Author version

Published as:

Orndorff, N. C., & Hwang, J. T. (2025). A distributed method for solving large-scale multidisciplinary optimization problems. In AIAA AVIATION FORUM AND ASCEND 2025 (Paper No. AIAA 2025-3736). https://doi.org/10.2514/6.2025-3736

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/orndorff2025distributed/

```
@inproceedings{orndorff2025distributed,
  author = {Nicholas C. Orndorff and John T. Hwang},
  title = {A Distributed Method for Solving Large-Scale Multidisciplinary Optimization
    Problems},
  booktitle = {AIAA AVIATION FORUM AND ASCEND 2025},
  year = {2025},
  doi = {10.2514/6.2025-3736},
  url = {https://doi.org/10.2514/6.2025-3736}
}
```

A Distributed Method for Solving Large-Scale Multidisciplinary Optimization Problems

Nicholas C. Orndorff* and John T. Hwang.†
University of California San Diego, La Jolla, CA, 92092

In this paper, we propose an iterative and distributed method for solving multidisciplinary design optimization problems using a block-coordinate-descent algorithm. The method is motivated by scenarios in which the monolithic formulation of the optimization problem either fails to reliably converge, or cannot be used at all due to implementation challenges. In these situations, we decompose the monolithic problem into multiple optimization sub-problems, each smaller and easier to solve than the original. Our work has three major contributions. First, we introduce a novel distributed architecture for multidisciplinary optimization problems that is solved using block coordinate descent. Second, we present a theoretical result that guarantees global convergence to a stationary point given an unconstrained optimization problem. Third, we demonstrate this theoretical result by solving distributed versions of a control co-design problem and two aero-structural optimization problems. In each of these examples we recover the monolithic solution, and, for the aero-structural problems, we show that distributed warm-starting improves the reliability of convergence from several random initializations.

I. Introduction

Multidisciplinary design optimization (MDO) problems can be solved with a number of alternative formulations. Conceptually, the simplest (and the most widely used) is the monolithic formulation, in which a single optimization is performed to minimize the objective function with respect to all design variables. However, the monolithic formulation cannot always be used because of reasons such as:

- its memory requirements are too demanding,
- its poor numerical conditioning prevents reliable convergence,
- the problem incorporates protected models from multiple institutions,
- it must be solved in a geographically distributed manner, severely degrading monolithic performance.

In these circumstances, it makes sense to explore alternative formulations that avoid these issues.

Because of the problem's multidisciplinary structure, it is natural to consider distributed formulations, in which the monolithic problem is divided into several smaller and simpler optimization problems that are solved iteratively. However, it is often unclear whether a distributed solution is also a solution for the original monolithic formulation. Moreover, solving distributed formulations is not necessarily straightforward.

Many distributed MDO architectures and algorithms already exist [1]. Nearly all of these are hierarchical. In other words, there is at least one system-level optimization problem that coordinates multiple sub-problems. This includes collaborative optimization [2], concurrent subspace optimization [3, 4], and analytical target cascading [5]. One issue with this approach is that sub-problems can jump between different local minima, causing discontinuities in the system-level problem [6–8]. For this reason, among others, many existing distributed MDO architectures have no theoretical guarantees of convergence (at least without restrictive assumptions). Also, the hierarchical formulation requires a centralized algorithm structure which may not be possible or practical depending on the situation. This last issue is often present in industrial MDO problems, where distributed formulations are required in order to create optimization problems that can be solved using computing hardware from multiple organizations [9]. Another factor of increasing interest is the privacy of models used during optimization. Distributed architectures inherently have the potential to maintain privacy to a much higher degree than any monolithic architectures. In this paper, we attempt to address some of these issues with a fully-decentralized and distributed formulation that is solved with a block-coordinate-descent (BCD) algorithm.

*PhD Student, Department of Mechanical and Aerospace Engineering, AIAA Student Member.
†Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA Member.

Block coordinate descent is an established method with many variations. In this paper, we consider cyclic block coordinate descent with exact minimization. This means that each block is minimized in a cyclic order within each coordinate descent iteration while all remaining variables are fixed. Other names for block coordinate descent include nonlinear block Gauss-Seidel [10, 11], alternating minimization, and best response algorithms. These other methods are similar in structure to coordinate descent algorithms, although they do not always use exact minimization for each block update. In the block coordinate descent algorithm, each coordinate-wise minimization is smaller and typically computationally cheaper to solve than the original monolithic optimization problem. So, it follows logically that convergence can often be achieved more reliably. And, if each sub-problem is sufficiently smaller in dimension and complexity, block coordinate descent may be a competitive strategy when compared with monolithic optimization algorithms. (In that case, model and gradient evaluations might be faster and less computationally expensive, especially when time, memory, or both scale worse than linearly with dimension.) In many ways, these advantages are analogous to the rationale for using stochastic/mini-batch gradient descent for very-large-scale machine-learning problems, and the advantages we hope to achieve are the same: faster and more reliable iterations at the cost of a slower convergence rate. These and other advantages of coordinate descent methods have been demonstrated for training support vector machines [12], LASSO regressions [13], and convex optimization [14, 15]. Naturally, this is expected to suit very-large-scale MDO problems with many expensive disciplines. One example of this would be a multi-point aircraft optimization (e.g., [16]), where there are several repeated copies of the same physics-based solvers (e.g., aerodynamics or structures) that are instantiated at different operating conditions.

Like other distributed MDO architectures, guaranteeing the convergence of coordinate descent methods to local minima of the original optimization problem is not straightforward. A well-known proof by Bertsekas [17] demonstrates convergence to stationary points if the feasible set is convex and the minimum of each block is uniquely attained at every iteration, assumptions that hold in general only for strictly convex functions. Grippo and Sciandrone [15] extended this analysis to general nonconvex functions on convex feasible sets if there are only two blocks. These and other existing convergence guarantees are overly restrictive for most MDO problems where coordinate-wise global minimums can rarely be guaranteed and constraints are often non-convex. In this paper, we prove global convergence to stationary points of unconstrained non-convex problems, under the primary assumption that the gradient of the objective function is Lipschitz continuous. This is a very general assumption that is used in many gradient descent and line-search convergence proofs, and is applicable to most continuously differentiable MDO problems.

Block coordinate descent methods have been successfully demonstrated on non-convex problems with non-convex constraints [18, 19]. However, because the assumptions required to guarantee convergence are almost never met for MDO problems, achieving convergence can be a trial-and-error process. One obvious solution is to convert all constraints to penalty terms in the objective function. This is a viable but not ideal solution because the penalty functions create ill-conditioned problems with approximate solutions. Given that, we suggest an augmented Lagrangian modification of the block coordinate descent algorithm that allows for convergence guarantees in the presence of equality constraints with better conditioning.

