural problem.

Figure 10, and note how some constraints are easily satisfied for small penalty parameters. The step-like behavior in the error profile occurs immediately after each outer-loop iteration when the Lagrange multipliers and the penalty parameter are updated based on the current feasibility, which often results in a sharp drop in solution error.

To demonstrate the benefits of the augmented Lagrangian method over conventional penalty methods, we also construct a version of the problem in which the global constraints are added to the objective function using quadratic exterior penalty functions with varying coefficients. These problems no longer have any global constraints, so we solve them using the basic BCD Algorithm 1. Figure 12 shows that the quadratic penalty methods are very sensitive to the values of the penalty coefficients, and, in general, converge to solutions with considerable error. For this problem, none of the quadratic penalty configurations tested can match the accuracy of ALBCD-G.

The ALBCD-G algorithm has several hyper-parameters that affect its performance. One of the most interesting is the penalty parameter increment ρ, which updates the penalty parameters in $\mathbf{M}$ for every outer-loop iteration. Figure 11 and Table 1 show the convergence of the simple aero-structural problem for different values of ρ. It is apparent that for this problem, higher values of ρ lead to faster convergence. We expect that there will be a tradeoff between solution accuracy and convergence speed, where smaller values of ρ yield better solutions at the expense of slower convergence. As ρ increases further, the penalty parameters can become unnecessarily large, which results in a feasible but sub-optimal solution.

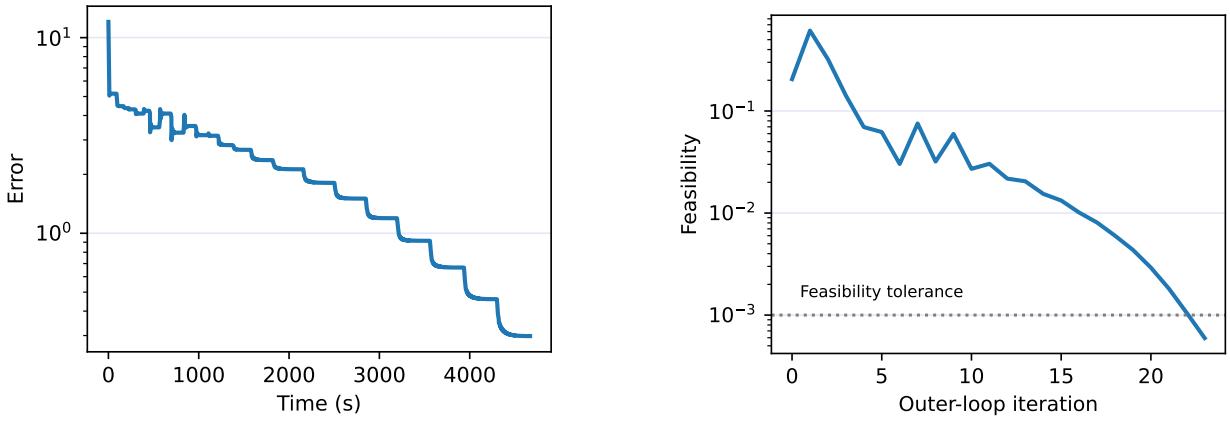

Fig. 9 The augmented Lagrangian block coordinate descent method converges to a feasible solution of the simple aero-structural problem.

