An SQP algorithm based on a hybrid architecture for accelerating optimization of large-scale systems

Anugrah Jo Joshy*, Ryan Dunn†, Mark Sperry‡, Victor Gandarillas§, and John T. Hwang¶
University of California San Diego, La Jolla, CA, 92093

The optimization of large-scale and multidisciplinary engineering systems is now prevalent in many fields, exemplified in areas such as aircraft, satellite, and wind turbine design. The rise of advanced modeling frameworks has expanded the scope of large-scale optimization techniques across various research domains. However, novel applications have emerged that pose efficiency-related computational challenges to the existing methods employed in large-scale, gradient-based optimization. The authors previously proposed a new paradigm for accelerating the optimization of large-scale and complex-engineered systems, laying the groundwork for a new approach. The new paradigm is based on a hybrid optimization architecture called SURF which stands for strong unification of reduced-space and full-space. SURF has the potential to expedite optimization of models with state variables that are iteratively computed by solving nonlinear systems. This paper extends the existing paradigm by providing new theoretical results that unify the reduced-space and full-space algorithms in a practical optimization setting that considers line searches and quasi-Newton methods. We also present a practical, SQP-based SURF algorithm that can be applied to general, inequality-constrained problems. The new algorithm also includes an adaptive hybrid selection strategy for robust convergence and faster solutions. We test the new algorithm on a low-fidelity motor optimization problem and a wind farm layout optimization problem to validate the optimization results, and to demonstrate its computational benefits. In one of the problems, SURF was able to speed up the traditional optimization by approximately 25 percent. In the other problem, SURF was able to converge to a better optimal solution.

I. Nomenclature

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

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

II. Introduction

Design optimization is the computation of a vector of optimal design variables x that minimize or maximize a given objective $F(x)$ while adhering to a vector of specified constraints $C(x)$. A typical design optimization workflow (shown in Fig. 1) involves a general-purpose optimization algorithm (also known as optimizer) iteratively evaluating a computational model until a design that fulfills the specified optimality is identified. The primary function of the model is to compute the objective, constraints, and their gradients with respect to the design variables, and forward them to the optimizer. The development of large-scale and multidisciplinary system models is typically facilitated by optimization-targeted modeling frameworks that also provide interfaces to various optimization algorithms.

*PhD Candidate, Department of Mechanical and Aerospace Engineering, AIAA Student Member
†MS Student, Department of Mechanical and Aerospace Engineering, AIAA Student Member
‡PhD Student, Department of Mechanical and Aerospace Engineering, AIAA Student Member
§PhD Candidate, Department of Mechanical and Aerospace Engineering, AIAA Student Member
¶Assistant Professor, Department of Mechanical and Aerospace Engineering, and AIAA Member

The application of system-level optimization in engineering design is on the rise. Many engineering systems are large-scale in nature with tens to thousands of design degrees of freedom, and this makes gradient-based algorithms the only viable approach for optimizing them. Moreover, large-scale systems are seldom restricted to a single field, and the inherent multidisciplinary nature of these systems often entails complex numerical modeling for gradient-based optimization. This often discourages new users from exploiting the benefits of large-scale system design optimization (LSDO). Recent enhancements in the usability of modeling frameworks have lowered the barrier-to-entry for new users, leading to an increase in the applications for LSDO over the last few years. With the expanding user base and emergence of new applications, the current optimization algorithms are faced with new challenges, and one of them being their computational efficiency.

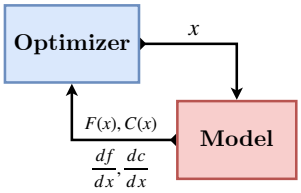


Fig. 1 Current optimization workflow [1]

Numerical optimization algorithms that are agnostic to specific models, commonly known as general-purpose optimization algorithms, have been in use for several decades. The sequential quadratic programming (SQP) and interior point (IP) methods are among the most effective optimization algorithms for large-scale optimization, capable of solving optimization problems with thousands of design variables in only hundreds or fewer model evaluations. These algorithms have a very high level of computational efficiency, which has nearly reached a saturation point over the last decade. This suggests that substantial reductions in optimization time can only be achieved by constructing optimizers tailored to specific applications. This directly follows from the no free lunch (NFL) theorems for optimization which establishes that "for any algorithm, any elevated performance over one class of problems is offset by performance over another class" [2]. Efficiency improvements could be achieved by developing optimization algorithms that are specialized to exploit the structure of specific classes of problems [3].

While there has been little progress in computational efficiency with modern general-purpose optimization algorithms for LSDO in the past decade, optimization frameworks developed by engineering practitioners have made significant strides. This is attested by the automation of efficient total derivative computation for complex multidisciplinary models by the MAUD (modular analysis and unified derivatives) [4] architecture, and its successful implementation in NASA's OpenMDAO [5] framework for multidisciplinary design, analysis and optimization. Another notable example is the Computational System Design Language (CSDL) [6] which automates adjoint-based sensitivity analysis using a graph-based methodology. These rapid pace of developments in optimization-targeted modeling environments have opened up new avenues for developing specialized optimization algorithms through innovations in a combined modeling and optimization paradigm.

One such innovation is enabling robust optimization convergence in the presence of partially converged model evaluations. This could provide potential speedups for the class of optimization problems with models having expensive nonlinear solvers. The speedups can be achieved by evaluating the model at high accuracy only when required, thereby effectively reducing the overall cost of model evaluations via adaptive adjustments in model fidelity as directed by an optimization algorithm. However, this moves away from the conventional rigid optimization architecture shown in Fig. 1 and necessitates the development of a new, flexible architecture capable of optimization with partial model evaluations. We also need modeling frameworks that are capable of facilitating such an architecture.

