This is the preprint version of the following article:
Anugrah Jo Joshy, and John T. Hwang. Unifying monolithic architectures for large-scale system design optimization. AIAA Journal, 2021.

Published article: https://doi.org/10.2514/1.J059954
Preprint pdf: https://github.com/LSDOlab/lsdo_bib/blob/main/pdf/joshy2021unifying.pdf
Bibtex: https://github.com/LSDOlab/lsdo_bib/blob/main/bib/joshy2021unifying.bib

Unifying Monolithic Architectures for Large-Scale System Design Optimization

Anugrah Jo Joshy* and John T. Hwang†

University of California San Diego, La Jolla, CA, 92093

Abstract

Large-scale system design optimization is a numerical technique used in solving system design problems that involve a large number of design variables. These systems are often multidisciplinary, with many disciplines interacting with each other. The scale of these problems demands a gradient-based approach for efficient solutions, and it is often implemented by coupling an engineering model with an optimizer. A recently developed theory on multidisciplinary derivative computation has made it feasible to solve large-scale system design optimization problems in only hundreds of model evaluations. This has led to an increase in the number of applications for large-scale system design optimization with new applications still emerging. This paper presents a new optimization formulation that can further reduce the required number of model evaluations by unifying two widely used optimization architectures, namely, multidisciplinary feasible, and simultaneous analysis and design. Complex engineering systems that require solutions of large nonlinear systems can potentially benefit from this new formulation, and the optimized solutions can be reached in just tens of equivalent model evaluations. We demonstrate this order of magnitude improvement using a bar design problem. The paper also provides details on the practical implementation of this new formulation in an equality-constrained optimization setting.

Nomenclature

$\frac{\mathrm{d}f}{\mathrm{d}x}$ = total derivative of a variable f with respect to a variable x

$\frac{\partial F}{\partial x}$ = partial derivative of a function F with respect to a variable x

This paper was originally presented at the *AIAA AVIATION 2020 FORUM* held virtually on June 15-19, 2020. Paper number: AIAA 2020-3125. https://doi.org/10.2514/6.2020-3125

*PhD Student, Department of Mechanical and Aerospace Engineering

†Assistant Professor, Department of Mechanical and Aerospace Engineering

1 Introduction

Multidisciplinary design optimization (MDO) is a field of engineering that uses numerical optimization techniques in design problems that involve multiple engineering disciplines. An MDO problem usually takes the form of a numerical optimization problem that minimizes an objective subject to constraints. In practice, such problems are solved by coupling a general-purpose optimization algorithm, called the optimizer, with an engineering model that computes the objective and constraints, as well as their derivatives. The optimizer evaluates the model at different design variable values until convergence, reading the outputs of the model.

Large-scale system design optimization (LSDO) focuses on problems that are high-dimensional, defined as having hundreds or more design variables, and that represent system-level design problems. High-dimensional design spaces and coupled multidisciplinary models are characteristics of design problems involving large-scale complex engineered systems as these systems have a large number of parts that interact with each other in unintuitive ways. For instance, a modern commercial airliner has millions of parts, and its design involves a large number of disciplines such as aerodynamics, structures, and propulsion that are coupled via feedback loops. In these complex design problems, intuition and experience have their limits; when that is the case, LSDO provides an alternative, which is a rigorous and automatic way to compute the best design given a sufficiently accurate model.

High-dimensional design spaces in large-scale systems necessitate the use of gradient-based optimizers in LSDO to ensure efficient scalability. Applying large-scale optimization to system design is challenging, because of the conflicting requirements of efficient derivative computation (for scalability) and coupling multiple disciplines (for system-level modeling). However, a recently developed theory [1] that unifies different total derivative computation methods (such as the chain rule, the coupled chain rule and the adjoint method) overcomes these challenges. OpenMDAO [2] is an open-source software framework from NASA that implements this theory to automate total derivative computation, following a modular approach in the construction of models.

These developments have enabled large-scale MDO to be accessible to a larger audience from different fields in the scientific community. The number of LSDO applications implemented in OpenMDAO has grown rapidly in the past few years. However, the current algorithms in LSDO still require hundreds of model evaluations to solve a problem and this poses a significant barrier to its adoption into an industrial setting.

Computer-aided engineering (CAE) tools such as computational fluid dynamics (CFD), finite element analysis (FEA), and other partial differential equation (PDE) solvers have significantly impacted industrial design processes by permeating into day-to-day workflows of engineers. For LSDO to have the same impact, the jump in computational cost from running a single simulation to solving an optimization problem with the same model should be limited to only tens of times. This would allow a 1-hour simulation to be optimized over a weekend rather than a few weeks, or a 1-minute simulation to be optimized in an hour rather than a day. In practice, these differences are significant in assessing whether LSDO can be a vital, integrated part of practical design cycles. This motivates the primary objective of this paper which is to accelerate the current LSDO algorithms through a significant reduction in the number of equivalent model evaluations.

In the current paradigm for LSDO, the optimizer views the model as a black-box that outputs the objective, constraints and their gradients. This paper presents a new, intrusive paradigm where the internal components of the model are exposed to the optimizer. An intrusive paradigm enables

a novel optimization algorithm that could achieve the robustness of a reduced-space formulation (also known as the MDF architecture) and the efficiency of a full-space formulation (also known as the SAND architecture) if the two formulations can be unified. The difference between the two is that full-space treats the model's state variables as design variables and the implicit equations that define the state variables as constraints.

Such an intrusive paradigm is used in the field of PDE-constrained optimization. PDE-constrained optimization is a field of research that deals with the optimization of partial differential equations (PDEs) involving large, three-dimensional meshes with up to billions of degrees of freedom. The envisioned feasibility of a reduction in the number of model evaluations is partly based on the success of the Lagrange–Newton–Krylov–Schur (LNKS) algorithm in solving PDE-constrained optimization problems with the cost of as low as five model evaluations [3, 4]. In this setting, the PDE solver, i.e., the model, and the optimizer are often integrated in a single piece of software where the optimizer is tailored to the PDE.

In LSDO, the model is complex, heterogeneous, and multidisciplinary—three reasons why the black-box model has been favored over the intrusive model, thus far. However, the proposed paradigm shift is timely because recent work in LSDO is trending towards the construction of models within computational frameworks (as in OpenMDAO). In this case, the work required to expand the interface is entirely on the framework side.

This paper presents a new, hybrid architecture and provides numerical results that validate its efficacy. The paper proceeds as follows. In Sec. 2, we provide some background for the unification algorithm with details on the unified derivatives equation (UDE), the reduced-space formulation, and the full-space formulation. In Sec. 3, we present a new algorithm that can achieve our aggressive speed improvement target for an equality-constrained case. This section also provides some details on the practical implementation of the algorithm. In Sec. 4, we solve a bar design problem using the reduced-space approach and our novel approach and then compare the results.

2 Background

2.1 Current paradigm

In the current approach for LSDO, an engineering model is built within a software framework (such as OpenMDAO) that couples an optimizer, from a library of optimizers, with the model. The coupled optimizer-model structure (shown in Fig. 1) built in the software framework solves a problem iteratively; the optimizer generates new design variable values based on the model outputs from the previous iteration. The iterations proceed until the optimality and feasibility criteria for the problem are satisfied, where optimality is the reduced-gradient norm and feasibility is the norm of constraint violations.

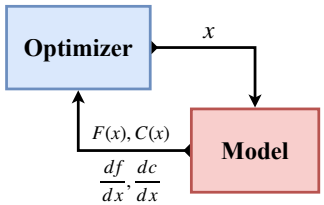


Figure 1: Current approach. The design variables are x. The objective and the constraints are respectively, $F(x)$ and $C(x)$.

With this approach, the computational cost of solving an optimization can be measured via the number of model evaluations required. For simplicity, we can treat both computing the objective and constraints, and computing the derivatives as one model evaluation because in efficient implementations, the computation times are similar. State-of-the-art LSDO methods can solve problems [5–11] with up to tens of thousands of design variables in only hundreds of model evaluations [12] as can be seen in Fig. 2.

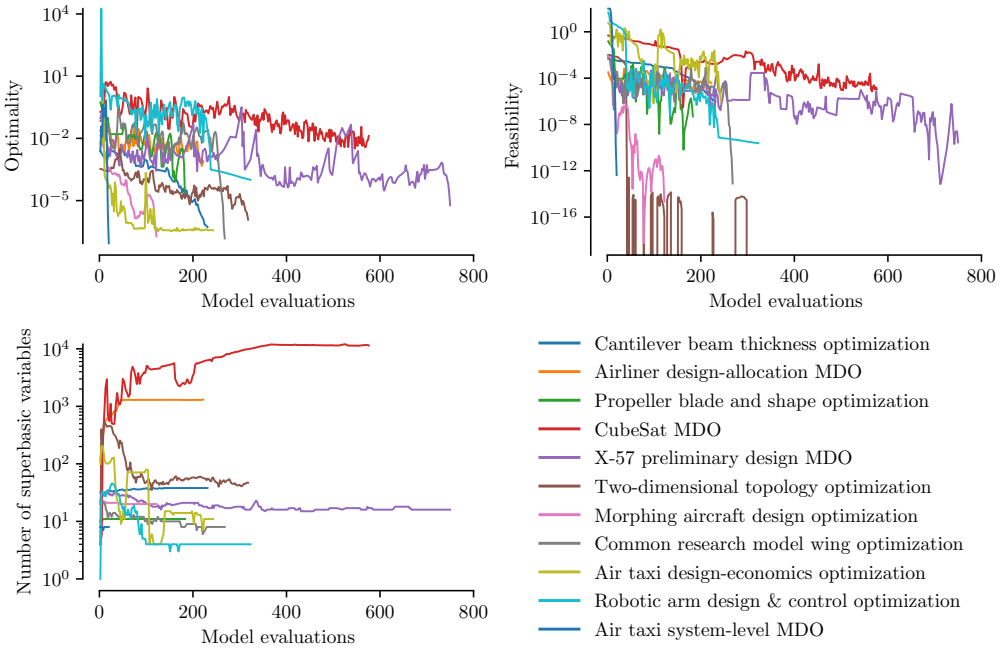


Figure 2: The state-of-the-art in LSDO. Previous LSDO problems require hundreds of model evaluations. Superbasic variables are the design variables that are not fixed due to bounds or constraints.