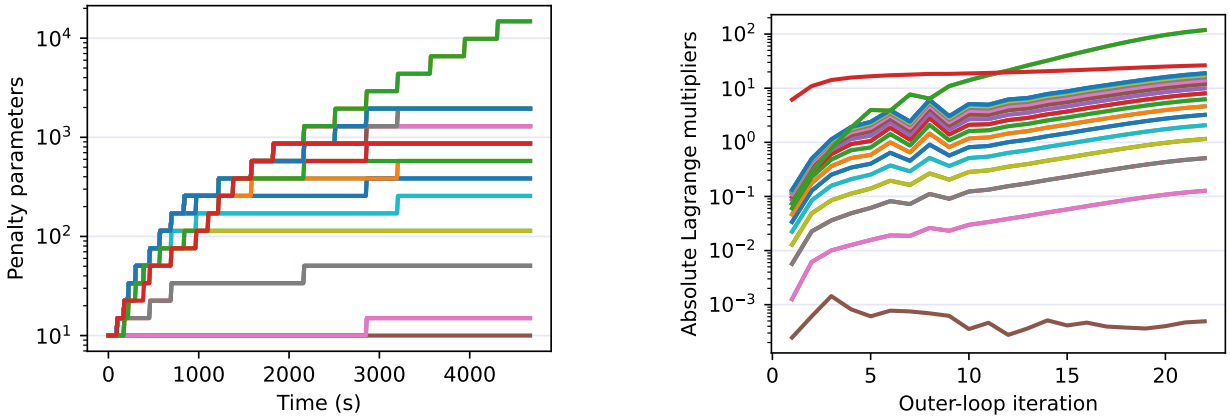

Fig. 10 Left: The penalty parameters converge to different values per scalar constraint. Right: Convergence of the Lagrange multipliers.

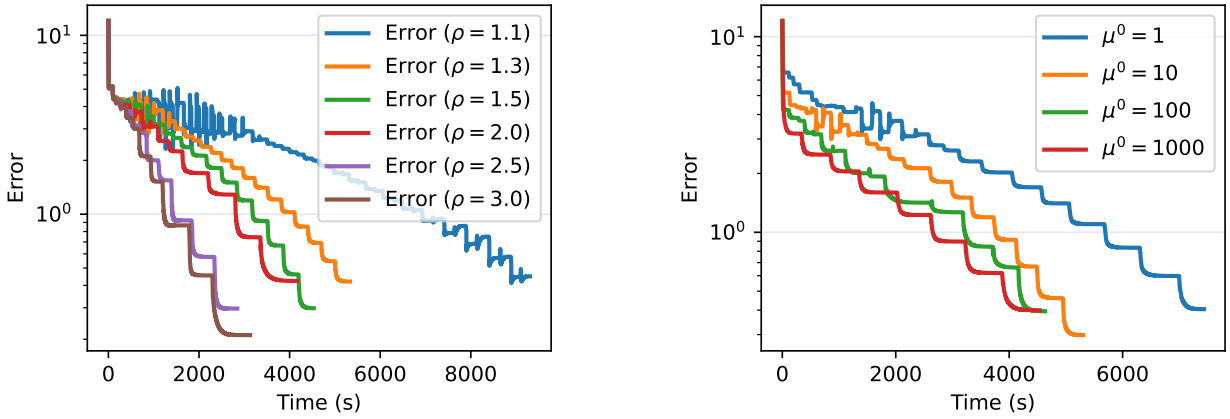

Fig. 11 Both the penalty parameter increment factor ρ and the initial value of the penalty parameter μ^0 significantly impact the speed of convergence for the simple aero-structural optimization problem.

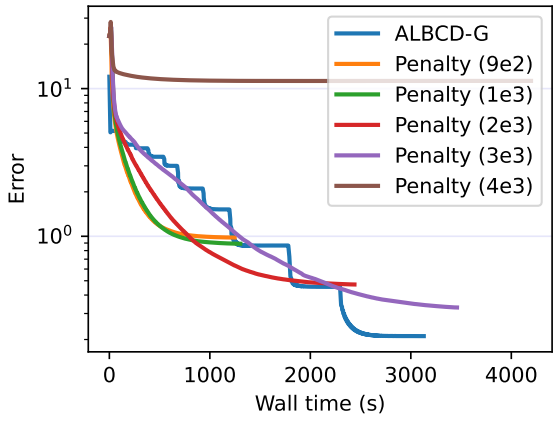

Fig. 12 Algorithm 2 converges to more accurate solutions than quadratic penalty methods.

Table 1 Time and iteration counts for various values of ρ. Inner-loop iterations are the total number of subproblem optimizations and outer-loop iterations are the total number of updates to the dual variables.