Sometimes, a well-posed monolithic optimization problem will fail to reliably converge. For these problems the reliability of convergence can often be improved by finding a new initialization point from which the problem has improved numerical properties. Therefore, we propose a continuation warm-starting scheme, which begins by iteratively solving the distributed formulation using block coordinate descent, and ends by solving the monolithic formulation using the distributed solution as a warm-start initialization. The intent is to use the smaller problems to reach a point from which the monolithic optimization problem can converge. For this to work, every sub-problem must have better (i.e., more reliable) convergence properties than the original monolithic problem. Warm starting with coordinate descent is rarely expected to converge to a solution faster than the monolithic formulation (if the monolithic problem converges). Instead, it is most applicable to monolithic problems that are extremely sensitive to initial guesses for variable values. In this case, the monolithic problem might fail to converge at all, so the warm-start approach, if successful, is an improvement even if it is slow.

Distributed architectures are not widely used for solving MDO problems. Reasons for this include the difficulty of their implementation, their lack of convergence guarantees, and their poor convergence rate. However, as MDO problems become larger and more complicated, distributed optimization can fill an important niche. We foresee that distributed formulations like the one proposed in this paper will have better scalability, conditioning, and modularization than existing monolithic architectures. With that in mind, the main contributions of this paper include:

1) a fully decentralized and distributed architecture for MDO problems,
2) global convergence guarantees for unconstrained problems,
3) application to a control co-design problem and two aero-structural problems.

Section II presents important notation and definitions that are used throughout this paper. Distributed formulations and our block coordinate descent algorithm are presented and analyzed in Section III, and demonstrated on several MDO problems in Section V. Our conclusion VI summarizes our work and results.

II. Preliminaries

In the following optimization problem, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a continuously differentiable function that is minimized with respect to the variables $x \in \mathbb{R}^n$, subject to the inequality constraints $g : \mathbb{R}^n \mapsto \mathbb{R}^m$, and equality constraints $h : \mathbb{R}^n \mapsto \mathbb{R}^p$:

$$\begin{aligned} x^* = \underset{x \in \mathbb{R}^n}{\arg\min} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{1}$$

Specifically, we consider MDO problems where there exists some subdivision of x into $N > 1$ disciplines. In this case $x_i \in \mathbb{R}^{n_i}$ represents the i-th discipline's variables and $n = n_1 + ... + n_N$. Likewise, $g(x)$ and $h(x)$ are vectors containing all the constraints for every discipline: $g(x) = (g_1(x), ..., g_N(x))$ and $h(x) = (h_1(x), ..., h_N(x))$, where $g_i \in \mathbb{R}^{m_i}$ and $h_i \in \mathbb{R}^{p_i}$. If the feasible set for each discipline is denoted by $\mathcal{X}_i(x) = \{x_i \in \mathbb{R}^{n_i} |\ g_i(x) \leq 0,\ h_i(x) = 0\}$, then the feasible set for x is: $\mathcal{X} = \{x \in \mathbb{R}^n |\ g(x) \leq 0,\ h(x) = 0\}$.

The monolithic optimization problem 1 is considered to be *well-posed* if at least one solution exists. For most MDO problems, we know this is the case from engineering intuition, and when in doubt, Wierstrass' theorem is generally applicable to MDO problems and can be used to guarantee the existence of minima (e.g., Prop. A.8 [17]). Given a well-posed problem, a *perfect optimizer* will always return a solution. A (local) solution is a point x^* that satisfies the Karush–Kuhn–Tucker (KKT) conditions. Unless stated otherwise, we assume that the monolithic optimization problem is always well posed and that every optimization problem is solved with a perfect optimizer.

A constraint is considered to be *global* if, for a given decomposition, it is a function of some x_i and some x_j ($j \neq i$). Conversely, a *local* constraint depends only on x_i. This notation does not require that the constraints for any discipline are local constraints. The notation in Equation 1 and the following paragraph is simply one of many possible ways of rewriting the standard MDO problem in a form that is more amenable to decomposition.

III. A decentralized and distributed architecture

A. A distributed formulation for MDO problems

When constructing a distributed version of a monolithic MDO problem, the variables must be partitioned and assigned to each sub-problem. We propose the following definition for this partitioning problem. Given the variables $x \in \mathbb{R}^n$, we wish to decompose x into N sub-vectors $x_i \in \mathbb{R}^{n_i}$, ($i = 1, ..., N$, $1 \leq N \leq n$), henceforth called the *i-th block* of x, with the following properties:

1) $n = \sum_{i=1}^{N} n_i$,
2) $x_i = \mathrm{U}_i^\top x$, for the binary selection matrix $\mathrm{U}_i \in \mathbb{R}^{n \times n_i}$,
3) $\nabla_i f(x) = \mathrm{U}_i^\top \nabla f(x)$ (*i-th block-gradient*),
4) $\mathrm{U}_i^\top \mathrm{U}_j = 0$ for all $i \neq j$ (*non-overlapping partitions*).

The matrices U_i are column permutations of an $n \times n$ identity matrix, which allows each block to be constructed from non-consecutive components in x, and, is therefore a more generally applicable notation than can be found in other literature where the partitioning of x is expected to be contiguous. Conditions 1 and 4 ensure full coverage of x. In other words, the matrix $\mathrm{U} = [\mathrm{U}_1, ..., \mathrm{U}_N]_{n \times n}$ is full rank. If the decomposition of x into $x_1, ..., x_N$ satisfies all these conditions, we call it a *complete decomposition* of x. Note that, if $N = 1$ then $\mathrm{U} = \mathrm{U}_1 = \mathrm{I}_n$, and we recover the original monolithic optimization problem. In Section II, N is defined as the number of disciplines for convenience. However, these disciplines/sub-problems can be defined in any way that makes sense in order to achieve an appropriate decomposition. Sometimes, the partitioning of x will be pre-determined by discipline boundaries or organizational requirements. In other circumstances we can choose partitions that give favorable convergence properties or result in minimal computational expense.

Consider the following distributed version of the monolithic optimization problem (1):

$$
\begin{aligned}
x_1^* = \underset{x_1 \in \mathbb{R}^{n_1}}{\arg\min} \quad & f(x_1, x_2^*, ..., x_N^*) \qquad \ldots \quad x_N^* = \underset{x_N \in \mathbb{R}^{n_N}}{\arg\min} \quad f(x_1^*, ..., x_{N-1}^*, x_N) \\
\text{subject to} \quad & g(x_1, x_2^*, ..., x_N^*) \leq 0 \qquad\qquad \text{subject to} \quad g(x_1^*, ..., x_{N-1}^*, x_N) \leq 0 \\
& h(x_1, x_2^*, ..., x_N^*) = 0 \qquad\qquad\qquad\quad h(x_1^*, ..., x_{N-1}^*, x_N) = 0.
\end{aligned} \tag{2}
$$

Each of the N optimization problems is smaller than the monolithic problem (1), and is typically easier to solve. However, it is not immediately obvious that a solution to this distributed formulation exists, and, if it does exist, whether it is also a solution to the monolithic formulation. If the objective function and all the constraints are separable with respect to each x_i then a single solution to the N sub-problems is also a solution to the monolithic formulation. However, most MDO problems are formulated to discover solutions that reflect non-obvious interdisciplinary interactions, so it is unlikely that any meaningful multidisciplinary optimization problem is separable.