Given the current level of efficiency of LSDO, a paradigm shift is necessary to achieve further reductions in computation time. The only way to reduce the optimization time further is to reduce the cost of each running of the model which is possible by enabling partial model evaluations. Partial model evaluations are possible only with an intrusive paradigm where the optimizer has access to the operations performed inside a model. This makes it necessary to have computational frameworks that can facilitate such an intrusive environment. Therefore, an improvement over the state-of-the-art in LSDO calls for the creation of new optimizers and computational frameworks that can handle an intrusive paradigm.

2.2 MAUD: modular analysis and unified derivatives

LSDO is challenging because the difficult requirements for efficiency in large-scale problems are exacerbated by the complex, multidisciplinary models in system-level design problems. The scalability aspect necessitates gradient-based optimization and efficient, accurate derivative computation. The adjoint method is a critical technique because for a problem with n_x design variables,

it can reduce the gradient computation time in each optimization iteration from the cost of n_x model evaluations (in the finite-difference method) to less than one model evaluation.

The challenge is that the implementation of the adjoint method is time-intensive and it is specific to a particular choice of the output (objective or constraints). Therefore, any change in the model or the optimization problem requires deriving and implementing the new adjoint equations. This significantly reduces the usability of the adjoint method in a practical design setting, where the model is modified, disciplines are added or temporarily removed, and the optimization problem is tweaked frequently, within an iterative cycle. Moreover, the adjoint method only applies to a particular model structure where there is a single set of state variables implicitly defined by residuals. If all states are explicitly defined, or there are multiple disciplines (i.e., sub-models) with implicit states, or there is a combination of disciplines with implicit and explicit states, the adjoint method cannot be applied, and a different method such as the chain rule or the coupled adjoint method must be used. The modular analysis and unified derivatives (MAUD) architecture unifies all derivative computation methods using a single equation so that regardless of the model structure, solving this equation is mathematically equivalent to using the appropriate method: the adjoint method, chain rule, and so on.

We now present the equations underlying MAUD. A general optimization problem given a model with internal state variables can be stated as

$$\begin{aligned} \min_{x} \quad & F(x) && & \mathcal{R}(x, \mathcal{Y}(x)) = 0 \\ \text{subject to} \quad & x \geq 0 && \text{where} \quad & F(x) = \mathcal{F}(x, \mathcal{Y}(x)) \\ & C(x) = 0, && & C(x) = \mathcal{C}(x, \mathcal{Y}(x)), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ represents the design variables, $F : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}$ represent the *objective function*, $C : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathcal{C} : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^m$ represent the vector-valued *constraint function*, and $\mathcal{Y} : \mathbb{R}^n \to \mathbb{R}^r$ represents the implicit solution of $\mathcal{R}(x, \mathcal{Y}(x)) = 0$ as an explicit function. Therefore, we have n design variables, m constraints and r state variables. Moreover, we can define $y \in \mathbb{R}^r$ as the vector of *state variables*, $f \in \mathbb{R}$ as the *objective value*, and $c \in \mathbb{R}^m$ as the vector of *constraint values*.

The components in a complex model can be of many types: PDE solvers, surrogate models, closed-form expressions, or algebraic systems of equations. In all cases, we can describe the model by concatenating all variables into a single vector, $u \in \mathbb{R}^N$, and define an appropriate residual function $R : \mathbb{R}^N \to \mathbb{R}^N$. Solving the system $R(u) = 0$ is then equivalent to evaluating the model.

Under mild conditions, we can apply the inverse function theorem to R to obtain [1, 13]

$$\frac{\partial R}{\partial u}\frac{du}{dr} = \mathcal{I} = \frac{\partial R}{\partial u}^T \frac{du}{dr}^T , \tag{2}$$

where $\partial R/\partial u$ consists of partial derivatives of $\mathcal{R}$, $\mathcal{F}$, $\mathcal{C}$ and du/dr contains the derivatives we need: df/dx and dc/dx. Based on the model structure, the chain rule, the adjoint method, hybrid methods, and all other methods for computing discrete derivatives can be derived from Eq. (2). For example, we can derive the adjoint method by choosing $u = [x,\ y,\ f]^T$ and $R(u) = [x - x^*,\ -\mathcal{R}(x, y),\ f - \mathcal{F}(x, y)]^T$ (where x^* is the design variable vector at which we compute the adjoint), inserting into the right equality of Eq. (2), and applying block back-substitution.

MAUD can be summarized as follows: the user can implement their model as a modular set of components within a computational framework and provide partial derivatives of each component's outputs with respect to its inputs, where the combined set of partial derivatives form $\partial R/\partial u$. Then,

regardless of the model structure, the framework only needs to solve Eq. (2) to compute the model-level derivatives in the most efficient way (e.g., the adjoint method for a model with internal state variables).

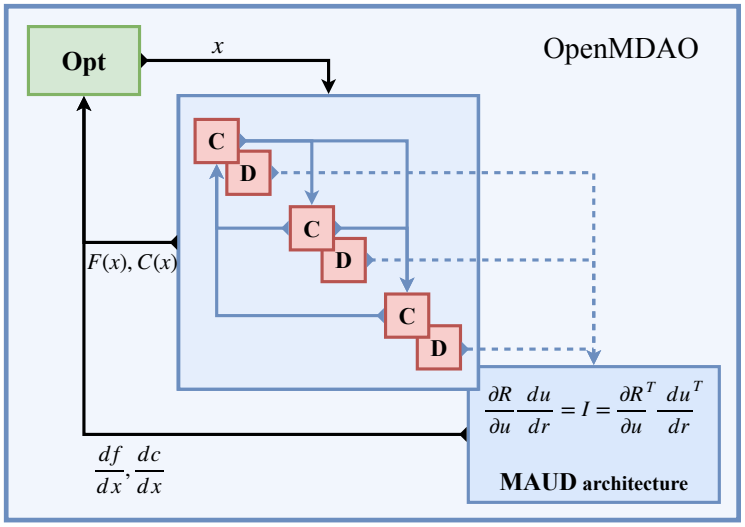


Figure 3: OpenMDAO framework. OpenMDAO couples the model with an optimizer and automatically computes the model derivatives using MAUD architecture. Opt is the optimizer, C and D denote the internal components of the model and their partial derivatives, respectively.

MAUD is implemented in NASA's OpenMDAO software framework (shown in Fig. 3), through which it has enabled LSDO problems in aircraft wing design [9, 10, 14, 15], satellite design [7, 16], airline route allocation optimization [17–19], jet engine design [20–24], and wind turbine design [25–29], among others [8, 30–32].

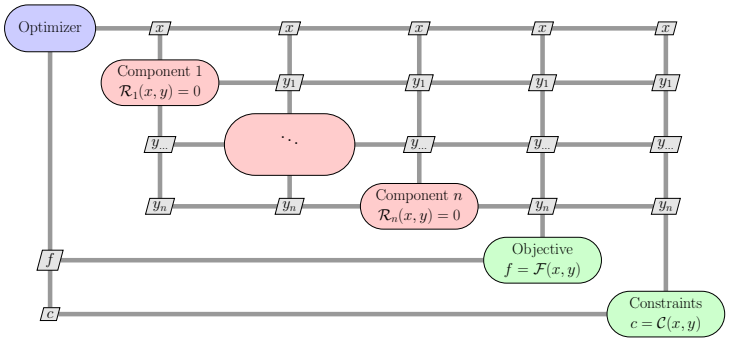
2.3 Optimization formulations

In MDO, 'architecture' or 'problem formulation' refers to the particular way in which a problem is defined and its solution is reached. Given a design optimization problem (1), there are multiple ways in which we can formulate it in order to find its solution. In this paper, we consider two popular monolithic MDO architectures (shown in Fig. 4): (1) the reduced-space (RS) architecture, and (2) the full-space (FS) architecture. The full-space formulation treats the model's state variables as additional design variables and the reduced-space formulation solves for the state variables within the model. The reduced-space architecture is also known as the multidisciplinary feasible (MDF) architecture or nested analysis and design (NAND), and the full-space architecture is also known as the simultaneous analysis and design (SAND) architecture.