ρ	μ^0	time (s)	inner itr.	outer itr.
1.1	10	9308	1803	68
1.3	10	5334	1027	30
1.5	10	4534	929	22
2.0	10	4163	807	13
2.5	10	2843	535	11
3.0	10	3129	599	10
monolithic		27		

B. Distributed optimization of a blended-wing-body aircraft using ALBCD-G

In this example, adapted from Scotzniovsky et al. [36], we design a blended-wing-body cargo airplane to minimize fuel burn for a fixed 2000 nm mission flying at 30 kft and Mach 0.8 while carrying a 160,000 lb payload:

$$\begin{aligned} \text{minimize} \quad & \text{fuel} \\ \text{with respect to} \quad & \theta, t_{\text{web}}, t_{\text{cap}}, \theta_{\text{qc}}, b, c, \lambda_{\text{qc}}, r_p, \delta_e \\ \text{subject to} \quad & L - W = 0 \\ & M = 0 \\ & h = 0.1 \\ & \max(\sigma) = \sigma_{\text{all}} \end{aligned} \tag{16}$$

where θ is the pitch angle, t_{web} and t_{cap} are wing-box thickness distributions across the wing span, θ_{qc} is the quarter-chord twist distribution, b is the sectional span distribution, c is the sectional chord distribution, λ_{qc} is the sectional sweep, r_p is the payload position vector, δ_e is the elevator deflection. The optimization problem has two cruise trim constraints: $L = W$ and $M_y = 0$. Additionally, the static margin h is constrained to be greater than 10%, and the maximum wing-box stress must be less than the allowable stress (computed using a smooth maximum). The fuel burn objective is computed from the Breguet range equation. This problem is not reformulated using the IDF method. Instead, each subproblem evaluates the full model to compute the global constraints, primarily to ease the implementation effort. The N^2 diagram for this problem is shown in Figure 13, along with the original and optimized aircraft planforms.

The augmented Lagrangian merit function for this problem is:

$$\mathscr{L}(x; \lambda, \mu) := f + \lambda^\top c + \frac{1}{2} c^\top \mathbf{M} c, \tag{17}$$

where the equality constraints c are:

$$c = \begin{bmatrix} L - W \\ M \\ h - 0.1 \\ \max(\sigma) - \sigma_{\text{all}} \end{bmatrix}. \tag{18}$$

The distributed version of this optimization problem contains two subproblems. The first subproblem includes all of the geometric and planform variables:

$$\begin{aligned} \text{minimize} \quad & \mathscr{L}(x; \lambda, \mu) \\ \text{with respect to} \quad & \theta, \theta_{\text{qc}}, b, c, \lambda_{\text{qc}}, \delta_e, \end{aligned} \tag{19}$$