An immediate issue with this distributed formulation is that if the feasible set $\mathcal{X}_i$ depends on some x_j ($j \neq i$), there is a risk that no feasible solution exists for the i-th block ($\mathcal{X}_i = \emptyset$). Therefore, we only consider problems with local feasible sets. We achieve this by converting any global constraints to penalty terms in the objective function. The i-th problem with only local constraints is:

$$
x_i^* = \underset{x_i \in \mathcal{X}_i}{\arg\min} \quad f(x), \quad i = 1, 2..., N, \tag{3}
$$

which we can solve iteratively using the following block coordinate descent algorithm.

B. Block coordinate descent algorithm

The MDO problems that we consider in this paper are not separable, which means that an iterative algorithm is required to find a solution that satisfies the optimality conditions for the distributed formulation. We propose to use cyclic block coordinate descent to iteratively solve each of the N distributed optimization problems using the most up-to-date values of x. Because no computations are performed at the system level, Algorithm 1 can be fully decentralized. This means that each optimization problem can be solved using dedicated algorithms and/or on heterogeneous computer systems where the only requirement is that the most recent values of x be sent to the subsequent optimization problem(s).

Algorithm 1 Cyclic block coordinate descent

1: Initialize $x^{(0)} \in \mathbb{R}^n$
2: $k \leftarrow 0$
3: **while** not converged **do**
4: **for** $i = 1, 2, ..., N$ **do**
5: $x_i^* = \underset{x_i \in \mathcal{X}_i}{\arg\min} f(x)$ with initial values $x^{(k)}$
6: update the i-th block of $x^{(k)}$ with x_i^*
7: **end for**
8: $k \leftarrow k + 1$
9: $x^{(k)} \leftarrow x^{(k-1)}$
10: **end while**

The most up-to-date value of x in the block coordinate descent algorithm is represented as $z_i^{(k)}$:

$$
\begin{aligned}
& z_0^{(k)} = x^{(k)}, \\
& \vdots \\
& z_i^{(k)} = \mathrm{U}_1 x_1^{(k+1)} + ... + \mathrm{U}_i x_i^{(k+1)} + ... + \mathrm{U}_N x_N^{(k)}), \quad i = 1, 2, ..., N-1, \\
& \vdots \\
& z_N^{(k)} = x^{(k+1)},
\end{aligned} \tag{4}
$$

where $z_{N+1}^{(k)} = z_1^{(k+1)}$. And, $x_i^{(k+1)} \in \mathcal{X}_i$ is the (local) solution in the i-th subspace during the k-th Gauss-Seidel cycle. The sequence $\{x^{(k)}\}_{k=0}^{\infty}$ generated by coordinate descent is defined by the points $x^{(k)} := \sum \mathrm{U}_i x_i^{(k)} \in \mathcal{X}$. Each

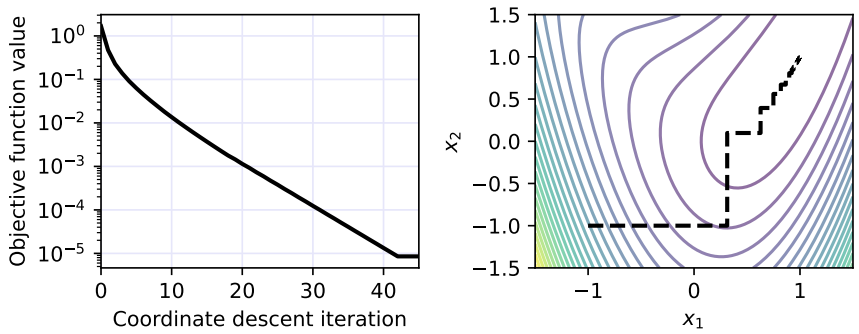

Fig. 1 Block coordinate descent (Algorithm 1) applied to the Rosenbrock function.

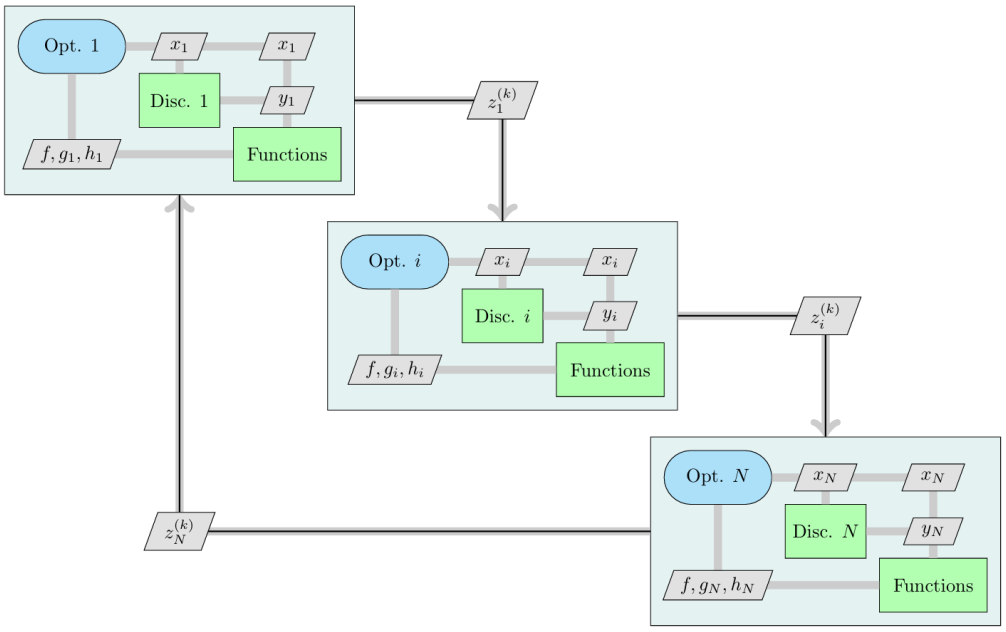

Fig. 2 The structure of the block coordinate descent algorithm viewed as an XDSM.

coordinate-wise minimization can be thought of as a mapping from $x_i^{(k)}$ to $x_i^{(k+1)}$ using the update vector $p_i^{(k)}$ resulting from the exact minimization:

$$x_i^{(k+1)} = x_i^{(k)} + p_i^{(k)}, \quad (i\text{-th coordinate update}). \tag{5}$$

Or, using the up-to-date vector z within one coordinate descent iteration:

$$z_{i+1}^{(k)} = z_i^{(k)} + \mathrm{U}_i p_i^{(k)}, \quad (\text{full vector update}). \tag{6}$$

This notation helps us to define sequences of iterates generated by Algorithm 1 and to prove convergence of the block coordinate descent algorithm in the following section. Figure 1 demonstrates Algorithm 1 on the Rosenbrock function.