In the reduced-space formulation, only x constitutes the optimization design variables, and y is computed implicitly as $\mathcal{Y}(x)$ by solving $\mathcal{R}(x, \mathcal{Y}(x)) = 0$. In the full-space formulation, both x and y are treated as design variables, and instead of evaluating $\mathcal{Y}(x)$, the responsibility of enforcing $\mathcal{R}(x, y) = 0$ is handed to the optimizer by making these constraints. The two formulations are so-named because full-space refers to the $(n+r)$-dimensional space of both x and y, while reduced-

Reduced-space:

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x, \mathcal{Y}(x)) \\ \text{subject to} \quad & \mathcal{C}(x, \mathcal{Y}(x)) = 0 \\ & x \geq 0 \\ \text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0 \end{aligned} \tag{3}$$



Full-space:

$$\begin{aligned} \min_{x,y} \quad & \mathcal{F}(x, y) \\ \text{subject to} \quad & \mathcal{C}(x, y) = 0 \\ & \mathcal{R}(x, y) = 0 \\ & x \geq 0 \end{aligned} \tag{4}$$

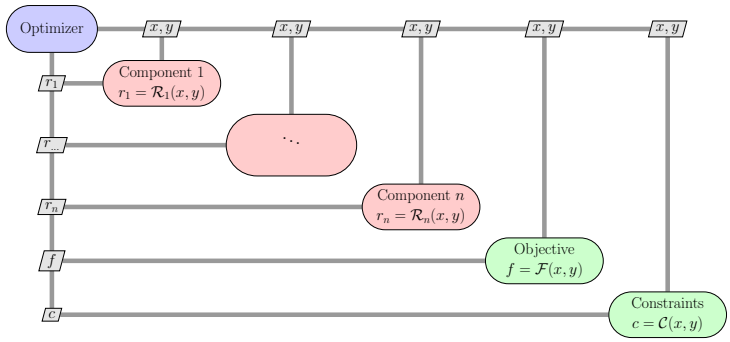


Figure 4: Reduced-space versus full-space formulation given a complex model with internal state variables.

space refers to the n-dimensional space of only x.

The tradeoff is that the model evaluation for reduced-space is more costly because it requires the nonlinear solution of $\mathcal{R}(x, y) = 0$, but the optimization problem is smaller and simpler. The model evaluation for full-space is simpler and less costly because it simply evaluates $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{R}$ without solving any nonlinear systems, but the optimization problem is larger and more difficult with additional constraints and design variables.

In general, full-space optimization has the potential to be more efficient but it is typically less robust. The larger problem may cause optimization convergence issues or negate the benefits from the more inexpensive model because of a greater increase in the number of evaluations. Reduced-space optimization is more robust, but it is inefficient because it wastes computation time accurately solving $\mathcal{R}(x, y) = 0$ to compute y in each optimization iteration even when x is far from converged. Tight convergence is necessary, in reduced-space methods, to ensure accurate computation of $\mathcal{F}(x, \mathcal{Y}(x))$ and $\mathcal{C}(x, \mathcal{Y}(x))$ so that the objective and constraint values are consistent with their derivatives with respect to x that are also provided by the model to the optimizer.

Next, we provide the optimality conditions and Karush-Kuhn-Tucker (KKT) systems for both formulations in the equality-constrained setting.

2.3.1 Reduced-space equations

We start with Eq. (3) with the bounds on x dropped, and we define the Lagrangian $l = \mathcal{L}(x, \lambda)$ where

$$\mathcal{L}(x,\lambda) = \mathcal{F}(x,\mathcal{Y}(x)) + \lambda^T \mathcal{C}(x,\mathcal{Y}(x)) \tag{5}$$

Applying the method of Lagrange multipliers, we obtain the first order necessary optimality conditions:

$$\begin{aligned} \frac{\mathrm{d}l}{\mathrm{d}x} &= \left(\frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y}\frac{\partial \mathcal{R}}{\partial y}^{-1}\frac{\partial \mathcal{R}}{\partial x}\right) + \lambda^T\left(\frac{\partial \mathcal{C}}{\partial x} - \frac{\partial \mathcal{C}}{\partial y}\frac{\partial \mathcal{R}}{\partial y}^{-1}\frac{\partial \mathcal{R}}{\partial x}\right) \\ \frac{\mathrm{d}l}{\mathrm{d}\lambda} &= \mathcal{C}(x,\mathcal{Y}(x)). \end{aligned} \tag{6}$$

Using p_k to denote the search direction, it then follows that the KKT system for reduced-space optimization is given by

$$\begin{bmatrix} l_{xx} & {c_x}^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ -\mathcal{C}(x_k,\mathcal{Y}(x_k)) \end{bmatrix}, \tag{7}$$

where $l_{xx} = d^2l/dx^2$, $c_x = dc/dx$, and $l_x = dl/dx$.

2.3.2 Full-space equations

We start with Eq. (4) with the bounds on x dropped, and we define the Lagrangian $m = \mathcal{M}(x,y,\psi,\lambda)$ where

$$\mathcal{M}(x,y,\psi,\lambda) = \mathcal{F}(x,y) + \psi^T \mathcal{R}(x,y) + \lambda^T \mathcal{C}(x,y) \tag{8}$$

Once again, applying the method of Lagrange multipliers, we obtain the first order necessary optimality conditions:

$$\begin{aligned} \frac{\mathrm{d}m}{\mathrm{d}x} &= \frac{\partial \mathcal{F}}{\partial x} + \psi^T\frac{\partial \mathcal{R}}{\partial x} + \lambda^T\frac{\partial \mathcal{C}}{\partial x}, \quad \frac{\mathrm{d}m}{\mathrm{d}\psi} = \mathcal{R}(x,y) \\ \frac{\mathrm{d}m}{\mathrm{d}y} &= \frac{\partial \mathcal{F}}{\partial y} + \psi^T\frac{\partial \mathcal{R}}{\partial y} + \lambda^T\frac{\partial \mathcal{C}}{\partial y}, \quad \frac{\mathrm{d}m}{\mathrm{d}\lambda} = \mathcal{C}(x,y). \end{aligned} \tag{9}$$

It then follows that the KKT system for full-space optimization is given by

$$\begin{bmatrix} m_{xx} & m_{xy} & {\mathcal{R}_x}^T & {\mathcal{C}_x}^T \\ m_{yx} & m_{yy} & {\mathcal{R}_y}^T & {\mathcal{C}_y}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ p_k^{(\psi)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k,y_k) \\ -\mathcal{C}(x_k,y_k) \end{bmatrix}, \tag{10}$$

where $m_{xx} = d^2m/dx^2$, $m_{xy} = d^2m/dydx$, $m_{yx} = d^2m/dxdy$, $m_{yy} = d^2m/dy^2$, $\mathcal{R}_x = \partial\mathcal{R}/\partial x$, $\mathcal{R}_y = \partial\mathcal{R}/\partial y$, $\mathcal{C}_x = \partial\mathcal{C}/\partial x$, $\mathcal{C}_y = \partial\mathcal{C}/\partial y$, $m_x = dm/dx$ and $m_y = dm/dy$.

3 Methodology

3.1 Corrected full-space method

Before introducing the hybrid algorithm, we define a new optimization method, based on the full-space formulation which we call the *corrected full-space method* (CFS). This method, as the name suggests, is a modified version of the full-space method and has the ability to generate

the same sequence of x and λ iterates as the reduced-space method, in an equality-constrained optimization setting. The feature that differentiates this method from the FS method is that it modifies the state variable vector, y, and the vector of Lagrange multipliers, ψ, associated with the residual equations before solving the full-space KKT system. If $[x_k, y_k, \psi_k, \lambda_k]^T$ are the values at the end of the kth iteration, this method corrects y_k to y'_k by solving $\mathcal{R}(x, y) = 0$ and then ψ_k to ψ'_k by setting $dm/dy = 0$. The FS KKT system (10) is then solved at the updated point $[x_k, y'_k, \psi'_k, \lambda_k]^T$ to obtain the new set of values for the next iteration.

We now prove that the sequence of iterates generated by the corrected full-space method and the reduced-space method are the same.

Theorem 1. *Assume (x_0, λ_0) are given. Then the sequence of iterates $\{(x_k, \lambda_k)\}$ generated by the reduced-space method and the corrected full-space method are identical in an equality-constrained optimization setting.*

Proof. We prove this theorem by induction.

At $k = 0$, the theorem holds trivially. Assuming that the theorem holds true at k, we prove that it holds true at $k + 1$. Let the kth iterate be $[x_k, \lambda_k]^T$ in the RS method and $[x_k, y_k, \psi_k, \lambda_k]^T$ in the corrected FS method.

In the $(k + 1)$th iteration of the corrected FS method, we start at $[x_k, y_k, \psi_k, \lambda_k]^T$ and solve $\mathcal{R}(x_k, y'_k) = 0$ to get an updated y'_k. We note that $y'_k = \mathcal{Y}(x_k)$ as $\mathcal{R}(x_k, y'_k) = 0$. Functions and gradients are now evaluated at $[x_k, y'_k]^T$.

Setting $dm/dy = 0$, we solve for ψ'_k from Eq. (9):

$$\frac{\mathrm{d}m}{\mathrm{d}y} = \frac{\partial \mathcal{F}}{\partial y} + \psi_k'^T \frac{\partial \mathcal{R}}{\partial y} + \lambda_k^T \frac{\partial \mathcal{C}}{\partial y} = 0 \quad \Longrightarrow \quad \psi'_k = -\left(\frac{\partial \mathcal{R}}{\partial y}\right)^{-T} \left(\frac{\partial \mathcal{F}}{\partial y}\right)^T - \left(\frac{\partial \mathcal{R}}{\partial y}\right)^{-T} \left(\frac{\partial \mathcal{C}}{\partial y}\right)^T \lambda_k. \tag{11}$$

This gives us the corrected values $[x_k, y'_k, \psi'_k, \lambda_k]^T$. We insert the expression for ψ'_k into the expression for dm/dx in Eq. (9) to obtain

$$\frac{\mathrm{d}m}{\mathrm{d}x} = \left(\frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y}\left(\frac{\partial \mathcal{R}}{\partial y}\right)^{-1} \frac{\partial \mathcal{R}}{\partial x}\right) + \lambda_k^T \left(\frac{\partial \mathcal{C}}{\partial x} - \frac{\partial \mathcal{C}}{\partial y}\left(\frac{\partial \mathcal{R}}{\partial y}\right)^{-1} \frac{\partial \mathcal{R}}{\partial x}\right). \tag{12}$$

Comparing with Eq. (6), we can see that $dm/dx = dl/dx$.