Joshy and Hwang [1] had earlier proposed a hybrid architecture that has the potential to reduce overall optimization times for large-scale systems with expensive nonlinear systems by utilizing an intrusive paradigm combining modeling and optimization. Based on this intrusive paradigm, a new algorithm called SURF (strong unification of reduced-space and full-space) was shown to accelerate the optimization by an order of magnitude for an equality-constrained optimization problem. This algorithm was further extended as a hypothesis [7] for general, inequality-constrained optimization, and the validity of the paradigm was verified experimentally on a nonlinear topology optimization problem.

SURF is a hybrid of two multidisciplinary optimization (MDO) architectures popularly known as the multidisciplinary feasible architecture (MDF), and the simultaneous analysis and design architecture (SAND). SURF provides the capability to select a hybrid of the two architectures by adjusting the tolerances on the nonlinear systems within the computational

models. Previous investigations on SURF relied primarily on the theoretical unification of the two architectures for a simplified, equality-constrained optimization case which disregards some of the practical components of an optimization algorithm.

In this paper, we address this deficiency and present new theoretical results on the unification in the presence of line searches that impose the Armijo or strong Wolfe conditions, as well as Hessian approximations that enforce the quasi-Newton condition. We also present the theoretical unification for general-inequality constrained optimization that was only conjectured previously. Furthermore, we provide other essential details for a complete practical implementation of the hybrid algorithm with sequential quadratic programming (SQP), and a strategy for adaptively selecting solver tolerances for ensuring convergence.

We apply the complete, SQP-based SURF algorithm to two distinct engineering design optimization problems. The first problem deals with the optimization of a permanent magnet synchronous motor (PMSM) for maximizing its efficiency by tuning the sizing parameters. The second problem concerns the optimization of a wind farm layout with the objective of maximizing the annual energy production (AEP) through the optimal placement of wind turbines.

The remainder of the paper is structured as follows. In Sec. III, we discuss the two parent architectures for the SURF algorithm, which we call the reduced-space and full-space architectures. We also discuss the prior unification results on SURF in this section. In Sec. IV, we present the new theoretical results along with a complete, practical SQP algorithm implementing SURF. We also discuss an adaptive hybrid selection method in this section. In Sec. V, we solve two optimization problems using the new SURF approach and the conventional approaches, and discuss the results. In Sec. VI, we conclude by summarizing the work, and identifying directions for future research.

III. Background

A. Optimization problem statement

A general optimization problem with models having implicitly computed state variables can be written as

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

where $x \in \mathbb{R}^n$ are 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 $C : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^m$ 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. We define $y \in \mathbb{R}^r$ as the vector of *state variables*, $f \in \mathbb{R}$ as the *objective*, and $c \in \mathbb{R}^m$ as the vector of *constraint variables*. Design variables are the parameters that define the design space and are often referred to as optimization variables or decision variables depending on the context in which they are used.

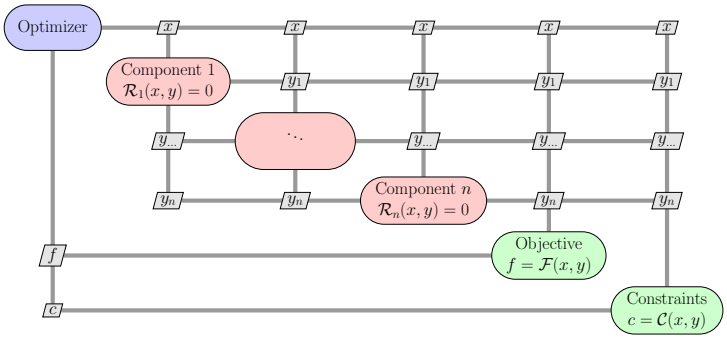
B. Reduced-space and full-space formulations

Within the domain of multidisciplinary design optimization (MDO), optimization formulations or architectures [8] are broadly categorized as either monolithic or distributed. Monolithic architectures solve a single system-level optimization problem whereas distributed architectures decompose the the system-level optimization problem into suboptimization problems and solve them sequentially to obtain a solution for the overarching optimization problem. Distributed architectures often fail to capture the coupling between different subdisciplines within the system and result in suboptimal designs. For this reason, monolithic architectures are recommended whenever possible to solve large-scale and multidisciplinary optimization problems.

Our work in this paper focuses on the unification of two widely-used monolithic MDO architectures: the reduced-space (RS) architecture, and the full-space (FS) architecture. The RS architecture is also known as the multidisciplinary feasible (MDF) architecture [9] or nested analysis and design (NAND) [10, 11], and the FS architecture is also known as the simultaneous analysis and design (SAND) [12] architecture. In a reduced-space architecture, the nonlinear systems within each subdiscipline are solved to the tightest tolerances possible during each model evaluation to compute the exact state variable values that are used elsewhere in the model. However, the full-space architecture treats the residual equations that define the nonlinear systems as additional constraints and the state variables in these nonlinear systems as additional design variables.

Reduced-space

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x, \mathcal{Y}(x)) \\ \text{s.t.} \quad & C(x, \mathcal{Y}(x)) \geq 0 \\ \text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0 \end{aligned} \tag{2}$$



Full-space

$$\begin{aligned} \min_{x,y} \quad & \mathcal{F}(x, y) \\ \text{s.t.} \quad & C(x, y) \geq 0 \\ & \mathcal{R}(x, y) = 0 \end{aligned} \tag{3}$$

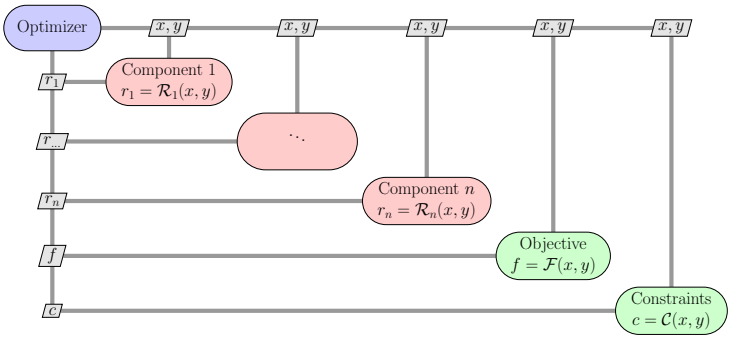


Fig. 2 Reduced-space versus full-space formulation for a model containing implicit state variables [1].