IV. Guarantee of global convergence

For an optimization problem without constraints ($\mathcal{X}_i = \mathbb{R}^{n_i}$), it is immediately apparent that the sequence of iterates:

$$f(x^{(k)}) \geq f(z_1^{(k)}) \geq \ldots \geq f(z_{N-1}^{(k)}) \geq f(x^{(k+1)}), \tag{7}$$

is monotonically decreasing by the definition of the coordinate descent algorithm. And, if f is bounded below then the sequence $\{f(x^{(k)})\}_{k=0}^{\infty}$ will converge (using the monotone convergence theorem). However, it is not clear that the

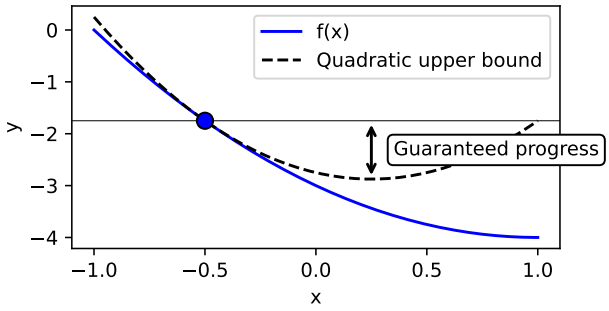

Fig. 3 The Lipschitz continuous gradient results in a quadratic upper bound on the objective function.

coordinate descent algorithm will converge to a stationary point where $\nabla_i f(x) = 0$ simultaneously for all blocks. To show this, we introduce some new assumptions.

We assume that the gradient of f is block Lipschitz continuous with respect to the block Lipschitz constant L_i if $\nabla_i f$ satisfies:

$$||\nabla_i f(x + \mathrm{U}_i p_i) - \nabla_i f(x)|| \leq L_i ||p_i||, \quad \text{for every } p_i \in \mathbb{R}^{n_i}, x \in \mathbb{R}^n. \tag{8}$$

In other words, there is an upper bound on the singular values of the Hessian: $\nabla^2_{ii} f \preceq L_i I$. This assumption implies that the gradient of f cannot change arbitrarily fast. If the gradient of f is Lipschitz continuous with Lipschitz constant L, then it follows that the gradient of f is also *block* Lipschitz continuous with $L_i \leq L$ for all $i = 1, ..., N$ because $\mathbb{R}^{n_i} \subset \mathbb{R}^n$. The block Lipschitz gradient condition implies a block form of the standard descent lemma (see e.g., Bertsekas Prop. A.24 [17] for a derivation).

(*Block Descent Lemma [20]*). Suppose that f is a continuously differentiable function over $\mathbb{R}^n$ that satisfies the block Lipschitz gradient condition and assume that $i \in \{1, 2, ..., N\}$. Let $u, v \in \mathbb{R}^n$ be two vectors that differ in only the i-th block, that is, there exists an $h \in \mathbb{R}^n$ such that $v - u = \mathrm{U}_i h$. Then,

$$f(v) \leq f(u) + \langle \nabla_i f(u), h \rangle + \frac{L_i}{2} ||h||^2. \tag{9}$$

The descent lemma provides a quadratic upper bound for f on the i-th block at any given point u (Fig. 3). We use the block descent lemma to derive a lower bound on guaranteed progress for the block coordinate descent algorithm when applied to unconstrained optimization problems.

Theorem IV.1. *(Global convergence for unconstrained block coordinate descent). Suppose that f is continuously differentiable and bounded below in $\mathbb{R}^n$. And suppose that the gradient of f is block Lipschitz continuous. Let $\{z_i^{(k)}\}_{k=0}^{\infty}$ be a sequence generated by the block coordinate descent algorithm. Then, every limit point of $\{z_i^{(k)}\}$ is a stationary point of f.*

Proof. The block Lipschitz continuous gradient implies the descent lemma:

$$f(z_{i+1}^{(k)}) \leq f(z_i^{(k)}) + \langle \nabla_i f(z_i^{(k)}), p_i^{(k)} \rangle + \frac{L_i}{2} ||p_i^{(k)}||^2, \tag{10}$$

where the right-hand-side is quadratic in $p_i^{(k)}$. Exact coordinate-wise minimization implies that the actual function decrease is at least as large as the decrease predicted by minimizing the quadratic upper bound (w.r.t $p_i^{(k)}$). Substituting the exact minimizer results in:

$$f(z_{i+1}^{(k)}) \leq f(z_i^{(k)}) - \frac{1}{2L_i} ||\nabla_i f(z_i^{(k)})||^2, \tag{11}$$

which strengthens the inequality. The inequality holds for all k and all i, so, rearranging and summing over all iterations gives a telescoping sum. Moreover, f is bounded below, so,

$$\sum_{k=0}^{\infty} \sum_{i=1}^{N} ||\nabla_i f(z_i^{(k)})||^2 < \infty, \tag{12}$$

and

$$\lim_{k\to\infty} ||\nabla_i f(z_i^{(k)})|| = 0, \quad i = 1, 2, ..., N, \tag{13}$$

which implies stationarity of any limit point. □

Equation 11 is essentially a sufficient-decrease type condition, and the following steps required to prove convergence parallel those in the Zoutendijk line-search convergence proof (e.g., Nocedal and Wright Thm. 3.2 [21]). The Lipschitz gradient assumption is a common assumption used in Zoutendijk's theorem as well as conventional gradient-descent convergence proofs. Most differentiable MDO problems will have a Lipschitz-continuous gradient, and, if necessary the assumptions in this convergence theorem can be relaxed to the level set:

$$\mathbb{L} := \{x : f(x) \leq f(x^{(0)}\}, \tag{14}$$

because by the definition of the block coordinate descent algorithm it is guaranteed that the iterates never leave the level set $\mathbb{L}$.

So long as the sufficient decrease condition (11) is met (or even a relaxed version) the algorithm converges to a stationary point. This is a big difference between our convergence analysis and other convergence analyses that use exact optimality conditions to guarantee convergence. This opens up many possibilities, notably each block minimization does not need to be exact in order to guarantee convergence to a stationary point and convexity of f is not required. This suits MDO applications well, because MDO problems are rarely convex and each sub-problem minimization is always inexact to some degree because of finite convergence tolerances. Accelerated variants of our coordinate descent algorithm could exploit this by truncating each sub-problem prior to full convergence or loosening convergence tolerances during earlier coordinate descent iterations.

For constrained optimization problems, the logic used in Theorem IV.1 no longer applies because the minimum of the quadratic upper bound is not guaranteed to be feasible. This means that there is no simple way to place a guaranteed lower bound on objective function decrease. In fact, the exact minimum of the quadratic upper bound lies in the direction of the gradient, which for constrained problems will almost always lie outside the feasible space. Clearly, different logic is required to prove convergence of the coordinate descent algorithm for constrained MDO problems.

As with the unconstrained case, the sequence $\{f(x^{(k)})\}_{k=0}^{\infty}$ is monotonically decreasing, so it is guaranteed to converge to some f^* (with f still bounded below). Additionally, $\{x^{(k)}\}_{k=0}^{\infty} \subset \mathcal{X}$ is a feasible sequence because each sub-problem optimizer is guaranteed to return a feasible solution if the problem is well posed with local constraints and $x^{(0)}$ is feasible. Unfortunately this is not enough to conclude that $\{x^{(k)}\}$ converges or that $x^{(\infty)}$ is a KKT point.