We now solve the FS KKT system (10) at the corrected point $[x_k, y'_k, \psi'_k, \lambda_k]^T$ where $\mathcal{R}(x_k, y'_k) = 0$, $m_y = 0$ and $m_x = l_x$. This gives us the following system:

$$\begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x^T & \mathcal{C}_x^T \\ m_{yx} & m_{yy} & \mathcal{R}_y^T & \mathcal{C}_y^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ p_k^{(\psi)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ 0 \\ 0 \\ -\mathcal{C}(x_k, \mathcal{Y}(x_k)) \end{bmatrix}. \tag{13}$$

Solving third row for $p_k^{(y)}$ and then second row for $p_k^{(\psi)}$, we get

$$p_k^{(y)} = -\mathcal{R}_y^{-1}\mathcal{R}_x p_k^{(x)} \quad \text{and} \quad p_k^{(\psi)} = -\mathcal{R}_y^{-T}\left((m_{yx} - m_{yy}\mathcal{R}_y^{-1}\mathcal{R}_x)p_k^{(x)} + \mathcal{C}_y^T p_k^{(\lambda)}\right). \tag{14}$$

Substituting $p_k^{(y)}$ and $p_k^{(\psi)}$ from above, the first and fourth rows become the following system:

$$\begin{bmatrix} \mathcal{Q} & {\mathcal{C}_x}^T - {\mathcal{R}_x}^T {\mathcal{R}_y}^{-T} {\mathcal{C}_y}^T \\ \mathcal{C}_x - \mathcal{C}_y {\mathcal{R}_y}^{-1} \mathcal{R}_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ -\mathcal{C}(x_k, \mathcal{Y}(x_k)) \end{bmatrix} \tag{15}$$

where $\mathcal{Q} = m_{xx} - m_{xy}\mathcal{R}_y^{-1}\mathcal{R}_x - \mathcal{R}_x^T\mathcal{R}_y^{-T}m_{yx} + \mathcal{R}_x^T\mathcal{R}_y^{-T}m_{yy}\mathcal{R}_y^{-1}\mathcal{R}_x$.

Using $dy/dx = -\mathcal{R}_y^{-1}\mathcal{R}_x$ and $c_x = \mathcal{C}_x - \mathcal{C}_y {\mathcal{R}_y}^{-1} \mathcal{R}_x$ gives us

$$\begin{bmatrix} m_{xx} + m_{xy}\dfrac{\mathrm{d}y}{\mathrm{d}x} + \dfrac{\mathrm{d}y}{\mathrm{d}x}^T m_{yx} + \dfrac{\mathrm{d}y}{\mathrm{d}x}^T m_{yy}\dfrac{\mathrm{d}y}{\mathrm{d}x} & c_x^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ -\mathcal{C}(x_k, \mathcal{Y}(x_k)) \end{bmatrix}. \tag{16}$$

Whenever $m_y = 0$, we can prove that

$$(m_{xx} + m_{xy}\frac{\mathrm{d}y}{\mathrm{d}x} + \frac{\mathrm{d}y}{\mathrm{d}x}^T m_{yx} + \frac{\mathrm{d}y}{\mathrm{d}x}^T m_{yy}\frac{\mathrm{d}y}{\mathrm{d}x}) = \frac{\mathrm{d}^2 f}{\mathrm{d}x^2} + \sum_{i=1}^{m} \lambda_i \frac{\mathrm{d}^2 c_i}{\mathrm{d}x^2} = l_{xx} \tag{17}$$

using the identity

$$\frac{\mathrm{d}^2 f}{\mathrm{d}x^2} = \frac{\partial^2 \mathcal{F}}{\partial x^2} + \frac{\partial^2 \mathcal{F}}{\partial y \partial x}\frac{\mathrm{d}y}{\mathrm{d}x} + \frac{\mathrm{d}y}{\mathrm{d}x}^T \frac{\partial^2 \mathcal{F}}{\partial x \partial y} + \frac{\mathrm{d}y}{\mathrm{d}x}^T \frac{\partial^2 \mathcal{F}}{\partial y^2}\frac{\mathrm{d}y}{\mathrm{d}x} + \sum_{i=1}^{r} \frac{\partial \mathcal{F}}{\partial y_i}\frac{\mathrm{d}^2 y_i}{\mathrm{d}x^2} \tag{18}$$

on functions $\mathcal{F}$, $\mathcal{C}_i$ and $\mathcal{R}_i$.

Substituting Eq. (17) in Eq. (16), we get

$$\begin{bmatrix} l_{xx} & c_x^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ -\mathcal{C}(x_k, \mathcal{Y}(x_k)) \end{bmatrix}. \tag{19}$$

This is exactly the RS KKT system (Eq. (7)) which means that (x_{k+1}, λ_{k+1}) from the $(k+1)$th iteration of reduced-space and corrected full-space methods are identical. □

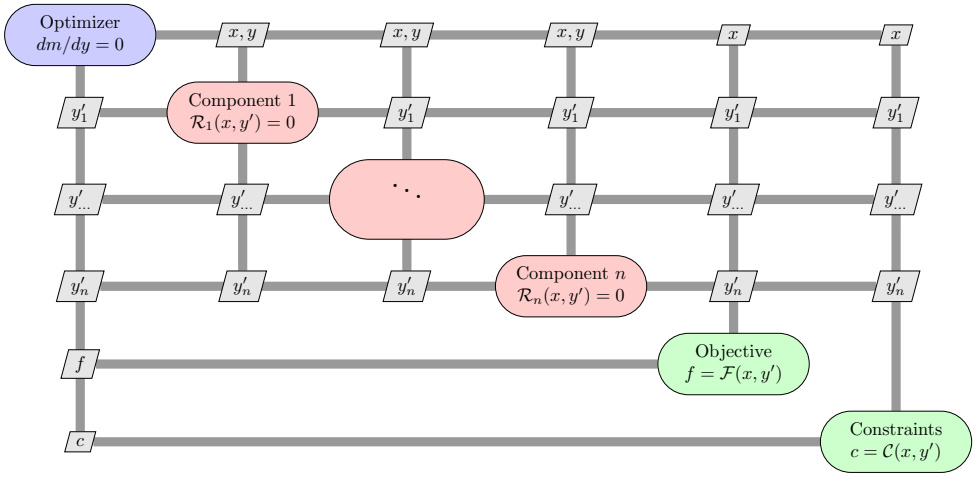


Figure 5: Corrected full-space architecture.

The corrected full-space method (CFS) shown in Fig. 5 is, in a way, a hybrid of the FS and RS methods and requires an intrusive paradigm as the optimizer updates the state variable vector y_k to y'_k with the help of solvers inside the model. However, CFS does not offer us much advantage in terms of computational efficiency as we still need to solve the nonlinear system $\mathcal{R}(x, y) = 0$, the same number of times as in the reduced-space method, before we reach the solution.

CFS is still a useful tool since it has the characteristics of both the full-space and reduced-space formulations. In the next subsection, we use CFS to formulate a novel algorithm that has the ability to vary the extent to which it can exhibit the properties of FS and RS algorithms. The new algorithm provides a complete unification of the full-space and reduced-space formulations for equality-constrained optimization problems.

A comparison of the reduced-space, full-space and corrected full-space methods:

Algorithm 1 RS method

1: **loop**
2: Run the model at x_k solving $\mathcal{R}(x_k, \mathcal{Y}(x_k)) = 0$
3: Assemble A_k, b_k
4: Solve $A_k p_k = b_k$
5: Update $\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} + p_k$
6: **end loop**

$$A_k = \begin{bmatrix} l_{xx} & c_x \\ c_x & 0 \end{bmatrix}$$

$$b_k = \begin{bmatrix} -l_x \\ -\mathcal{C}(x_k, \mathcal{Y}(x_k)) \end{bmatrix}$$

Algorithm 2 FS method

1: **loop**
2: Run the model at (x_k, y_k)
3: Assemble A_k, b_k
4: Solve $A_k p_k = b_k$
5: Update $\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \psi_k \\ \lambda_k \end{bmatrix} + p_k$
6: **end loop**

$$A_k = \begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & \mathcal{C}_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & \mathcal{C}_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix}$$

$$b_k = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k, y_k) \\ -\mathcal{C}(x_k, y_k) \end{bmatrix}$$

Algorithm 3 CFS method

1: **loop**
2: Run the model at (x_k, y_k) solving $\mathcal{R}(x_k, y'_k) = 0$
3: Compute ψ'_k by solving $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - \mathcal{C}_y^T \lambda_k$
4: Assemble A_k, b_k
5: Solve $A_k p_k = b_k$
6: Update $\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y'_k \\ \psi'_k \\ \lambda_k \end{bmatrix} + p_k$
7: **end loop**

$$A_k = \begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & \mathcal{C}_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & \mathcal{C}_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix}$$

$$b_k = \begin{bmatrix} -m_x \\ 0 \\ 0 \\ -\mathcal{C}(x_k, y'_k) \end{bmatrix}$$

3.2 A unification for the equality-constrained optimization setting

To demonstrate the feasibility of unifying the reduced-space and full-space algorithms, we present the unification for an equality-constrained optimization setting. We call the basic algorithm that achieves the unification SURF, which stands for *strong unification of reduced-space and full-space*, where "strong" references the fact that LNKS only links the linear systems in the RS and FS methods while SURF provides a complete unification of both the methods. SURF is presented in Algorithm 4. The loop that begins in line 1 marks the start of each outer iteration. Line 4 updates the approximation to the Hessian of the full-space Lagrangian and lines 5 and 6 represent the assembly and solution of the KKT system for full-space optimization with a preconditioner. Lines 7 and 8 are standard steps for computing the step size and applying the step, respectively. Note that we assume iterative solution of the KKT system; to consider the SQP setting, we can

simply ignore the preconditioner.