The difference between the formulations for the RS and FS architectures is shown in Fig. 2. The reduced-space formulation solves $\mathcal{R}(x, \mathcal{Y}(x)) = 0$ implicitly inside the model to compute the exact state variables $y = \mathcal{Y}(x))$ as a function of the design variables x. The full-space formulation, on the other hand, treats the implicit state variables y as additional design variables and the residuals $\mathcal{R}(x, y) = 0$ as additional constraints thereby ensuring that y at an optimally converged solution solves the nonlinear systems. The nonlinear solutions in the RS model evaluation can make it significantly more expensive than the FS model evaluation. However, the RS optimization problem can be significantly smaller than the FS optimization problem which contains additional design variables and constraints. The nomenclature of the two formulations stems from their respective design space dimensions: the "full space" signifies the $(n + r)$-dimensional design space encompassing both variables x and y, while the "reduced space" corresponds to the n-dimensional space consisting only variable x.

The FS approach can be beneficial for certain problems where it converges faster than the RS approach. However, the full-space approach is more prone to convergence issues and is therefore less robust. Additionally, a full-space approach is not applicable for models containing nonlinear systems that are not solvable using Newton's method, for example, nonlinear systems whose solutions require a bracketing method. Therefore, the more robust reduced-space approach is always recommended for LSDO problems. However, the RS formulation can be highly inefficient since it has to ensure the tightest tolerances possible when solving nonlinear systems to ensure that the derivatives are consistent at a given design variable value x.

We proceed by examining the optimality conditions and Karush-Kuhn-Tucker (KKT) systems for the two formulations in an equality-constrained optimization setting. The design variable values satisfying the optimality conditions are obtained by solving the KKT systems iteratively (shown in algorithms 1, and 2).

Reduced-space equations

Starting with Problem (2) with only equality constraints, we define the Lagrangian $l = \mathcal{L}(x, \lambda)$ where $\mathcal{L}(x, \lambda) = \mathcal{F}(x, \mathcal{Y}(x)) + \lambda^T C(x, \mathcal{Y}(x))$. The first order necessary optimality conditions for this problem are

$$\begin{aligned} \frac{\mathrm{d}l}{\mathrm{d}x} &= \left(\frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y} \frac{\partial R}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x} \right) + \lambda^T \left(\frac{\partial C}{\partial x} - \frac{\partial C}{\partial y} \frac{\partial \mathcal{R}}{\partial y}^{-1} \frac{\partial \mathcal{R}}{\partial x} \right) = 0 \\ \frac{\mathrm{d}l}{\mathrm{d}\lambda} &= C(x, \mathcal{Y}(x)) = 0. \end{aligned} \tag{4}$$

Applying the method of Newton-Lagrange, we obtain the following KKT system for reduced-space optimization:

$$\begin{bmatrix} l_{xx} & {c_x}^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -l_x \\ -C(x_k, \mathcal{Y}(x_k)) \end{bmatrix}, \tag{5}$$

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

Full-space equations

Starting with Problem (3) with only equality constraints, we define the Lagrangian $m = \mathcal{M}(x, y, \psi, \lambda)$ where $\mathcal{M}(x, y, \psi, \lambda) = \mathcal{F}(x, y) + \psi^T \mathcal{R}(x, y) + \lambda^T C(x, y)$. The first order necessary optimality conditions for this problem are

$$\begin{aligned} \frac{\mathrm{d}m}{\mathrm{d}x} &= \frac{\partial \mathcal{F}}{\partial x} + \psi^T \frac{\partial \mathcal{R}}{\partial x} + \lambda^T \frac{\partial C}{\partial x} = 0 \qquad & \frac{\mathrm{d}m}{\mathrm{d}\psi} &= \mathcal{R}(x, y) = 0 \\ \frac{\mathrm{d}m}{\mathrm{d}y} &= \frac{\partial \mathcal{F}}{\partial y} + \psi^T \frac{\partial \mathcal{R}}{\partial y} + \lambda^T \frac{\partial C}{\partial y} = 0 \qquad & \frac{\mathrm{d}m}{\mathrm{d}\lambda} &= C(x, y) = 0. \end{aligned} \tag{6}$$

Applying the method of Newton-Lagrange, we obtain the following KKT system for full-space optimization:

$$\begin{bmatrix} m_{xx} & m_{xy} & {\mathcal{R}_x}^T & {C_x}^T \\ m_{yx} & m_{yy} & {\mathcal{R}_y}^T & {C_y}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ p_k^{(\psi)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k, y_k) \\ -C(x_k, y_k) \end{bmatrix}, \tag{7}$$

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

C. The modified full-space method [1]

Here we describe the modified full-space (MFS) method proposed in [1] for an equality-constrained setting. The MFS method is derived from the conventional FS method by incorporating two updates. During each optimization iteration of the MFS method, the state variables y and the vector of Lagrange multipliers ψ corresponding to the nonlinear residuals are updated before solving the full-space KKT system (7). The optimization variables $[x_k, y_k, \psi_k, \lambda_k]^T$ upon completion of the kth iteration are updated to $[x_k, y'_k, \psi'_k, \lambda_k]^T$ at the beginning of the $(k+1)$th iteration. The state variables y'_k are obtained by solving $\mathcal{R}(x_k, y'_k) = 0$ and the associated Lagrange multipliers ψ'_k by setting $dm/dy = 0$. The FS KKT system is then solved at $[x_k, y'_k, \psi'_k, \lambda_k]^T$ for obtaining the search direction p_k toward the next iterate. The following theorem proves that with these two modifications, the MFS method can generate the same iterates (x_k, λ_k) as the RS method.