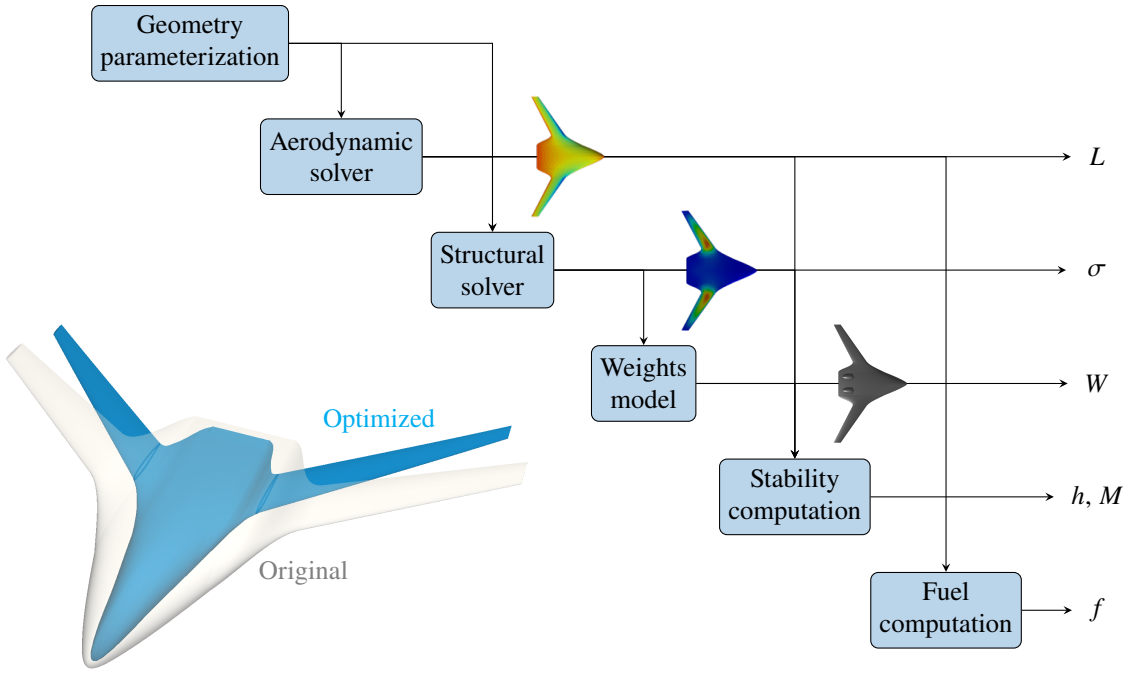

Fig. 13 The N^2 diagram for the BWB optimization problem.

and the second subproblem includes the structural and payload layout variables:

$$\begin{aligned} &\text{minimize} && \mathscr{L}(x;\lambda,\mu) \\ &\text{with respect to} && t_{\text{web}}, t_{\text{cap}}, r_p, \end{aligned} \tag{20}$$

where both subproblems have local constraints consisting of upper and lower variable bounds. The subproblems are implemented in CSDL and solved using the PySLSQP optimizer from modOpt [37].

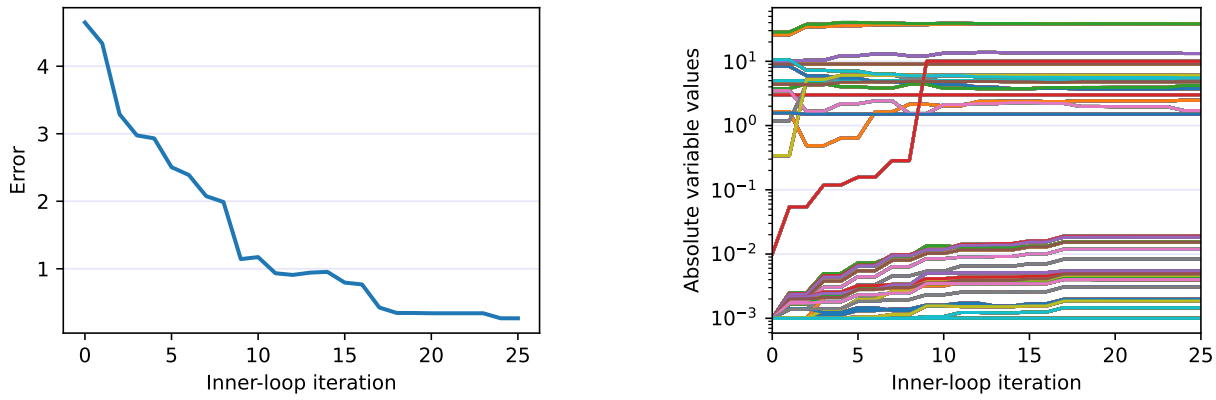

Fig. 14 Left: Solution error versus iteration for the BWB problem. Right: Convergence of the BWB variables.

Algorithm 2 solves this problem to a reasonable error, as shown in Figure 14. Each subproblem is expensive to solve, and the results shown here were generated in approximately 75 hours. Tighter convergence is expected if the algorithm was allowed to run longer. Despite this, convergence to a consistent solution can still be seen by examining the histories of the design variables. Convergence with respect to iterations is actually faster for the BWB problem than for the simple aero-structural problem because we do not consider any problem reformulations. Instead, each

subproblem evaluates the entire model to compute the global constraints. This results in far fewer global constraints, and the augmented Lagrangian method has a relatively easy time satisfying them.

V. Conclusion

In this paper, we have presented a distributed MDO formulation based on an augmented Lagrangian relaxation of global constraints. The formulation is solved using a block coordinate descent algorithm, which iterates in an inner loop to minimize the augmented Lagrangian merit function while an outer loop updates the Lagrange multipliers and penalty parameters. Together, this double-loop structure is guaranteed to converge to stationary points of the original MDO problem, provided the inner loop is solved to sufficient tolerance and the problem satisfies some basic assumptions. This convergence guarantee is a significant advantage over many existing distributed MDO architectures, which lack such theoretical guarantees.