In fact, without some restrictive assumptions, convergence to a KKT point is not guaranteed for non-convex problems with non-convex constraints. Convergence proofs by Bertsekas [17] or Grippo and Sciandrone [15] require, among other things, uniqueness of block minimizers, and/or convex feasible sets, both of which are rarely true for MDO problems. One way to address this is to convert all constraints to penalty terms in the objective function, thereby creating an unconstrained optimization problem with guaranteed convergence to a stationary point. However, this leads to ill-conditioned problems whose solutions may be slightly infeasible. One way to improve upon this is by using an augmented Lagrangian formulation for any equality constraints. The augmented Lagrangian relaxation of the monolithic optimization problem 1 is:

$$x^* = \underset{x\in\mathbb{R}^n}{\arg\min} \quad \mathcal{L}_A(x, \lambda; \mu, \gamma) := f(x) + \lambda^T h(x) + \tfrac{\mu}{2}||h(x)||^2 + \tfrac{\gamma}{2}||g(x)^+||^2, \tag{15}$$

where $g(x)^+ = \max(0, g(x))$, for the component-wise maximum. Note that the augmented Lagrangian method is only applied to the equality constraints, and the inequality constraints are handled with quadratic penalties. This is because the conventional way of adding inequalities to augmented Lagrangian methods requires slack variables which must be added as new constraints in the final problem formulation, which would defeat the purpose.

The distributed version of the monolithic augmented Lagrangian problem is:

$$x_1^* = \underset{x_1\in\mathbb{R}^{n_1}}{\arg\min} \quad \mathcal{L}_A(x, \lambda; \mu, \gamma) \quad \dots \quad x_N^* = \underset{x_N\in\mathbb{R}^{n_N}}{\arg\min} \quad \mathcal{L}_A(x, \lambda; \mu, \gamma). \tag{16}$$

Because the augmented Lagrangian distributed formulation is unconstrained, we can extend our global convergence theorem and proof for general unconstrained problems to also cover the augmented Lagrangian formulation. Algorithm 2

Algorithm 2 Augmented-Lagrangian block coordinate descent

Initialize $x^{(0)} \in \mathbb{R}^n$, $\mu^{(0)} > 0$, $\lambda^{(0)} = 0$, $\rho > 1$
$j \leftarrow 0$
while infeasible **do**
    $k \leftarrow 0$
    while not converged **do**
        for $i = 1, 2, ..., N$ **do**
            $x_i^* \leftarrow \arg\min_{x_i} \mathcal{L}_A(x^{(k)}, \lambda^{(j)}; \mu^{(j)})$
            update the i-th block of $x^{(k)}$ with x_i^*
        end for
        $x^{(k+1)} = x^{(k)}$
        $k \leftarrow k + 1$
    end while
    $\lambda^{(j+1)} = \lambda^{(j)} + \mu^{(j)} h(x^{(k)})$
    $\mu^{(j+1)} = \rho \mu^{(j)}$
    $j \leftarrow j + 1$
end while

shows one way that the standard augmented Lagrangian update can be incorporated into the block coordinate descent framework.

For many MDO problems, the initial guess $x^{(0)}$ will not be feasible. However, because we only consider local constraints in the distributed formulation, $x^{(1)}$ and all subsequent iterations will be feasible (i.e., $\{x^{(k)}\}_{k=1}^{\infty}$ is always a feasible sequence). Another way to think about this point is that each distributed problem is well posed, so a feasible solution is guaranteed to exist. And, a perfect optimizer is, by definition, guaranteed to return a feasible solution to a well posed problem.

Our convergence analysis for algorithm 1 holds for any variant of the block coordinate descent algorithm that satisfies an *essentially cyclic* rule [22]. That is, as long as each block is updated at least once in every Gauss-Seidel cycle, the algorithm is still guaranteed to converge. For example, on a problem with three blocks the update order at any iteration could be lexicographic (i.e., 1, 2, 3) or any permutation (e.g., 2, 1, 3). Cyclic coordinate descent is arguably the simplest framework, but it need not be used if some other ordering makes more sense for a specific problem. Moreover, the ordering does not need to remain fixed for every iteration.

A. Warm-starting with block coordinate descent

In this section we propose a distributed warm-starting strategy to improve the reliability of convergence for monolithic formulations by generating improved initial guesses. We do this by performing a few coordinate descent iterations to generate a warm-start point from which the monolithic formulation is more likely to converge. This warm-starting algorithm (3) adds a single monolithic solution to the end of the original block coordinate descent algorithm. The goal is not to run block coordinate descent until convergence, instead the (empirically determined) minimum possible number of distributed iterations are performed to generate a sufficiently good initialization for the monolithic problem. This warm starting strategy only works if the distributed problem is sufficiently simpler than the monolithic problem. For example, if most of the convergence difficulties in the monolithic problem arise from one discipline, then the distributed formulation will likely exhibit the same convergence difficulties. Because of this, our warm-starting approach is expected to suit situations where the convergence difficulty arises primarily from the coupling between disciplines.

V. Results

This section demonstrates the application of Algorithms 1, 2, and 3 to several MDO problems. Specifically, our test cases are engineering design problems in which each discipline models a distinct aspect of a complex and coupled system. All example problems are implemented in Python using either JAX or the computational system design language (CSDL), which employs a computational graph framework to automate the computation of adjoint derivatives. Optimization is carried out using modOpt [23], a modular optimization library that facilitates rapid experimentation with different optimization algorithms without requiring modifications to the problem formulation.

Algorithm 3 Warm-starting block coordinate descent

Initialize $x^{(0)} \in \mathbb{R}^n$
$k \leftarrow 0$
while not converged **do**
  for $i = 1, 2, ..., N$ **do**
    $x_i^* = \arg\min_{x_i \in \mathcal{X}_i} f(x)$ with initial values $x^{(k)}$
    update the i-th block of $x^{(k)}$ with x_i^*
  end for
  $k \leftarrow k + 1$
  $x^{(k)} \leftarrow x^{(k-1)}$
end while
$x^* = \arg\min_{x \in \mathcal{X}} f(x)$ with initial values $x^{(k)}$

All results were generated on a single workstation equipped with an Intel i7-7700HQ processor and 16 GB of memory. While our numerical experiments were conducted on a single machine, the implementation is general and could easily be adapted to more heterogeneous or distributed computing environments.

A. Example: distributed solution of a control co-design toy problem

Our first test problem has two coupled disciplines: control and design. The control problem involves computing an optimal swing-up trajectory for a cart-pole system, while the design problem focuses on optimizing the pole's length and mass. The overall objective is to minimize the control effort. The controls problem is solved with a collocation approach and trapezoidal integration. For this problem, we know that the IPOPT optimizer works well for the controls sub-problem and SciPy's SLSQP works well for the smaller, two-variable, design sub-problem. While either optimizer could likely solve both disciplines, we use this as a convenient way to demonstrate how each discipline in a distributed formulation can leverage a dedicated solver that is tailored for the specific problem. Subsequent results use the augmented Lagrangian block coordinate descent algorithm (2) due to the large number of equality constraints in the collocation residual.