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 modified full-space method are identical in an equality-constrained optimization setting.*

MFS as an architecture is shown in Fig. 3. A comparison of the algorithms for the three architectures (see algorithms 1, 2, 3) is also given below. MFS bridges the RS and FS architectures as it generates RS iterates using an underlying FS architecture. However, MFS is only the first step toward a hybrid architecture unifying the FS and RS architectures since we still cannot access hybrids of FS and RS.

It is straightforward to see that by not applying the two updates on y_k and ψ_k, MFS reverts back to the FS method. This can be seen as applying the nonlinear solver in line 2 and the linear solver in line 3 of the MFS algorithm 3 with an infinite tolerance. This suggests that by relaxing the tolerance on these solvers, the MFS algorithm moves closer toward the FS algorithm. Since the standard MFS method is equivalent to the RS method, we can get any hybrids of the FS and RS by applying inexact tolerances on the two solvers in the MFS method. The SURF (strong unification of reduced-space and full-space) architecture proposed in [1] results from the introduction of this inexactness into the MFS architecture. Therefore, SURF provides a complete theoretical unification of the RS and FS architectures as we can now select one of the two architectures or any of its hybrids by just varying the inexact solver tolerances.

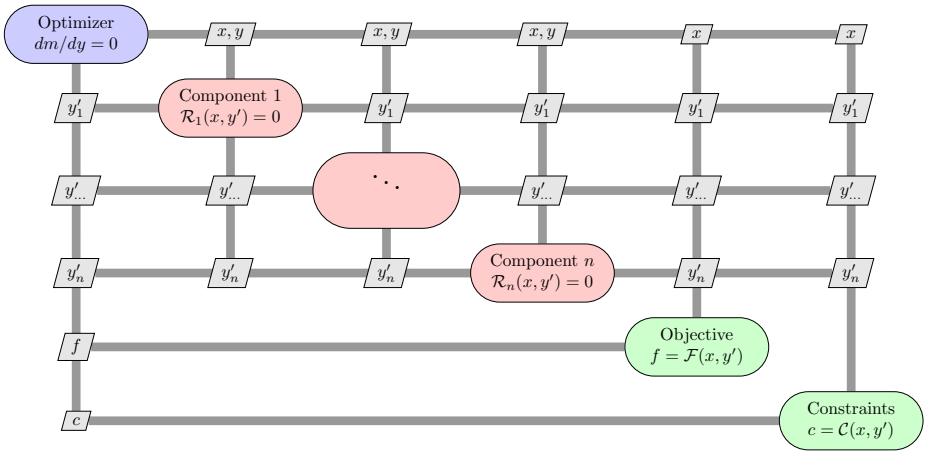


Fig. 3 Modified full-space architecture [1].

The reduced-space, full-space and modified full-space methods [1]:

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**

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

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

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**

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

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

Algorithm 3 MFS 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 $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - C_y^T \lambda_k$
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**

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

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

D. Notation:

For conciseness, we use the following notation for the rest of this paper. Note that the corresponding formulation (RS, FS, or MFS) needs to be interpreted from the context.

- $z = (x, y)$ or $z = x$ refers to the vector of design variables,
- $\pi = (\psi, \lambda)$ or $\pi = \lambda$ refers to the vector of Lagrange multipliers, and
- $v = (x, y, \psi, \lambda)$ or $v = (x, \lambda)$ refers to the concatenated vector of design variables and Lagrange multipliers.

IV. Methodology

In this section, we present our new theoretical findings that generalize the SURF algorithm for a practical optimization setting. We also discuss a scheme for selecting inexact tolerances for the nonlinear solvers.

A. Line searches

Line searches are an essential part of many optimization algorithms to ensure global convergence. Although there exist multiple merit functions that can be used to check the acceptability of a step length $\alpha \in (0, 1]$ along the predicted direction p_k, only smooth merit functions allow for fast line searches that enforce the strong Wolfe conditions. Therefore, the results that follow consider the smooth augmented Lagrangian merit function.

The augmented Lagrangian for RS and MFS are respectively

$$L_A(x, \lambda; \rho) = \mathcal{F}(x, \mathcal{Y}(x)) + \lambda^T C(x, \mathcal{Y}(x)) + \frac{1}{2}\rho(C(x, \mathcal{Y}(x))^T C(x, \mathcal{Y}(x))), \text{ and} \tag{8}$$

$$L_A(x, y, \psi, \lambda; \rho) = \mathcal{F}(x, y) + \psi^T \mathcal{R}(x, y) + \lambda^T C(x, y) + \frac{1}{2}\rho(C(x, y)^T C(x, y) + \mathcal{R}(x, y)^T \mathcal{R}(x, y)), \tag{9}$$

where ρ is the penalty parameter that controls the penalization for constraint violations. After obtaining the search direction p_k and an updated ρ, the merit function as a function of step length α can be written as

$$\phi_{RS}(\alpha) = L_A(x_k + \alpha p_k^{(x)},\ \lambda_k + \alpha p_k^{(\lambda)}; \rho), \text{ and} \tag{10}$$

$$\phi_{MFS}(\alpha) = L_A(x_k + \alpha p_k^{(x)},\ y'_k + \alpha p_k^{(y)},\ \psi'_k + \alpha p_k^{(\psi)},\ \lambda_k + \alpha p_k^{(\lambda)}; \rho). \tag{11}$$

For fast line searches, an acceptable step length α satisfies the strong Wolfe conditions

$$\phi(\alpha) \leq \phi(0) + \eta_A \alpha \phi'(0) \quad \text{and} \quad |\phi'(\alpha)| \leq \eta_W |\phi'(0)|\,, \tag{12}$$

where η_A and η_W are constants such that $0 < \eta_A \leq \eta_W < 1$ and $\eta_A < 0.5$. Typical values used are $\eta_A = 10^{-4}$ and $\eta_W = 0.5$. We refer to the first and second conditions as the Armijo condition and the Wolfe condition respectively.

The following theorem proves the equivalence of the Armijo condition for the RS and MFS methods. Note that model evaluations during an MFS line search solves $\mathcal{R}(x, y) = 0$. The theorem follows from showing $\phi_{RS}(\alpha) = \phi_{MFS}(\alpha)$, and $\phi'_{RS}(0) = \phi'_{MFS}(0)$.