Algorithm 4 SURF (strong unification of reduced-space and full-space)

SURF unifies the reduced- and full-space methods for an equality-constrained optimization setting.

1: **loop**
2: Run the model at (x_k, y_k) inexactly solving $\mathcal{R}(x_k, y'_k) = 0$
3: Compute ψ'_k by inexactly solving $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - \mathcal{C}_y^T \lambda_k$
4: Update the approximation to the Lagrangian Hessian, H_k, at $[x_k, y'_k, \psi'_k, \lambda_k]^T$
5: Assemble A_k, b_k, $\tilde{M}_k^{-1}$
6: Solve $\tilde{M}_k^{-1} A_k p_k = \tilde{M}_k^{-1} b_k$
7: Compute α_k via a line search
8: Update $\begin{bmatrix} x_{k+1}, y_{k+1}, \psi_{k+1}, \lambda_{k+1} \end{bmatrix}^T = \begin{bmatrix} x_k, y'_k, \psi'_k, \lambda_k \end{bmatrix}^T + \alpha_k p_k$
9: **end loop**

Note:

1. $H_k = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix}$ is the Hessian of the Lagrangian, p_k is the search direction and α_k is the step size.
2. $A_k p_k = b_k$ is the FS KKT system (10), and M_k^{-1} and $\tilde{M}_k^{-1}$ are exact and approximate preconditioners for A_k such that $M_k = M_1 M_{2,k} M_{3,k} M_4$, and

$$A_k = \begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & \mathcal{C}_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & \mathcal{C}_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix}, b_k = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k, y'_k) \\ -\mathcal{C}(x_k, y'_k) \end{bmatrix}, M_1 = \begin{bmatrix} 0 & 0 & \mathcal{I} & 0 \\ 0 & \mathcal{I} & 0 & 0 \\ \mathcal{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{I} \end{bmatrix}, M_{2,k} = \begin{bmatrix} \mathcal{R}_y & 0 & 0 & 0 \\ m_{yy} & \mathcal{R}_y^T & 0 & 0 \\ m_{xy} & \mathcal{R}_x^T & \mathcal{I} & 0 \\ \mathcal{C}_y & 0 & 0 & \mathcal{I} \end{bmatrix}$$

$$M_{3,k} = \begin{bmatrix} \mathcal{I} & 0 & \mathcal{R}_y^{-1}\mathcal{R}_x & 0 \\ 0 & \mathcal{I} & \mathcal{R}_y^{-T} m_{yx} - \mathcal{R}_y^{-T} m_{yy} \mathcal{R}_y^{-1} \mathcal{R}_x & \mathcal{R}_y^{-T} \mathcal{C}_y^T \\ 0 & 0 & m_{xx} - m_{xy}\mathcal{R}_y^{-1}\mathcal{R}_x - \mathcal{R}_x^T \mathcal{R}_y^{-T} m_{yx} + \mathcal{R}_x^T \mathcal{R}_y^{-T} m_{yy} \mathcal{R}_y^{-1} \mathcal{R}_x & \mathcal{C}_x^T - \mathcal{R}_x^T \mathcal{R}_y^{-T} \mathcal{C}_y^T \\ 0 & 0 & \mathcal{C}_x - \mathcal{C}_y \mathcal{R}_y^{-1} \mathcal{R}_x & 0 \end{bmatrix}, M_4 = \begin{bmatrix} 0 & \mathcal{I} & 0 & 0 \\ 0 & 0 & \mathcal{I} & 0 \\ \mathcal{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{I} \end{bmatrix}.$$

Without lines 2 and 3, Algorithm 4 is the Lagrange–Newton–Krylov–Schur (LNKS) algorithm [3, 4] developed for PDE-constrained optimization. LNKS is so-called because the method of Lagrange multipliers yields the KKT optimality conditions, which are solved with Newton's method, which in turn constructs linear systems solved using a Krylov subspace method with a Schur complement preconditioner. The defining characteristic of LNKS is the Schur complement preconditioner that amounts to an inexact version of the reduced-space linear system, but LNKS is still full-space.

Lines 2 and 3 are what enable SURF to unify the reduced- and full-space formulations. Before solving the full-space KKT system in lines 4, 5, and 6, we inexactly solve the nonlinear system representing the model to update y_k to y'_k at x_k. Subsequently, we compute ψ'_k using an equation that comes from setting dm/dy to zero in Eq. (9). *If lines 2 and 3 are skipped, SURF becomes a full-space algorithm; if the two systems in lines 2 and 3 are exactly solved, SURF becomes a*

reduced-space algorithm; and if lines 2 and 3 uses inexact solvers, SURF becomes a hybrid. We know that SURF (Algorithm 4) without lines 2 and 3 is just the full-space algorithm. We can also see that SURF with lines 2 and 3 performed with exact solvers is the corrected full-space algorithm; it produces the same (x_k, λ_k) iterates as the reduced-space algorithm. This follows directly from the property of the corrected full-space method discussed in the previous section. A similar algorithm where a subset of variables are inexactly solved within a Newton iteration was proposed previously by Yang et al. [33]. However, they apply this approach to field variables in a PDE, whereas we apply it to the state and adjoint variables.

The preconditioner M_k in Algorithm 4 corresponds to a Schur complement decomposition of the full-space matrix, following the LNKS approach [3]. M_1 and M_4 in the definition of M_k are permutation matrices, and they are easily inverted. The Schur complement is the $(3, 3)$ block of $M_{3,k}$, and the lower-right 2×2 block of $M_{3,k}$ is exactly the KKT matrix of the reduced-space algorithm when we set $m_y = 0$ in the full-space algorithm. A single matrix-vector product of this 2×2 block can be computed with two model Jacobian linear solutions, one of $\mathcal{R}_y$ and one of $\mathcal{R}_y^T$.

The significance of this unification is twofold. First, switching between reduced-space and full-space is simple; we only have to change the inexact solver tolerances. Second, the SURF algorithm allows easy access to a continuous spectrum of hybrid methods. We note that the choice on this spectrum can be made on a variable-by-variable basis. Since the model is implemented as a collection of modular components as shown in Fig. 6, certain components can be more tightly converged than others.

In summary, SURF unifies the reduced-space and full-space algorithms, and it enables effortlessly selecting one of the two or a hybrid, simply by changing the solver tolerances in lines 2 and 3 of Algorithm 4.

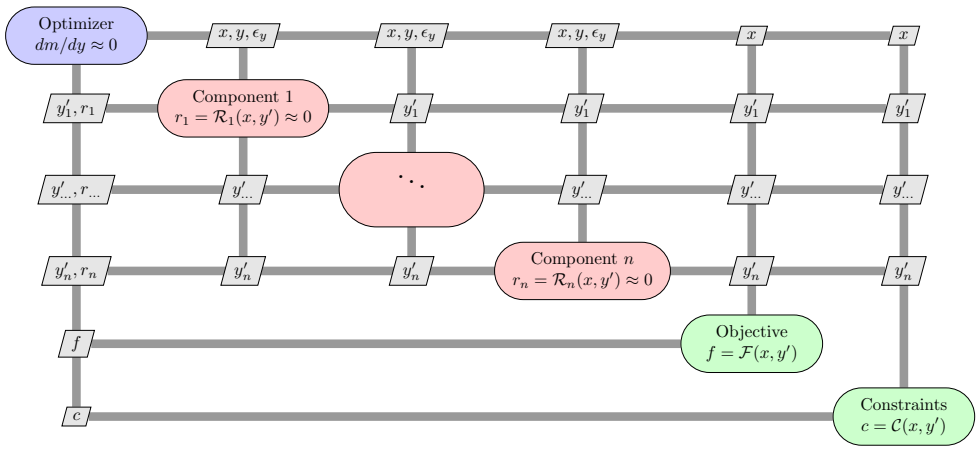


Figure 6: SURF architecture. Optimizer computes tolerances ϵ_y to which the residuals are solved within the model.

3.3 Some implementation details

In practice, equality-constrained optimization problems are solved using sequential quadratic programming (SQP) where the search directions are obtained from a sequence of quadratic pro-

gramming (QP) subproblems. Each QP subproblem minimizes a quadratic problem subject to linearized constraints, and under certain conditions, it is equivalent to solving the KKT system. The directions derived from the QP subproblem is then used in a line search to find the step towards the next iterate.

Implementation of a general-purpose optimizer requires addressing some practical aspects of an optimization algorithm: (1) Hessian approximations, and (2) a line search that guarantees global convergence. The following subsections give recommendations on using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for Hessian approximation and using a line search method that ensures global convergence, in the purview of SURF. These are suggestions by the authors based on the literature available on practical optimizers, and they are not validated on any test problem.

3.3.1 Positive-definite BFGS approximation of the Hessian of the full-space Lagrangian

Within a single iteration of SURF, we take two steps as opposed to a single step taken in any conventional algorithm. The first step updates just y and ψ, which we call the 'correction step' as it corrects the value of y and ψ from the previous iteration. The second step is the solution of the KKT system which we call the 'QP step' as it is the conventional step taken in a direction given by the QP subproblem.