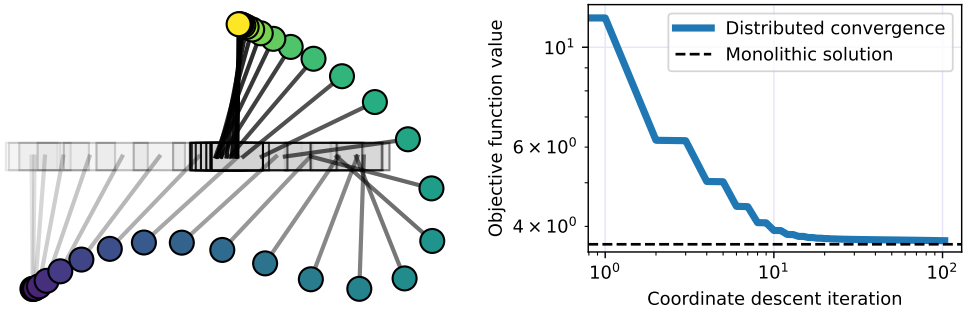

Fig. 4 Left: The cart-pole co-design toy problem. Right: Coordinate descent convergence history

The step-like behavior of the distributed convergence plot in Figure 4 is characteristic of the distributed method, where the objective has more sensitivity to one discipline than other disciplines. In this case, the objective is only a function of the control, which is only a variable in the trajectory optimization problem. In this example we show that the distributed formulation approaches the same solution as the monolithic formulation in a reasonable number of iterations, and from the same initial guess. In general, the distributed formulation may converge to a different local solution than the monolithic formulation.

Consider the following multi-point variation of the cart-pole control co-design problem where there are now multiple trajectory optimization problems each initialized from a different initial state (Fig. 5). This problem is intended to mimic the structure of very-large-scale aerospace MDO problems that have a single plant that is optimized for many design conditions. This type of problem is very expensive (1502 design variables and 1240 constraints), and represents one

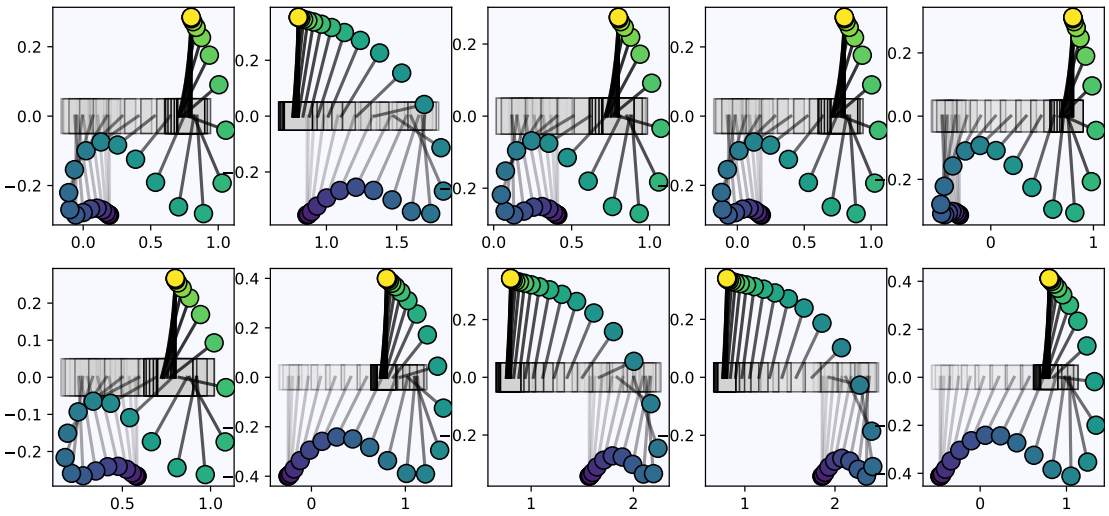

Fig. 5 The multi-point cart-pole co-design problem. Each cart-pole trajectory optimization problem is initialized at different initial conditions, creating a multi-point optimization problem that is representative of very large-scale optimization problems where computational resources may become restrictive.

situation where distributed optimization might be necessary due to computational limitations. Figure 5 shows the 10 optimal cart-pole trajectories for this problem.

As the monolithic problem size increases, so does the memory required for the model, its gradients, and any Hessian approximations. This is not the case for distributed formulations. Figure 6 shows that the peak memory allocated for the distributed formulation is simply the maximum required for any one of the distributed sub-problems. Because of this, the memory requirement remains constant as the number of cart-pole trajectory copies increases. While the memory required for the monolithic problem is not very large (approximately 400 MB), it does not take much effort to imagine a more complex problem that would quickly reach the limits of most personal computers. In this case, a distributed formulation is one way that the problem could still be solved without incurring such a large memory requirement. This benefit comes at a cost, which is that the wall time required to fully converge the distributed formulation is quite high. However, the time required for the monolithic formulation is also high, and convergence could not be achieved with more than nine trajectory copies. These results are for the vanilla block coordinate descent algorithm (1), and accelerations may be available that are not explored in this paper.

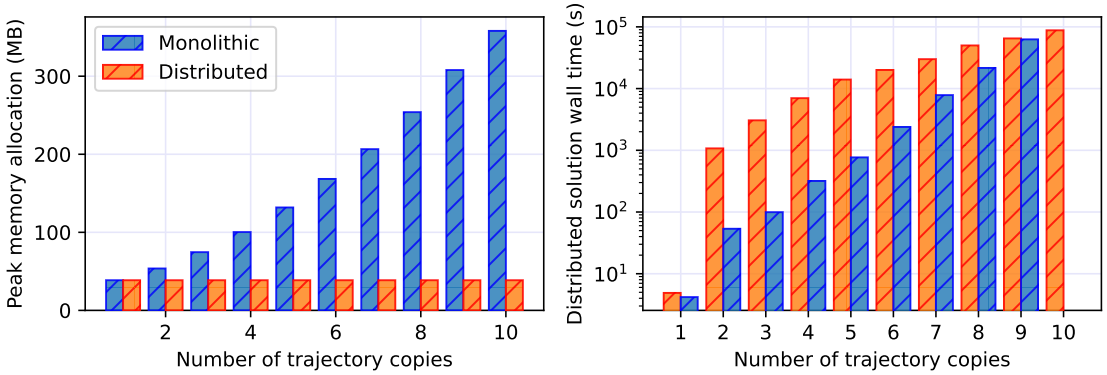

Fig. 6 Performance comparison between monolithic and distributed versions of the multi-point cart-pole control co-design problem.

B. Example: distributed aero-structural optimization

Next, we demonstrate our distributed solution method on two simple aero-structural optimization problems: NASA's Common Research Model (CRM) [24] and a blended-wing-body (BWB) cargo plane (see Fig. 7). Both problems have two disciplines, aerodynamics and structures. The aerodynamics discipline consists of a vortex-lattice-method model, which outputs spanwise loads into a one-dimensional beam structural sizing model. The objective of both optimization problems is to minimize fuel burn (computed using the Breguet range equation) for a fixed mission, subject to a lift-equals-weight constraint and a maximum wing displacement constraint. Fuel burn is minimized with respect to wing twist and wing box thickness, both of which are represented as vectors across the wing span. By default, the distributed version of this optimization problem is partitioned by discipline, with wing twist assigned to one sub-problem and thickness to the other.