Theorem 2 (Armijo condition)**.** *At a given (x_k, λ_k), the step lengths α along p_k satisfying the Armijo condition in a line search using an augmented Lagrangian merit function are identical for the reduced-space and modified full-space methods, in an equality-constrained optimization setting.*

Merely solving $\mathcal{R}(x, y) = 0$ in the MFS line search does not result in the equivalence for the Wolfe condition. We additionally require solving a new linear system $\mathcal{R}_y^T \psi_\alpha = -(\mathcal{F}_y^T + C_y^T \lambda + \rho C_y^T C)$ at each α to redefine the Lagrange multiplier from its value of $\psi'_k + \alpha p_k^{(\psi)}$. Note that this essentially decouples ψ from the line search. However, this redefinition results in $\phi'_{RS}(\alpha) = \phi'_{MFS}(\alpha)$, leading to the following theorem.

Theorem 3 (Wolfe condition)**.** *At a given (x_k, λ_k), the step lengths α along p_k satisfying the strong Wolfe conditions in a line search using an augmented Lagrangian merit function are identical for the reduced-space and modified full-space methods, in an equality-constrained optimization setting.*

B. Quasi-Newton Hessian updates

In many large-scale problems, it is impractical to obtain exact Hessians either due to its high computational cost or the extensive effort required to derive and implement the second derivatives. Most general-purpose algorithms approximate Hessians, and quasi-Newton approximations are a widely-used class of methods that enforce the quasi-Newton condition $\widehat{H}_{k+1} d_k = w_k$, where $\widehat{H}_{k+1}$ is the Hessian approximated after the kth iteration, $d_k = z_{k+1} - z_k = \alpha_k p_k^{(z)}$, and $w_k = \nabla_z L(z_{k+1}, \pi_{k+1}) - \nabla_z L(z_k, \pi_{k+1})$, where ∇L is the gradient of the Lagrangian. Enforcing the quasi-Newton condition incorporates the approximate curvature of the exact Hessian H_{k+1} along d_k into $\widehat{H}_{k+1}$. The approximate curvature along d_k is given by $w_k^T d_k$. The next result shows that the curvature $w_k^T d_k$ incorporated into the approximate Hessian is the same for RS and MFS if $d_k = \alpha_k p_k^{(z)}$ in MFS. The result follows from the MFS equations $\mathcal{R}_y^T \psi'_{k+1} = -\mathcal{F}_y^T - C_y^T \lambda_{k+1}$ and $p_k^{(y)} = \dfrac{\mathrm{d}y}{\mathrm{d}x} p_k^{(x)}$.

Theorem 4 (Quasi-Newton condition)**.** *At the end of the kth iteration, the approximate curvature $w_k^T d_k$ along $d_k = \alpha_k p_k^{(z)}$ for the reduced-space and modified full-space methods are identical in an equality-constrained optimization setting.*

This result has a major implication for positive-definite quasi-Newton Hessian updates such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method. BFGS method is the most widely-used Hessian approximation in practice due to its consistent superior performance over other methods. The BFGS update formula is given by

$$\widehat{H}_{k+1} = \widehat{H}_k - \frac{1}{d_k^T \widehat{H}_k d_k} \widehat{H}_k d_k d_k^T \widehat{H}_k + \frac{1}{w_k^T d_k} w_k w_k^T. \tag{13}$$

For a positive definite $\widehat{H}_k$, the $\widehat{H}_{k+1}$ given by the BFGS formula is positive definite if and only if $w_k^T d_k > 0$. This fact along with the previous theorem implies the following corollary.

Corollary 4.1 (Positive-definiteness of BFGS)**.** *At a given (x_k, λ_k), the updated BFGS Hessian for the RS method is positive definite if and only if the updated BFGS Hessian for the MFS method is positive definite.*

Altogether, theorem 1 in conjunction with theorems 2, 3, 4, and 4.1 prove the equivalence of the RS and MFS methods in a practical equality-constrained optimization setting with line searches and quasi-Newton Hessian approximations.

C. Inequality-constrained optimization

This subsection presents a result that extends the equivalence between RS and MFS methods to inequality-constrained problems when following a sequential quadratic programming (SQP) approach. The SQP algorithm for an inequality-constrained optimization problem solves a sequence of inequality-constrained quadratic programming (QP) subproblems to obtain the search direction toward the next iterate. Each QP subproblem minimizes a quadratic approximation of the objective subject to linearized constraints. Note that the quadratic approximation is based on the Hessian of the Lagrangian rather than the Hessian of the objective. The modified full-space QP subproblem for the $(k+1)$th iteration can be stated as

$$\begin{aligned} \min_{x,y} \quad & \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial x} & \frac{\partial \mathcal{F}}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y'_k \end{bmatrix} + \begin{bmatrix} (x - x_k)^T & (y - y'_k)^T \end{bmatrix} \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y'_k \end{bmatrix} \\ \text{s.t.} \quad & C(x_k, y'_k) + \begin{bmatrix} \frac{\partial C}{\partial x} & \frac{\partial C}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y'_k \end{bmatrix} \geq 0, \\ & \begin{bmatrix} \frac{\partial \mathcal{R}}{\partial x} & \frac{\partial \mathcal{R}}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y'_k \end{bmatrix} = 0 \end{aligned} \tag{14}$$

where

$$\begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 \mathcal{F}}{\partial x^2} & \frac{\partial^2 \mathcal{F}}{\partial y \partial x} \\ \frac{\partial^2 \mathcal{F}}{\partial x \partial y} & \frac{\partial^2 \mathcal{F}}{\partial y^2} \end{bmatrix} + \sum_{i=1}^{m} [\lambda_k]_i \begin{bmatrix} \frac{\partial^2 C_i}{\partial x^2} & \frac{\partial^2 C_i}{\partial y \partial x} \\ \frac{\partial^2 C_i}{\partial x \partial y} & \frac{\partial^2 C_i}{\partial y^2} \end{bmatrix} + \sum_{i=1}^{r} [\psi'_k]_i \begin{bmatrix} \frac{\partial^2 \mathcal{R}_i}{\partial x^2} & \frac{\partial^2 \mathcal{R}_i}{\partial y \partial x} \\ \frac{\partial^2 \mathcal{R}_i}{\partial x \partial y} & \frac{\partial^2 \mathcal{R}_i}{\partial y^2} \end{bmatrix} \tag{15}$$

is the Lagrangian Hessian at $[x_k, y'_k, \psi'_k, \lambda_k]^T$. Notice that the modified QP subproblem differs from the full-space QP subproblem only by the updates on y_k and ψ_k using exact solutions from lines 2 and 3 in Algorithm. 3.