We can define the different points in iteration $(k+1)$ as: $\xi_k = [x_k, y_k, \psi_k, \lambda_k]^T$ (at the start of the iteration), $\xi'_k = [x_k, y'_k, \psi'_k, \lambda_k]^T$ (at the corrected point) and $\xi_{k+1} = [x_{k+1}, y_{k+1}, \psi_{k+1}, \lambda_{k+1}]^T$ (at the end of the iteration). This gives us the correction step p'_k and the QP step $\alpha_k p_k$ as follows:

$$p'_k = \begin{bmatrix} x_k - x_k \\ y'_k - y_k \\ \psi'_k - \psi_k \\ \lambda_k - \lambda_k \end{bmatrix} = \begin{bmatrix} 0 \\ p_k'^{(y)} \\ p_k'^{(\psi)} \\ 0 \end{bmatrix} \quad \text{and} \quad \alpha_k p_k = \begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y'_k \\ \psi_{k+1} - \psi'_k \\ \lambda_{k+1} - \lambda_k \end{bmatrix} = \begin{bmatrix} \alpha_k p_k^{(x)} \\ \alpha_k p_k^{(y)} \\ \alpha_k p_k^{(\psi)} \\ \alpha_k p_k^{(\lambda)} \end{bmatrix} \tag{20}$$

We can denote the design variable vector as v, which includes both x and y such that $v_k = [x_k, y_k]^T$, $v'_k = [x_k, y'_k]^T$, and $v_{k+1} = [x_{k+1}, y_{k+1}]^T$.

General-purpose optimization algorithms based on SQP use a positive-definite BFGS approximation for the Lagrangian Hessian when solving a QP subproblem. In SURF, we can approximate the Hessian of the Lagrangian

$$H_k = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix} \tag{21}$$

at the corrected point ξ'_k using a modified BFGS algorithm. Let us denote the positive-definite BFGS approximation of H_k (generally indefinite, at a solution) as $\widehat{H}_k$. We note that $\widehat{H}_k$ is a positive-definite approximation of the Lagrangian Hessian rather than the projected Lagrangian Hessian (Hessian projected onto the null space of the Jacobian of the constraints) which is, in general, positive-definite near a solution. $\widehat{H}_k$ can be estimated from $\widehat{H}_{k-1}$ using the BFGS update formula

$$\widehat{H}_k = \widehat{H}_{k-1} - \frac{1}{d_k^T \widehat{H}_{k-1} d_k} \widehat{H}_{k-1} d_k d_k^T \widehat{H}_{k-1} + \frac{1}{w_k^T d_k} w_k w_k^T \quad , \tag{22}$$

where

$$d_k = v'_k - v'_{k-1} \quad \text{and} \quad w_k = \nabla \mathcal{M}(v'_k, \psi'_k, \lambda_k) - \nabla \mathcal{M}(v'_{k-1}, \psi'_k, \lambda_k)$$

The gradients of $\mathcal{M}$ are taken with respect to v and we note here that ψ'_k and λ_k are our best available estimates for the Lagrange multipliers.

Given $\widehat{H}_{k-1}$ is positive-definite, the updated BFGS approximation $\widehat{H}_k$ remains positive-definite if and only if $w_k^T d_k > 0$. The approximate curvature $w_k^T d_k$ may not be positive always since we are trying to approximate a Hessian that is, in general, indefinite. With SQP using quasi-Newton methods, Powell [34] states that the iterates converge towards a solution along a path that lies in the null space of the constraint Jacobian. Since H_k is positive-definite along the constraint surface near the solution, the approximate curvature $w_k^T d_k$ becomes positive as the iterates converge closer to a minimizer of the problem.

When the iterates are far from the solution, this generally doesn't hold. However, in cases where $w_k^T d_k$ is negative but the curvature along the constraint surface is positive, we can use a new scheme to find a positive approximate curvature. We compute a new step d_k^n that lies in the null space of the Jacobian of the constraints J'_k at v'_k. We note that the constraints now refer to both the constraints and the residuals, and the constraint Jacobian J'_k is the lower left 2×2 block of the matrix A_k in the SURF KKT system in Algorithm 4.

Under the assumption that J'_k has full row rank, we find $d_k^n = (I - J_k'^T[J'_k J_k'^T]^{-1} J'_k) d_k$ as the projection of d_k on the null space of J'_k. With the new step, we define a new point, $v_{k-1}'^n = v'_k - d_k^n$, which is the projection of v'_{k-1} on the null space of J'_k. We also define a new update pair (d_k^n, w_k^n) with respect to the points $v_{k-1}'^n$ and v'_k as

$$d_k^n = v'_k - v_{k-1}'^n \quad \text{and} \quad w_k^n = \nabla \mathcal{M}(v'_k, \psi'_k, \lambda_k) - \nabla \mathcal{M}(v_{k-1}'^n, \psi'_k, \lambda_k). \tag{23}$$

When $(w_k^n)^T d_k^n$ is positive, we update the Hessian using the new update pair (d_k^n, w_k^n). However, when the curvature along the constraint surface is negative, $(w_k^n)^T d_k^n$ is also negative and in such cases, we skip the update and set $\widehat{H}_k = \widehat{H}_{k-1}$.

In practice, we might encounter situations where inverting $[J'_k J_k'^T]$ is computationally expensive. In such situations, an alternative to the above-mentioned scheme could be used but the downside is that this alternate scheme is always well-defined and would be applied even in cases where it is unnecessary, i.e., even when $(w_k^n)^T d_k^n$ is negative. Let σ_k (> 0) be the minimum allowable approximate curvature. When $w_k^T d_k < \sigma_k$, we can define a Δw_k such that $\overline{w}_k = w_k + \Delta w_k$, and $\overline{w}_k^T d_k = \sigma_k$. We can now compute $\overline{w}_k$ from Δw_k given by

$$\Delta w_k = \frac{\sigma_k - w_k^T d_k}{w_k^T d_k - d_k^T \widehat{H}_{k-1} d_k} (w_k - \widehat{H}_{k-1} d_k). \tag{24}$$

The modified update pair $(d_k, \overline{w}_k)$ can then be used to compute the new Hessian approximation.

We should note the following about the original modification (23) for the Hessian approximation. Whenever the approximate curvature is not positive, we need to make an additional model evaluation at the new point $v_{k-1}'^n$ but this evaluation is not expensive as it does not invoke the nonlinear solvers for $\mathcal{R}(x, y) = 0$. Also, similar modifications are rarely needed more than a few times in conventional SQP algorithms [34] and therefore, we assume the same applies for SURF. In essence, with our new scheme, positive-definite Hessian updates can be made without incurring significant computational costs.

3.3.2 Line search guaranteeing global convergence

Line searches use merit functions as tools to measure the progress of the optimization algorithm. Any standard merit function, such as the l_1 or l_∞-penalty function, can be used in a line search to

find the step length $\alpha_k \in (0, 1]$ that satisfies the sufficient decrease condition. However, in order to make use of fast line search methods that enforce the strong Wolfe conditions and are not susceptible to the Maratos effect, we need smooth merit functions. The augmented Lagrangian merit function is one such smooth merit function used in state-of-the-art gradient-based optimizers. We provide an outline on using an augmented Lagrangian merit function in a line search algorithm within SURF that enforces the strong Wolfe conditions.

The augmented Lagrangian for SURF can be written as

$$L_A(x, y, \psi, \lambda; \rho) = \mathcal{F}(x, y) + \psi^T \mathcal{R}(x, y) + \lambda^T \mathcal{C}(x, y) + \frac{1}{2}\rho(\mathcal{C}(x, y)^T \mathcal{C}(x, y) + \mathcal{R}(x, y)^T \mathcal{R}(x, y)) \quad (25)$$

where ρ is the penalty parameter that penalizes the constraint violations and the residuals. The merit function for the $(k + 1)$th iteration of SURF based on the augmented Lagrangian is

$$\mathcal{G}_k(\alpha; \rho) = L_A(x_k + \alpha p_k^{(x)},\ y'_k + \alpha p_k^{(y)},\ \psi'_k + \alpha p_k^{(\psi)},\ \lambda_k + \alpha p_k^{(\lambda)}; \rho). \quad (26)$$

For a given ρ, $\mathcal{G}_k(\alpha; \rho)$ represents the augmented Lagrangian as a function of the step length, α. In order to guarantee a sufficient decrease along the direction p_k, ρ is updated, if required, before starting the line search in each iteration. With $\mathcal{G}'_k(\alpha; \rho) = \frac{d\mathcal{G}_k(\alpha;\rho)}{d\alpha}$, the step length, α_k, is found by enforcing the strong Wolfe conditions,

$$\mathcal{G}_k(\alpha; \rho) \leq \mathcal{G}_k(0; \rho) + \eta_a \alpha \mathcal{G}'_k(0; \rho) \quad \text{and} \quad |\mathcal{G}'_k(\alpha; \rho)| \leq \eta_w |\mathcal{G}'_k(0; \rho)| \ , \quad (27)$$

where η_a and η_w are pre-assigned constants such that $0 < \eta_a \leq \eta_w < 1$ and $\eta_a < 0.5$. Practical implementations use $\eta_a = 10^{-4}$ and $\eta_w = 0.5$.

Fast line search algorithms use safeguarded polynomial interpolation to find a Wolfe step α_k that satisfies both the above conditions in Eq. (27). Such algorithms are implemented in two stages: the first stage locates an interval $(\alpha_{low}, \alpha_{high})$ that contains a Wolfe step, and the second stage explores this interval using safeguarded polynomial interpolation to find the Wolfe step α_k.

4 Numerical results

The basic SURF algorithm was applied to a notional engineering optimization problem. We optimize the thickness distribution of a bar modeled with a variable number of elements with nonlinear stress-strain behavior. The nonlinear, equality-constrained optimization problem is

$$\begin{aligned} \min_x \quad & F^T d \\ \text{subject to} \quad & V(x) = V_0 \end{aligned} \qquad \text{with} \quad K(x, d)d - F = 0, \quad (28)$$

where d is the displacement vector, F is the force vector, V is the volume function, V_0 is the allowable volume, and K is the function that computes the stiffness matrix.