A key distinction of the proposed method is that exact global constraint satisfaction is achieved without requiring large penalty coefficients. In conventional penalty-based approaches, the penalty parameters must be driven to large values to adequately enforce constraints, which introduces severe ill-conditioning and typically precludes reliable convergence on anything beyond the simplest problems. The augmented Lagrangian approach avoids this issue, and the penalty parameter remains finite at convergence. Additionally, the proposed formulation offers practical advantages that make it well-suited for real-world engineering problems; by decomposing the problem by discipline, organizations and teams can retain ownership of their individual models, addressing privacy concerns and the challenges that arise when disciplines rely on proprietary or legacy codes.

In both of our examples, the distributed algorithm converges much more slowly than monolithic optimization. This is not ideal, but it is also expected, because the optimization trajectories are limited by the orthogonality of the subspaces for each subproblem. However, recall that this method targets applications where there is no option other than distributed optimization, and when viewed from this perspective, the optimization time appears more acceptable. Regardless, ALBCD-G is expected to be faster and more accurate than many of the existing distributed MDO methods that depend solely on penalty functions to resolve coupling and global constraints.

It is time-consuming and difficult to take a monolithic optimization problem and reformulate it to be solved using ALBCD-G. Future work will focus on alleviating this burden through automated strategies for decomposing MDO problems. Likewise, a suitable framework would increase the probability of a successful implementation. In particular, we plan to demonstrate a framework where ALBCD-G is implemented over a network, enabling geographically distributed implementations using Philote [38]. This would naturally support multi-fidelity approaches through continuation [39, 40], where simpler models are used in early BCD iterations, followed by higher-fidelity models later.

The primary alternative for our proposed method is monolithic optimization, where remote disciplinary models are queried by a central process [38], an approach that is functionally monolithic but computationally distributed. This partially preserves the privacy of each discipline's models, but not entirely, since it may be possible to reconstruct a model from a sufficiently large collection of remote queries. It is also possible that bandwidth limitations will be a bottleneck, since every function and gradient evaluation would require remote function calls over a network [41]. The proposed ALBCD-G formulation avoids both of these issues. By solving subproblems locally and exchanging only a small set of cached intermediate variables between disciplines, it more rigorously protects proprietary models and substantially reduces the volume of data that must be transmitted, making it better suited to settings where privacy and communication overhead are constraints.

Acknowledgments

The authors would like to acknowledge the Multidisciplinary Science and Technology Center of the Air Warfare Directorate, Air Force Research Laboratory for funding and supporting this effort through the Collaborative Center for Design and Research of Interdisciplinary Systems program. The authors acknowledge Gregory Reich for his valuable research guidance, and thank Luca Scotzniovsky and Andrew Fletcher for their assistance with the blended-wing-body optimization problem.

References

[1] Lupp, C. A., Deaton, J. D., Kao, J. Y., Clark, D. L., Smith, H. A., and Xu, A., "The Monolith Must Die: Why the Correct Partitioning of MDO Problems is Necessary to Enable Large-Scale Applications," *AIAA SCITECH Forum*, 2025, p. 0579. https://doi.org/10.2514/6.2025-0579.

[2] Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary aerospace design optimization: survey of recent developments," *Structural optimization*, Vol. 14, No. 1, 1997, pp. 1–23. https://doi.org/10.1007/BF01197554.

[3] Papalambros, P. Y., "The optimization paradigm in engineering design: promises and challenges," *Computer-Aided Design*, Vol. 34, No. 12, 2002, pp. 939–951. https://doi.org/10.1016/S0010-4485(01)00148-8.

[4] Ashley, H., "On making things the best-aeronautical uses of optimization," *Journal of Aircraft*, Vol. 19, No. 1, 1982, pp. 5–28. https://doi.org/10.2514/3.57350.

[5] Kroo, I. M., "MDO for Large-Scale Design," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, Philadelphia, PA, 1997, pp. 22–44.