Theorem 5 (Inequality-constrained SQP)**.** *At a given (x_k, λ_k), the solution (x_{k+1}, λ_{k+1}) provided by the modified full-space QP subproblem is also a solution to the reduced-space QP subproblem.*

This theorem is proved in two steps. In the first step, we show that both the RS and MFS QP subproblems are equivalent which implies that the minimizer $\hat{x}_k$ from the MFS QP subproblem is a solution to the RS QP subproblem and vice-versa. In the next step, we show that the solution $\hat{\lambda}_k$ provided by the MFS QP subproblem is a solution to the RS QP subproblem and vice-versa. Additionally, this means that whenever there is a unique $(\hat{x}_k, \hat{\lambda}_k)$ solution for the RS subproblem, the $(\hat{x}_k, \hat{\lambda}_k)$ obtained from solving both the QP subproblems are identical. Therefore, starting from an (x_0, λ_0), the sequence of iterates $\{(x_k, \lambda_k)\}$ generated by the reduced-space method and the modified full-space method are identical in an inequality-constrained optimization setting, assuming the uniqueness of the QP subproblem solutions.

We do not delve any deeper into the proof of this theorem in this paper. We also note that theorems 2, 3, 4, and 4.1 can be extended to an inequality-constrained algorithm based on SQP.

D. A practical SQP algorithm with SURF

We now present a generalized SQP-based SURF algorithm below, incorporating the results from the previous three subsections.

Algorithm 4 SURF with SQP

SURF unifies the reduced- and full-space methods for any general, 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 solving $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - C_y^T \lambda_k$
4: Apply the BFGS update to compute $\widehat{H}_k$, using $d_{k-1} = \alpha_{k-1} p_{k-1}^{(z)}$ and $w_{k-1} = \nabla_z \mathcal{M}(z'_k, \pi'_k) - \nabla_z \mathcal{M}(z'_{k-1}, \pi'_{k-1})$
5: Solve the modified QP subproblem (14) at $[x_k, y'_k, \psi'_k, \lambda_k]^T$ to obtain search direction p_k
6: Compute α_k via a line search (if enforcing strong Wolfe conditions, update ψ at α using $\mathcal{R}_y^T \psi_\alpha = -\mathcal{F}_y^T - C_y^T \lambda - \rho C_y^T C$)
7: Update $\left[x_{k+1}, y_{k+1}, \psi_{k+1}, \lambda_{k+1}\right]^T = \left[x_k, y'_k, \psi'_k, \lambda_k\right]^T + \alpha_k p_k$
8: **end loop**

Note: $z = (x, y)$, $\pi = (\lambda, \psi)$, $H_k = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix}$ is the Hessian of the Lagrangian $\mathcal{M}$, and α_k is the step length.

Note that this algorithm is at a very high level, and hides low level implementation details such as the line search algorithm, BFGS update, penalty parameter and slack variable updates, to name a few. If following an all-inequality approach for specifying constraints, all constraints will be of the form $C(x, y) \geq 0$. In that case, $\mathcal{R}(x, y) = 0$ will be implemented as two sets of constraints: $\mathcal{R}(x, y) \geq 0$, and $-\mathcal{R}(x, y) \geq 0$.

Computational convergence conditions

A point $[x_k, y'_k, \psi'_k, \lambda_k]^T$ is considered sufficiently optimal in Algorithm 4 if it satisfies the following first-order optimality conditions to specified tolerances:

$$\bar{C}(x, y) \geq 0, \quad \lambda \geq 0, \quad \lambda^T C(x, y) = 0, \quad \text{and} \quad m_z = dm/dz = 0, \tag{16}$$

where $\bar{C}(x, y) = (C(x, y), \mathcal{R}(x, y), -\mathcal{R}(x, y))$ is the concatenated vector of all constraints in all-inequality form. It can be shown that this set of optimality conditions in an MFS algorithm is equivalent to the optimality conditions in an RS algorithm.

Potential pitfalls

It should be noted that the *Lagrange multiplier updates* in an all-inequality implementation is slightly different. If ψ^+ and ψ^- are the Lagrange multipliers corresponding to $\mathcal{R}(x, y) \geq 0$ and $-\mathcal{R}(x, y) \geq 0$, the updates in lines 3 and 6 in Algorithm 4 must be done such that $\mathcal{R}_y^T(\psi^+ - \psi^-) = -\mathcal{F}_y^T - C_y^T \lambda$, and $\mathcal{R}_y^T(\psi_\alpha^+ - \psi_\alpha^-) = -\mathcal{F}_y^T - C_y^T \lambda - \rho C_y^T C$. For convenience, we recommend setting ψ^- as 0, and ψ^+ as the solution obtained from solving the linear system in lines 3 and 6.

It must also be noted that when switching from a reduced-space approach to SURF for any problem, the *scaling* of the additional design variables y must be taken into account. If the magnitudes of x and y differ by an order of magnitude or more, one of them must be scaled in order to approximately match the magnitude of the other. Otherwise, the optimization algorithm for SURF might be slow to converge depending on the problem.