The problem is solved using reduced-space, full-space, and SURF optimizers. All algorithms use a backtracking line search enforcing the strong Wolfe conditions and a direct solver for the KKT system. For SURF, we use pre-selected inexact solver tolerances. All the preliminary results discussed in this section use a positive-definite BFGS approximation of the Hessian of the Lagrangian, without any of the modifications presented earlier in Sec. 3.3.1. Including the modifications would give a more robust and efficient algorithm.

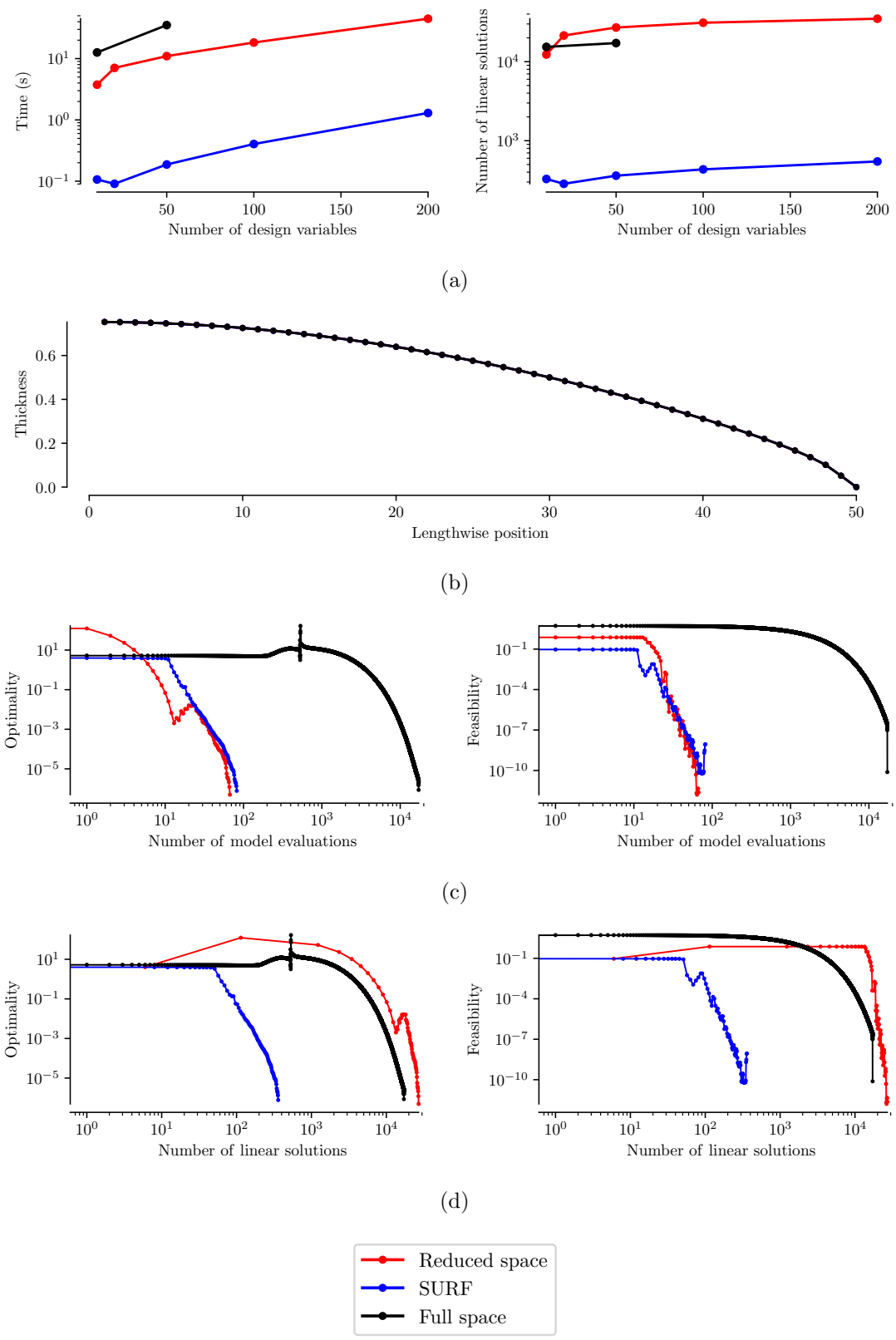


Figure 7: Bar optimization: (a) optimization time across varying number of bar elements, (b) optimized thickness distribution, (c) convergence with model evaluations, and (d) convergence with linear solutions. Note that Figs. 7b-7d show results for a bar modeled with 50 elements.

Fig. 7a shows that SURF with a hybrid formulation is, on average, roughly an order of magnitude more efficient than the RS and FS formulation in time and number of model Jacobian linear solutions across various numbers of bar elements. The full-space formulation for this problem converges only with 10 and 50 bar elements, and it fails for models with 20, 100, and 200 bar elements. This underlines the unreliability of full-space methods, and why reduced-space is still considered the state-of-the-art in LSDO. The full-space formulation can be very efficient for some problems; however, for the bar problem considered here, it is not competitive with the reduced-space method.

Fig. 7b shows the optimized thickness distribution for a bar modeled with 50 elements. Although all three methods converge to the same solution, SURF has a clear edge over the others in terms of the computation time, and the number of linear systems solved, as seen from Fig. 7a.

Fig. 7c and Fig. 7d compare the three methods with regard to convergence. Fig. 7c shows the progress in optimality and feasibility as model evaluations proceed while Fig. 7d characterizes the convergence with respect to the number of linear systems solved.

In Fig. 7c, we see that the number of model evaluations is the lowest for the reduced-space approach since we have exact values for the state variables at each iteration. We should note that the theoretical minimum number of model evaluations needed for convergence is achieved using the RS formulation. However, this can be a misleading metric to assess the computational efficiency of different optimization formulations since the computational cost of each model evaluation depends on the model used in the formulation. A more accurate metric for comparison would be the number of equations or linear systems of similar size solved during an optimization. Fig. 7a reinforces this argument by showing that the computation times and the number of linear solutions are closely related.

In our problem, each nonlinear system is solved iteratively using solutions from a sequence of linear systems. Exact models in the RS method require more linear system solutions to converge whereas inexact models in SURF need a lower number of linear solutions depending on the tolerances imposed on the solver. We see from Fig. 7d that the number of linear solutions required to reach the optimized design is minimum for SURF compared to the other two methods. Therefore, according to this performance metric, we observe that SURF is an order of magnitude more efficient than both its parent methods.

Although it seems that the RS approach is very efficient compared to the FS, Fig. 7d shows that both the methods solve almost the same number of linear systems before reaching a solution. This can also be seen from figure Fig. 7a as both RS and FS takes almost the same amount of time and linear solutions to solve the problem. The slight increase in computation time for the FS approach compared to RS can be attributed to the fact that the linear systems solved in FS are larger than the systems solved in RS. We also see from Fig. 7c and Fig. 7d that the number of model evaluations and the number of linear solutions are the same for an FS formulation as no nonlinear systems are solved during a model evaluation and the only linear system solved is the FS KKT system.

5 Conclusion

This paper presented a new, hybrid architecture for formulating large-scale system design optimization (LSDO) problems. The primary objective is to overcome the limits on computational efficiency that can be realized in conventional architectures. The best optimization algorithms implemented in popular architectures take hundreds of model evaluations to converge to a solu-

tion. The algorithm presented in this paper utilizes an intrusive paradigm to break the barriers on efficiency set by conventional architectures.

A review of the state-of-the-art in LSDO is presented in Sec.2, which includes details on the unified derivatives equation (UDE) and popular architectures for large-scale system design optimization: the reduced-space architecture and the full-space architecture. The reduced-space approach is inefficient but robust, whereas the full-space approach has the potential to be efficient but is not always robust.

In Sec.3, the corrected full-space method (CFS) is introduced, which is the first step toward the unification of the full-space (FS) and reduced-space (RS) methods in an equality-constrained optimization setting. Although the underlying KKT system in the corrected full-space method is the same as that in the full-space method, the corrected full-space method generates the same iterates as the reduced-space method. SURF (strong unification of full-space and reduced-space) is a hybrid algorithm based on the corrected full-space method that can generate all possible hybrids of the full-space and the reduced-space methods. Depending on the tolerances on the inexact solvers, SURF can exhibit the behavior of its parent algorithms to varying degrees. This property of SURF can be exploited, by deriving the best from both the parent methods, to obtain the maximum computational efficiency without compromising on the robustness of the algorithm. Some implementation-specific details of SURF are discussed at the end of Sec.3 which includes a scheme for the BFGS approximation of the Lagrangian Hessian, and a line search method based on an augmented Lagrangian merit function.

Sec.4 provided the results of SURF applied to a bar optimization problem. The numerical results suggest that SURF has the potential to improve the efficiency of the current LSDO algorithms by up to an order of magnitude. It is worth noting that these results were based on pre-selected constant solver tolerances and an even better efficiency could be achieved if we can compute optimal tolerances for each iteration.

SURF provides a way to generate any hybrid of the full-space and reduced-space methods but currently there is no mechanism for identifying the precise hybrid that offers the best computational efficiency. Strategies to compute optimal tolerances are needed, and these tolerances must be selected adaptively from one iteration to the next for extracting the maximum efficiency. Although the unification is complete for an equality-constrained setting, most problems in LSDO are inequality-constrained problems. Extending the present algorithm for general inequality-constrained optimization problems and formulating a strategy for adaptive hybrid selection are potential areas for further research.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1917142. The first author was also supported by the First Year Fellowship from the Department of Aerospace and Mechanical Engineering at the University of California San Diego. The authors would like to thank Philip E. Gill for his valuable suggestions on implementation with sequential quadratic programming.