The CRM and BWB aero-structural problems are highly coupled. In general, block coordinate descent methods converge slower with increased coupling. Figure 8 shows the convergence history for the CRM problem with varying modulus of elasticity. By changing the modulus we effectively change the coupling strength between sub-problems. As expected, increased coupling (lower modulus) results in more iterations required for convergence.

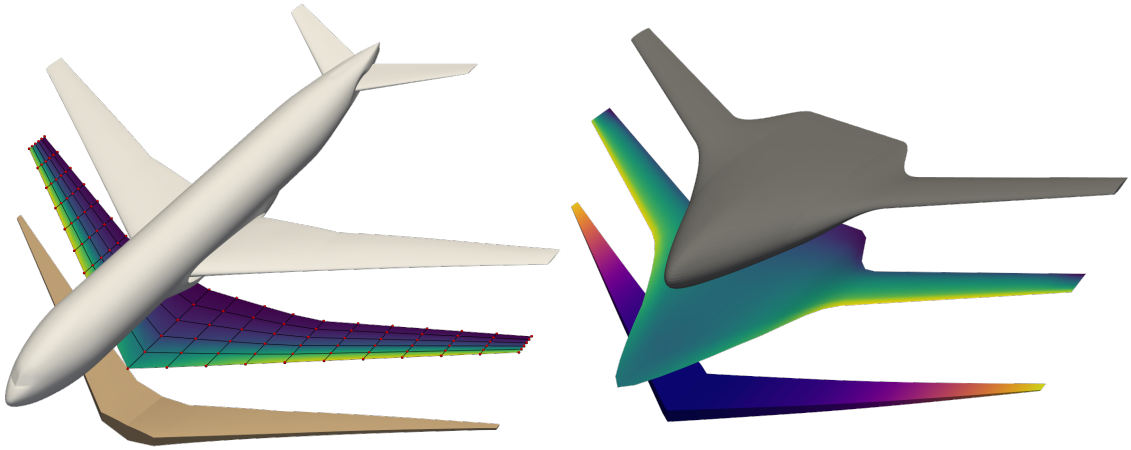

Fig. 7 Left: the aero-structural optimization problem for NASA's common research model [24]. Right: the blended-wing-body aero-structural optimization problem. Both problems use the vortex-lattice method for aerodynamics and 1D-beams for structures

It is logical but not necessarily optimal to partition these problems by discipline. Different convergence properties can be obtained by grouping some of the twist variables with some of the thickness variables. Figure 8 shows the results of three different decompositions. Formulation 1 is the default decomposition with twist assigned to one sub-problem and thickness to the other. Formulations 2 and 3 group some of the twist variables with some of the thickness variables for both sub-problems. It is clear that the block coordinate descent algorithm converges in the least number of iterations for formulation 3. Consequently, whenever possible, it makes sense to partition the problem based on testing or theoretical analysis instead of blindly following disciplinary boundaries.

Block coordinate descent exhibits slow convergence for highly coupled problems, a similar characteristic as the conventional Gauss-Seidel method for solving linear systems of equations and most distributed MDO architectures [25]. Specifically, our method slows down considerably near a solution, exhibiting tiny progress per step. Similar slow downs are expected in valleys. These effects are similar in nature to steepest descent methods, which are known for their zigzag behavior. It is difficult to perform a fair comparison between monolithic and distributed architectures, because the fundamental problem being solved at each iteration is different. Figure 9 shows solution error plotted against the number of function evaluations for the CRM and BWB problems. Clearly, our distributed approach requires several orders of magnitude more function evaluations than the monolithic problem. However, each distributed function evaluation is cheaper than any monolithic function evaluation, so this gap is not as bad as it looks upon first inspection.

Applying the warm-starting algorithm 3 to the CRM and BWB aero-structural problems has the potential to increase the odds of successful convergence from poor initializations. Figure 10 shows the results of 10 random initializations with and without warm-starting. For these results, failure is considered to be any error returned by the optimizer (SLSQP) or stalling/slow convergence with an iteration limit of 1000. For both problems, the number of successful optimizations

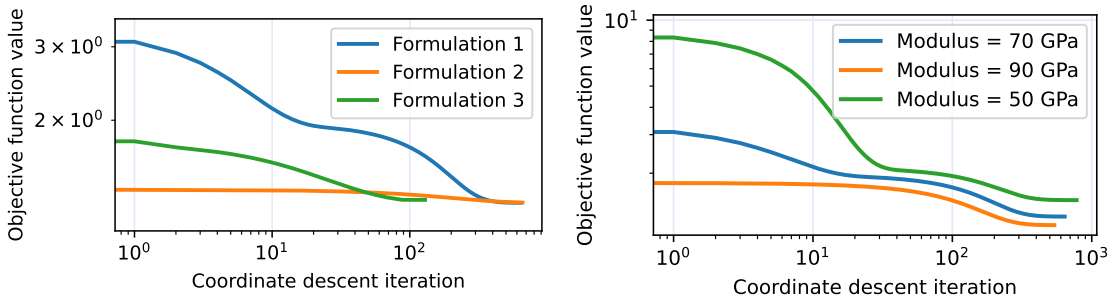

Fig. 8 Distributed convergence experiments for NASA's CRM. Left: different problem formulations affect convergence properties. Right: changing the modulus changes the amount of coupling between disciplines in the distributed formulation.

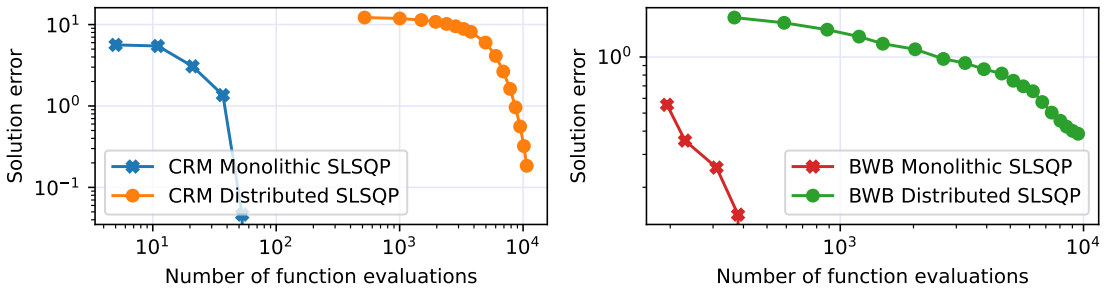

Fig. 9 Solution error vs. function evaluations for monolithic and distributed versions of the CRM and BWB optimization problems.

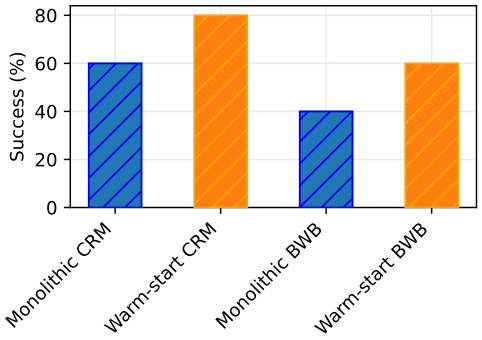

Fig. 10 Distributed warm-starting the CRM and BWB optimizations results in a higher percentage of successful optimizations initialized from 10 random initial guesses for all variables.

increased with just three coordinate descent iterations. This exemplifies one case where distributed formulations are better than monolithic formulations. We note that the success of our distributed approach for warm-starting is highly dependent on problem structure. The distributed problems must be sufficiently easier to solve than the monolithic problem to see any advantages, and any penalty functions on global constraints must be carefully tuned.