E. Adaptive tolerance selection

Our investigation has thus far focused on the unification of the RS and FS architectures using the SURF algorithm. However, in order to exploit the computational benefits of a hybrid algorithm, we need to determine optimal tolerances that maximize the efficiency. In this subsection, we present a strategy for dynamically determining tolerances for each model evaluation, leveraging the Armijo condition in the line search.

The Armijo condition in Eq. (12) ensures that an accepted step length α results in a sufficient decrease in the merit function ϕ. When applying inexact tolerances on $\mathcal{R}(x, y) = 0$ during model evaluations, the resulting error in the merit

function $\phi(\alpha)$ should only be a small percentage of the required sufficient decrease at α. This ensures that an acceptable step length in the RS is also acceptable for SURF. The required sufficient decrease at α is -$\eta_A \alpha \phi'(0)$. Therefore, we want the absolute error $|\epsilon_\phi|$ in the merit function to be less than or equal to -$\eta_T \eta_A \alpha \phi'(0)$ where $0 \leq \eta_T < 1$ is a small positive constant. Note that a negative sign is added to the sufficient decrease term since $\phi'(0) < 0$.

From Eq. (9) for the merit function, we can exactly measure the error in the terms $\psi^T \mathcal{R}(x, y)$ and $\frac{1}{2}\rho \mathcal{R}(x, y)^T \mathcal{R}(x, y)$ as respectively $\psi^T \epsilon_r$ and $\frac{1}{2}\rho \epsilon_r^T \epsilon_r$, where $\epsilon_r = \mathcal{R}(x, y)$ is the tolerance on the nonlinear solvers. We now approximate ϵ_ϕ using adjoint-based error estimation for the remaining three terms as

$$\epsilon_\phi = \frac{\mathrm{d}f}{\mathrm{d}r}\epsilon_r + (\lambda + \rho C(x, y))^T \frac{\mathrm{d}c}{\mathrm{d}r}\epsilon_r + \psi^T \epsilon_r + \frac{1}{2}\rho \epsilon_r^T \epsilon_r. \tag{17}$$

The sufficient decrease requirement $|\epsilon_\phi| \leq -\eta_T \eta_A \alpha \phi'(0)$ now becomes

$$\left| \left(\frac{\mathrm{d}f}{\mathrm{d}r} + (\lambda + \rho C(x, y))^T \frac{\mathrm{d}c}{\mathrm{d}r} \right) \epsilon_r + \psi^T \epsilon_r + \frac{1}{2}\rho \epsilon_r^T \epsilon_r \right| \leq -\eta_T \eta_A \alpha \phi'(0). \tag{18}$$

Using $\dfrac{\mathrm{d}f}{\mathrm{d}r} = -\dfrac{\partial \mathcal{F}}{\partial y}\left(\dfrac{\partial \mathcal{R}}{\partial y}\right)^{-1}$ and $\dfrac{\mathrm{d}c}{\mathrm{d}r} = -\dfrac{\partial C}{\partial y}\left(\dfrac{\partial \mathcal{R}}{\partial y}\right)^{-1}$, we get

$$\left| -\left(\frac{\partial \mathcal{F}}{\partial y} + (\lambda + \rho C(x, y))^T \frac{\partial C}{\partial y} \right) \left(\frac{\partial \mathcal{R}}{\partial y} \right)^{-1} \epsilon_r + \psi^T \epsilon_r + \frac{1}{2}\rho \epsilon_r^T \epsilon_r \right| \leq -\eta_T \eta_A \alpha \phi'(0). \tag{19}$$

Whenever we apply the Lagrange multiplier update in line 6 in Algorithm 4, the first term in the equation above equals $\psi^T \epsilon_r$, and the inequality reduces to

$$\left| 2\psi^T \epsilon_r + \frac{1}{2}\rho \epsilon_r^T \epsilon_r \right| \leq -\eta_T \eta_A \alpha \phi'(0). \tag{20}$$

Until this point in the paper, we only considered a scalar penalty parameter ρ instead of a vector of penalty parameters for simplifying the analysis. However, in practice, ρ is a vector initialized with zero, and the vector gets updated in each optimization iteration to ensure a descent direction. We note that within the SURF algorithm, the penalty parameters in the vector ρ corresponding to the residuals $\mathcal{R}(x, y) = 0$ remains at zero or extremely low values if some nominal tolerance is enforced. This means that our tolerance selection criterion above could be further simplified to

$$|2\psi^T \epsilon_r| \leq -\eta_T \eta_A \alpha \phi'(0). \tag{21}$$

Now applying the Cauchy-Schwarz inequality $|\psi^T \epsilon_r| \leq \|\psi\|_2 \|\epsilon_r\|_2$, and the inequality $\|\psi\|_2 \leq \sqrt{n_r}\|\psi\|_\infty$, we get

$$\|\epsilon_r\|_2 \leq -\frac{1}{2\sqrt{n_r}\|\psi\|_\infty} \eta_T \eta_A \alpha \phi'(0), \tag{22}$$

where n_r is the number of scalar residuals in $\mathcal{R}(x, y)$.

Converging the nonlinear solvers in each model evaluation such that the residual norm is below the tolerance limit derived in Eq. (22) ensures a sufficient decrease during the line search. These adaptively computed inexact tolerances save computation by relaxing tolerances whenever possible. A value of 0.01 for the constant η_T works well for many problems in practice. This value essentially ensures that the error in the merit function is less than one percent of the required sufficient decrease. However, we note that higher values of η_T, such as 0.1, also yield good results for some problems we tested.

We also note that Eq. (22) can be readily adapted to problems with multiple residual systems $\mathcal{R}_1(x, y_1)$, $\mathcal{R}_2(x, y_2)$, etc. In such problems, it is ideal for each term to contribute equally to the error in the merit function. Therefore, the tolerance selection criterion for each residual system can be written as

$$\|\epsilon_{r_i}\|_2 \leq -\frac{1}{2n_r\sqrt{n_{r_i}}\|\psi_i\|_\infty} \eta_T \eta_A \alpha \phi'(0), \tag{23}$$

where n_r is the number of nonlinear systems in the model, n_{r_i} is the length of the ith residual vector, and ψ_i is the Lagrange multiplier for the ith residual system. This criterion offers finer control as we can now choose tolerances on a variable-by-variable basis per model evaluation.