References

[1] John T. Hwang and Joaquim R. R. A. Martins. "A Computational Architecture for Coupling Heterogeneous Numerical Models and Computing Coupled Derivatives". In: *ACM Trans. Math. Softw.* 44.4 (June 2018), 37:1–37:39. ISSN: 0098-3500. DOI: 10.1145/3182393.

[2] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization". In: *Structural and Multidisciplinary Optimization* 59.4 (Apr. 2019), pp. 1075–1104. DOI: 10.1007/s00158-019-02211-z.

[3] George Biros and Omar Ghattas. "Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver". In: *SIAM Journal on Scientific Computing* 27.2 (2005), pp. 687–713. DOI: https://doi.org/10.1137/S106482750241565X.

[4] George Biros and Omar Ghattas. "Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part II: The Lagrange–Newton solver and its application to optimal control of steady viscous flows". In: *SIAM Journal on Scientific Computing* 27.2 (2005), pp. 714–739. DOI: https://doi.org/10.1137/S1064827502415661.

[5] John Hwang and Joaquim Martins. "Allocation-mission-design optimization of next-generation aircraft using a parallel computational framework". In: *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2016. DOI: 10.2514/6.2016-1662.

[6] John T Hwang and Andrew Ning. "Large-scale multidisciplinary optimization of an electric aircraft for on-demand mobility". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018. DOI: 10.2514/6.2018-1384.

[7] John T Hwang, Dae Young Lee, James W Cutler, and Joaquim R R A Martins. "Large-scale multidisciplinary optimization of a small satellite's design and operation". In: *Journal of Spacecraft and Rockets* 51.5 (2014), pp. 1648–1663. DOI: 10.2514/1.A32751.

[8] Hayoung Chung, John T Hwang, Justin S Gray, and Hyunsun A Kim. "Implementation of topology optimization using OpenMDAO". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018. DOI: 10.2514/6.2018-0653.

[9] John P Jasa, John T Hwang, and Joaquim Martins. "Design and Trajectory Optimization of a Morphing Wing Aircraft". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018. DOI: 10.2514/6.2018-1382.

[10] John P. Jasa, John T. Hwang, and Joaquim R. R. A. Martins. "Open-source coupled aerostructural optimization using Python". In: *Structural and Multidisciplinary Optimization* 57.4 (Apr. 2018), pp. 1815–1827. ISSN: 1615-1488. DOI: 10.1007/s00158-018-1912-8.

[11] Tae Hyun Ha, Keunseok Lee, and John T Hwang. "Large-scale design and economics optimization of eVTOL concepts for urban air mobility". In: *2019 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2019. DOI: 10.2514/6.2019-1218.

[12] John T Hwang, Arnav V Jain, and Tae H Ha. "Large-scale multidisciplinary design optimization—review and recommendations". In: *AIAA Aviation 2019 Forum*. 2019. DOI: 10.2514/6.2019-3106.

[13] Joaquim R R A Martins and John T Hwang. "Review and unification of methods for computing derivatives of multidisciplinary computational models". In: *AIAA journal* 51.11 (2013), pp. 2582–2599. DOI: 10.2514/1.J052184.

[14] John P Jasa, Charles A Mader, and Joaquim Martins. "Trajectory Optimization of a Supersonic Aircraft with a Thermal Fuel Management System". In: *2018 Multidisciplinary Analysis and Optimization Conference*. 2018, p. 3884. DOI: https://doi.org/10.2514/6.2018-3884.

[15] Sam Friedman, Seyede Fatemeh Ghoreishi, and Douglas L Allaire. "Quantifying the Impact of Different Model Discrepancy Formulations in Coupled Multidisciplinary Systems". In: *19th AIAA non-deterministic approaches conference*. 2017, p. 1950. DOI: https://doi.org/10.2514/6.2017-1950.

[16] Justin S Gray, Tristan A Hearn, Kenneth T Moore, John Hwang, Joaquim Martins, and Andrew Ning. "Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO". In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2014. DOI: 10.2514/6.2014-2042.

[17] Satadru Roy, Kenneth Moore, John T Hwang, Justin S Gray, William A Crossley, and Joaquim Martins. "A mixed integer efficient global optimization algorithm for the simultaneous aircraft allocation-mission-design problem". In: *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2017. DOI: 10.2514/6.2017-1305.

[18] Satadru Roy, William A Crossley, Kenneth T Moore, Justin S Gray, and Joaquim Martins. "Next generation aircraft design considering airline operations and economics". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018, p. 1647. DOI: https://doi.org/10.2514/6.2018-1647.

[19] John T. Hwang, John P. Jasa, and Joaquim R. R. A. Martins. "High-fidelity design-allocation optimization of a commercial aircraft maximizing airline profit". In: *Journal of Aircraft* (2019), pp. 1–15. DOI: 10.2514/1.C035082.

[20] Tristan A Hearn, Eric Hendricks, Jeffrey Chin, and Justin S Gray. "Optimization of turbine engine cycle analysis with analytic derivatives". In: *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2016, p. 4297. DOI: https://doi.org/10.2514/6.2016-4297.

[21] Justin Gray, Jeffrey Chin, Tristan Hearn, Eric Hendricks, Thomas Lavelle, and Joaquim RRA Martins. "Chemical-Equilibrium Analysis with Adjoint Derivatives for Propulsion Cycle Analysis". In: *Journal of Propulsion and Power* 33.5 (2017), pp. 1041–1052. DOI: https://doi.org/10.2514/1.B36215.

[22] Justin S Gray, Gaetan K Kenway, Charles A Mader, and Joaquim Martins. "Aero-propulsive Design Optimization of a Turboelectric Boundary Layer Ingestion Propulsion System". In: *2018 Aviation Technology, Integration, and Operations Conference*. 2018, p. 3976. DOI: https://doi.org/10.2514/6.2018-3976.

[23] Justin Gray. "Design Optimization of a Boundary Layer Ingestion Propulsor Using a Coupled Aeropropulsive Model". PhD thesis. 2018. DOI: http://hdl.handle.net/2027.42/147625.

[24] Justin S Gray and Joaquim RRA Martins. "Coupled aeropropulsive design optimisation of a boundary-layer ingestion propulsor". In: *The Aeronautical Journal* 123.1259 (2019), pp. 121–137. DOI: https://doi.org/10.1017/aer.2018.120.

[25] Ryan Barrett and Andrew Ning. "Integrated free-form method for aerostructural optimization of wind turbine blades". In: *Wind Energy* 21.8 (2018), pp. 663–675. DOI: https://doi.org/10.1002/we.2186.

[26] Frederik Zahle, Carlo Tibaldi, Christian Pavese, Michael K McWilliam, Jose PAA Blasques, and Morten H Hansen. "Design of an aeroelastically tailored 10 MW wind turbine rotor". In: *Journal of Physics: Conference Series*. Vol. 753. 6. IOP Publishing. 2016, p. 062008. DOI: https://doi.org/10.1088/1742-6596/753/6/062008.

[27] Frederik Zahle, Niels N Sørensen, Michael K McWilliam, and Athanasios Barlas. "Computational fluid dynamics-based surrogate optimization of a wind turbine blade tip extension for maximising energy production". In: *Journal of Physics: Conference Series*. Vol. 1037. 4. IOP Publishing. 2018, p. 042013. DOI: https://doi.org/10.1088/1742-6596/1037/4/042013.

[28] Michael K McWilliam, Frederik Zahle, Antariksh Dicholkar, David Verelst, and Taeseong Kim. "Optimal aero-elastic design of a rotor with bend-twist coupling". In: *Journal of Physics: Conference Series*. Vol. 1037. 4. IOP Publishing. 2018, p. 042009. DOI: https://doi.org/10.1088/1742-6596/1037/4/042009.

[29] Peter Graf, Katherine Dykes, Rick Damiani, Jason Jonkman, and Paul Veers. "Adaptive stratified importance sampling: hybridization of extrapolation and importance sampling Monte Carlo methods for estimation of wind turbine extreme loads". In: *Wind Energy Science (Online)* 3.NREL/JA-5000-72435 (2018). DOI: https://doi.org/10.5194/wes-3-475-2018.

[30] Robert D Falck, Jeffrey Chin, Sydney L Schnulo, Jonathan M Burt, and Justin S Gray. "Trajectory optimization of electric aircraft subject to subsystem thermal constraints". In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2017, p. 4002. DOI: https://doi.org/10.2514/6.2017-4002.

[31] Sydney L Schnulo, Jeff Chin, Robert D Falck, Justin S Gray, Kurt V Papathakis, Sean C Clarke, Nickelle Reid, and Nicholas K Borer. "Development of a multi-segment mission planning tool for SCEPTOR X-57". In: *2018 Multidisciplinary Analysis and Optimization Conference*. 2018, p. 3738. DOI: https://doi.org/10.2514/6.2018-3738.

[32] Eric S Hendricks, Robert D Falck, and Justin S Gray. "Simultaneous propulsion system and trajectory optimization". In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2017, p. 4435. DOI: https://doi.org/10.2514/6.2017-4435.

[33] Haijian Yang, Feng-Nan Hwang, and Xiao-Chuan Cai. "Nonlinear Preconditioning Techniques for Full-Space Lagrange–Newton Solution of PDE-Constrained Optimization Problems". In: *SIAM Journal on Scientific Computing* 38.5 (2016), A2756–A2778. DOI: https://doi.org/10.1137/15M104075X.

[34] Michael J. D. Powell. "The convergence of variable metric methods for nonlinearly constrained optimization calculations". In: *Nonlinear Programming, 3 (Proc. Sympos., Special Interest Group Math. Programming, Univ. Wisconsin, Madison, Wis., 1977)*. New York: Academic Press, 1978, pp. 27–63. ISBN: 0-12-468660-3. DOI: https://doi.org/10.1016/B978-0-12-468660-1.50007-4.