[6] Martins, J. R., and Ning, A., *Engineering design optimization*, Cambridge University Press, 2021.

[7] Alexandrov, N., and Lewis, R., "Analytical and computational properties of distributed approaches to MDO," *8th Symposium on Multidisciplinary Analysis and Optimization*, 2000, p. 4718. https://doi.org/10.2514/6.2000-4718.

[8] Cramer, E. J., Dennis, J. E., Jr., 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. https://doi.org/10.1137/0804044.

[9] Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary optimization methods for aircraft preliminary design," *5th symposium on multidisciplinary analysis and optimization*, 1994, p. 4325. https://doi.org/10.2514/6.1994-4325.

[10] Balling, R. J., and Sobieszczanski-Sobieski, J., "Optimization of coupled systems-a critical overview of approaches," *AIAA journal*, Vol. 34, No. 1, 1996, pp. 6–17. https://doi.org/10.2514/3.13015.

[11] Braun, R. D., *Collaborative optimization: an architecture for large-scale distributed design*, Stanford University, 1996.

[12] Wujek, B. A., Renaud, J. E., Batill, S. M., and Brockman, J. B., "Concurrent subspace optimization using design variable sharing in a distributed computing environment," *Concurrent Engineering*, Vol. 4, No. 4, 1996, pp. 361–377. https://doi.org/10.1177/1063293X960040040.

[13] Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jiang, T., "Target Cascading in Optimal System Design," *Journal of Mechanical Design*, Vol. 125, No. 3, 2003, pp. 474–480. https://doi.org/10.1115/1.1582501.

[14] Perez, R., Liu, H., and Behdinan, K., "Evaluation of multidisciplinary optimization approaches for aircraft conceptual design," *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2004, p. 4537. https://doi.org/10.2514/6.2004-4537.

[15] Roth, B. D., and Kroo, I. M., "Enhanced collaborative optimization: a decomposition-based method for multidisciplinary design," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 43253, 2008, pp. 927–936. https://doi.org/10.1115/DETC2008-50038.

[16] Yi, S.-I., Shin, J.-K., and Park, G., "Comparison of MDO methods with mathematical examples," *Structural and Multidisciplinary Optimization*, Vol. 35, No. 5, 2008, pp. 391–402. https://doi.org/10.1007/s00158-007-0150-2.

[17] Birgin, E. G., Fernández, D., and Martínez, J. M., "The boundedness of penalty parameters in an augmented Lagrangian method with constrained subproblems," *Optimization Methods and Software*, Vol. 27, No. 6, 2012, pp. 1001–1024. https://doi.org/10.1080/10556788.2011.556634.

[18] Conn, A. R., Gould, N. I., and Toint, P., "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Numerical Analysis*, Vol. 28, No. 2, 1991, pp. 545–572. https://doi.org/10.1137/0728030.

[19] Orndorff, N. C., and Hwang, J. T., "A Distributed Method for Solving Large-Scale Multidisciplinary Optimization Problems," *AIAA AVIATION FORUM AND ASCEND*, 2025, p. 3736. https://doi.org/10.2514/6.2025-3736.

[20] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J., "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, Vol. 3, No. 1, 2011, p. 1–122. https://doi.org/10.1561/2200000016.

[21] Shorinwa, O., Halsted, T., Yu, J., and Schwager, M., "Distributed Optimization Methods for Multi-Robot Systems: Part I – A Tutorial," , 2024. https://doi.org/10.48550/arXiv.2301.11313.

[22] Hong, M., Razaviyayn, M., Luo, Z.-Q., and Pang, J.-S., "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing," *IEEE Signal Processing Magazine*, Vol. 33, No. 1, 2015, pp. 57–77. https://doi.org/10.1109/MSP.2015.2481563.

[23] Dwork, C., and Roth, A., "The algorithmic foundations of differential privacy," *Foundations and trends in theoretical computer science*, Vol. 9, No. 3-4, 2014, pp. 211–487. https://doi.org/10.1561/0400000042.