VI. Conclusion

This work was originally inspired by control co-design problems in MDO, where the optimization problem has two disciplines: design and control. In many cases the controls problem requires (or benefits from) dedicated solution algorithms that require different software from the design problem. Distributed optimization is one way to facilitate this, and the block coordinate descent method is conceptually a very simple distributed algorithm. Naturally, these benefits extend to optimization problems that require discipline-specific optimization algorithms, and to MDO problems that are required to be solved in a distributed manner due to organizational divisions, proprietary processes, ITAR regulations, etc. In contrast with other distributed MDO architectures, our method is fully decentralized so no single organization/computer system needs to be in charge.

Our contributions are summarized as follows. First, we propose a simple and practical architecture for solving distributed MDO problems using block coordinate descent. Second, we prove global convergence for unconstrained problems, which can be extended to constrained problems that have appropriate penalty terms. Third, we demonstrate the distributed solutions of a control co-design problem and two aero-structural optimization problems, with an extension to warm-starting for particularly complex problems.

Future work will focus on the novel application of distributed MDO architectures for automatically warm-starting very large-scale MDO problems. One possibility is that algorithms or heuristics can be used to automatically partition monolithic problems into distributed forms with superior convergence properties. In some cases, overlapping decompositions and/or decompositions that are re-defined at every iteration might perform better than the complete decompositions used in this paper.

One caveat is that, as a general rule, monolithic architectures exhibit a much better convergence rate than any distributed architecture [26], so the methods presented here only become relevant if the distributed/decentralized structure is strictly required. This is the case when, for example, organizational requirements demand distribution, when computational resources are limited, or when the monolithic formulation fails to reliably converge.

Acknowledgments

The authors would like to acknowledge the Multidisciplinary Science and Technology Center of the Aerospace Systems Directorate, Air Force Research Laboratory for funding and supporting this effort through the Collaborative Center for Design and Research of Interdisciplinary Systems program.

References

[1] 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.

[2] Braun, R. D., Moore, A. A., and Kroo, I. M., "Collaborative approach to launch vehicle design," *Journal of spacecraft and rockets*, Vol. 34, No. 4, 1997, pp. 478–486. https://doi.org/10.2514/2.3237.

[3] Sobieszczanski-Sobieski, J., "Optimization by decomposition: a step from hierarchic to non-hierarchic systems," *Second NASA/Air force symposium on recent advances in multidisciplinary analysis and optimization, Hampton, VA, NASA CP*, Vol. 3031, 1988, pp. 51–78.

[4] 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/10.1177/1063293X96004004.

[5] Michelena, N., Park, H., and Papalambros, P. Y., "Convergence properties of analytical target cascading," *AIAA journal*, Vol. 41, No. 5, 2003, pp. 897–905. https://doi.org/10.2514/2.2025.

[6] Braun, R., Gage, P., Kroo, I., and Sobieski, I., "Implementation and performance issues in collaborative optimization," *6th symposium on multidisciplinary analysis and optimization*, 1996, p. 4017. https://doi.org/10.2514/6.1996-4017.

[7] Pan, J., and Diaz, A., "Some results in optimization of non-hierarchic systems," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 3684, American Society of Mechanical Engineers, 1989, pp. 15–20. https://doi.org/10.1115/DETC1989-0069.

[8] Shankar, J., Haftka, R. T., and Watson, L. T., "Computational study of a nonhierarchical decomposition algorithm," *Computational Optimization and Applications*, Vol. 2, 1993, pp. 273–293. https://doi.org/10.1007/BF01299452.

[9] 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 2025 Forum*, 2025, p. 0579. https://doi.org/10.2514/6.2025-0579.

[10] Chauhan, S. S., Hwang, J. T., and Martins, J. R., "An automated selection algorithm for nonlinear solvers in MDO," *Structural and Multidisciplinary Optimization*, Vol. 58, 2018, pp. 349–377. https://doi.org/10.1007/s00158-018-2004-5.

[11] Heil, M., Hazel, A. L., and Boyle, J., "Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches," *Computational Mechanics*, Vol. 43, 2008, pp. 91–101. https://doi.org/10.1007/s00466-008-0270-6.

[12] Platt, J., "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. rep., Microsoft Research Technical Report, 1998.

[13] Friedman, J., Hastie, T., and Tibshirani, R., "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, Vol. 9, No. 3, 2008, pp. 432–441. https://doi.org/10.1093/biostatistics/kxm045.

[14] Luo, Z.-Q., and Tseng, P., "On the convergence of the coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, Vol. 72, No. 1, 1992, pp. 7–35. https://doi.org/10.1007/BF00939948.

[15] Grippo, L., and Sciandrone, M., "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints," *Operations research letters*, Vol. 26, No. 3, 2000, pp. 127–136. https://doi.org/10.1016/S0167-6377(99)00074-7.

[16] Sarojini, D., Ruh, M. L., Joshy, A. J., Yan, J., Ivanov, A. K., Scotzniovsky, L., Fletcher, A. H., Orndorff, N. C., Sperry, M., Gandarillas, V. E., et al., "Large-scale multidisciplinary design optimization of an evtol aircraft using comprehensive analysis," *AIAA SciTech 2023 forum*, 2023, p. 0146. https://doi.org/10.2514/6.2023-0146.

[17] Bertsekas, D. P., "Nonlinear programming," *Journal of the Operational Research Society*, Vol. 48, No. 3, 1997, pp. 334–334.

[18] Tosserams, S., Etman, L., and Rooda, J., "Augmented Lagrangian coordination for distributed optimal design in MDO," *International journal for numerical methods in engineering*, Vol. 73, No. 13, 2008, pp. 1885–1910. https://doi.org/10.1002/nme.2158.

[19] David, Y., Gallard, F., and Rondepierre, A., "Locally convergent bi-level MDO architectures based on the block coordinate descent algorithm," *Journal of Optimization Theory and Applications*, 2024.

[20] Beck, A., and Tetruashvili, L., "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, Vol. 23, No. 4, 2013, pp. 2037–2060. https://doi.org/https://doi.org/10.1137/120887679.

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

[22] Tseng, P., and Yun, S., "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, Vol. 117, 2009, pp. 387–423. https://doi.org/10.1007/s10107-007-0170-0.

[23] Joshy, A. J., and Hwang, J. T., "modOpt: A modular development environment and library for optimization algorithms," *arXiv preprint*, 2024. https://doi.org/10.48550/arXiv.2410.12942.

[24] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R., "Development of a common research model for applied CFD validation studies," *26th AIAA applied aerodynamics conference*, 2008, p. 6919. https://doi.org/10.2514/6.2008-6919.

[25] Lambe, A., and Martins, J. R., "A New Approach to Multidisciplinary Design Optimization via Internal Decomposition," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010, p. 9325. https://doi.org/10.2514/6.2010-9325.

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