[24] Zhu, L., Liu, Z., and Han, S., "Deep leakage from gradients," *Advances in neural information processing systems*, Vol. 32, 2019. https://doi.org/10.5555/3454287.3455610.

[25] Huang, Z., Mitra, S., and Vaidya, N., "Differentially private distributed optimization," *Proceedings of the 16th international conference on distributed computing and networking*, 2015, pp. 1–10. https://doi.org/10.1145/2684464.2684480.

[26] Nocedal, J., and Wright, S. J., *Numerical optimization*, Springer, 1999.

[27] Byrd, R. H., Nocedal, J., and Waltz, R. A., *Knitro: An Integrated Package for Nonlinear Optimization*, Springer US, Boston, MA, 2006, pp. 35–59. https://doi.org/10.1007/0-387-30065-1_4.

[28] Byrd, R. H., Nocedal, J., and Waltz, R. A., "Steering exact penalty methods for nonlinear programming," *Optimization Methods and Software*, Vol. 23, No. 2, 2008, pp. 197–213. https://doi.org/10.1080/10556780701394169.

[29] Hestenes, M. R., "Multiplier and gradient methods," *Journal of optimization theory and applications*, Vol. 4, No. 5, 1969, pp. 303–320. https://doi.org/10.1007/BF00927673.

[30] Birgin, E. G., and Martínez, J. M., *Practical augmented Lagrangian methods for constrained optimization*, SIAM, 2014.

[31] Morelli, A. C., Hofmann, C., and Topputo, F., "Warm start of interior-point methods applied to sequential convex programming," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 60, No. 4, 2024, pp. 3837–3846. https://doi.org/10.1109/TAES.2024.3371405.

[32] Rockafellar, R. T., "The multiplier method of Hestenes and Powell applied to convex programming," *Journal of Optimization Theory and applications*, Vol. 12, No. 6, 1973, pp. 555–562. https://doi.org/10.1007/BF00934777.

[33] Birgin, E. G., Castillo, R. A., and Martínez, J. M., "Numerical comparison of augmented Lagrangian algorithms for nonconvex problems," *Computational Optimization and Applications*, Vol. 31, No. 1, 2005, pp. 31–55. https://doi.org/10.1007/s10589-005-1066-7.

[34] Martins, J. R., and Lambe, A. B., "Multidisciplinary design optimization: a survey of architectures," *AIAA journal*, Vol. 51, No. 9, 2013, pp. 2049–2075. https://doi.org/10.2514/1.J051895.

[35] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., "JAX: composable transformations of Python+NumPy programs," , 2018. URL http://github.com/jax-ml/jax.

[36] Scotzniovsky, L., Orndorff, N. C., Fletcher, A., Kao, J., and Hwang, J. T., "Demonstration of a Large-Scale Multidisciplinary Design Optimization Methodology on a Blended Wing Body Configuration using Mid-Fidelity, Multi-Point Analysis," *AIAA AVIATION FORUM*, 2026.

[37] Joshy, A. J., and Hwang, J. T., "modopt: A modular development environment and library for optimization algorithms," *Advances in Engineering Software*, Vol. 213, 2026, p. 104084. https://doi.org/10.1016/j.advengsoft.2025.104084.

[38] Lupp, C. A., and Xu, A., "Creating a Universal Communication Standard to Enable Heterogeneous Multidisciplinary Design Optimization," *AIAA SCITECH Forum*, 2024, p. 1799. https://doi.org/10.2514/6.2024-1799.

[39] Bryson, D. E., and Rumpfkeil, M. P., "Multifidelity quasi-newton method for design optimization," *AIAA Journal*, Vol. 56, No. 10, 2018, pp. 4074–4086. https://doi.org/10.2514/1.J056840.

[40] Beran, P. S., Bryson, D., Thelen, A. S., Diez, M., and Serani, A., "Comparison of multi-fidelity approaches for military vehicle design," *AIAA Aviation Forum*, 2020, p. 3158. https://doi.org/10.2514/6.2020-3158.

[41] Berahas, A. S., Bollapragada, R., Keskar, N. S., and Wei, E., "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, Vol. 64, No. 8, 2018, pp. 3141–3155. https://doi.org/10.1109/TAC.2018.2880407.