V. Numerical Results

In this section, we present the results from applying the general, SQP-based SURF algorithm on two different application problems. In the first problem, we maximize the efficiency of a low-fidelity motor model by optimizing its sizing variables. In the next problem, we maximize the annual energy production (AEP) from a wind farm by optimizing the locations of wind turbines.

The implementation of our SQP optimizer follows the algorithm outlined in [13] for the SNOPT optimizer. The QP subproblems within the SQP algorithm are solved using the open-source QP solver OSQP [14]. The computational models used in both studies are written in the Computational System Design Language (CSDL) [6].

A. Motor optimization

We first apply the SURF algorithm to a permanent magnet synchronous motor (PMSM) optimization problem. The motor model is based on a differentiable method [15] proposed by Cheng et al. for the low-fidelity analysis of a PMSM. The motor model employed in this study was originally developed as a part of a large-scale system model of an eVTOL aircraft [16]. In our study, we maximize the efficiency of a PMSM subject to lower and upper bounds on the length and diameter of the motor. The motor optimization problem can be stated as

$$\begin{aligned} \text{maximize} \quad & \eta \\ \text{with respect to} \quad & D, L \in \mathbb{R} \\ \text{subject to} \quad & 0.05 \leq D \leq 0.15, \\ & 0.15 \leq L \leq 0.35 \\ \text{where} \quad & \mathcal{R}_B(B_m, D, L) = 0, \quad \mathcal{R}_\tau(\tau_{EM}, D, L) = 0, \end{aligned} \tag{24}$$

where η is the efficiency, D and L are the diameter and length of the motor in m, $\mathcal{R}_B$ is the residual that computes the flux density B_m of the magnet, and $\mathcal{R}_\tau$ is the residual that computes the electromagnetic torque τ_{em} . A schematic diagram for the full motor model is shown in Fig. 4. For our study, the inputs angular speed ω, load torque τ_L, and voltage limit V_{max} are taken as 3400 rpm, 500 Nm, and 800 V respectively.

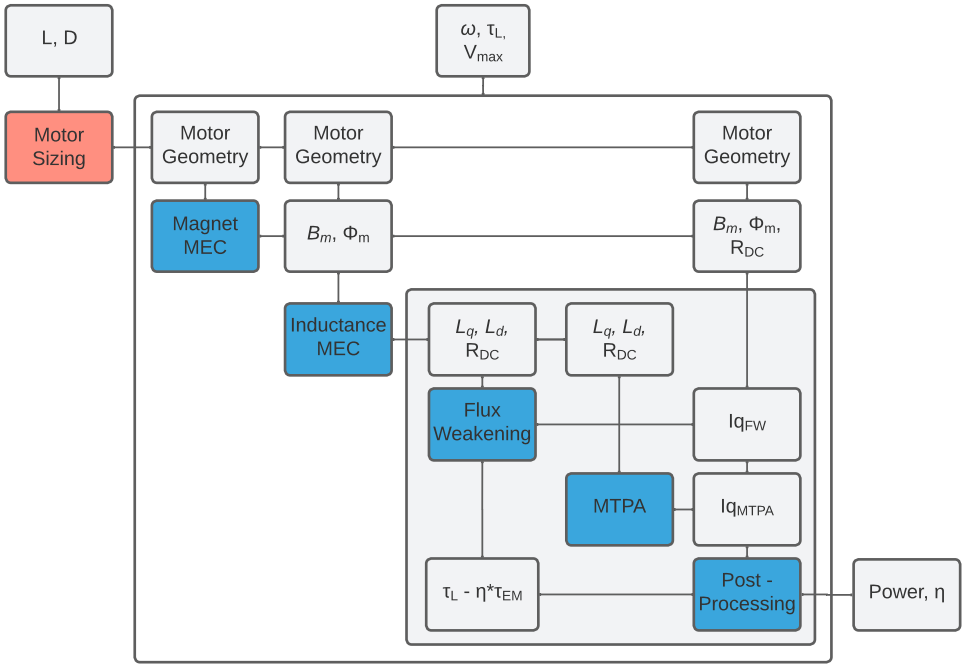


Fig. 4 Motor model architecture [16].

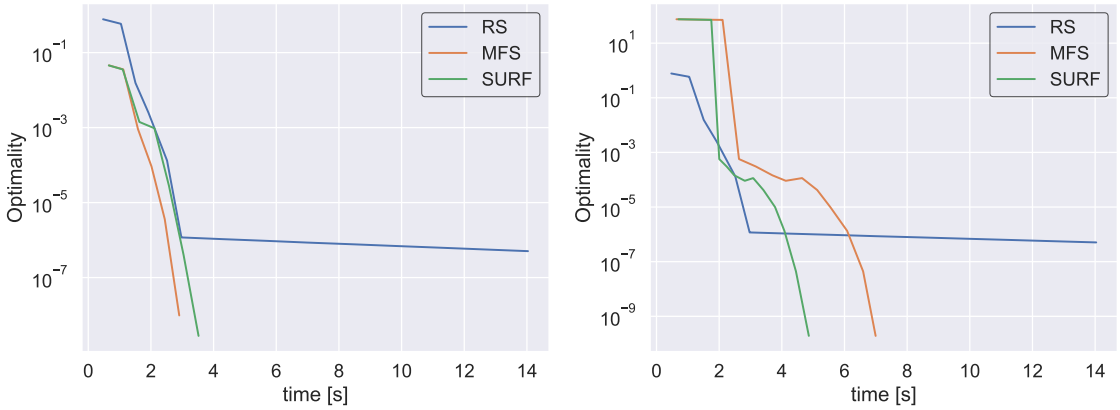


Fig. 5 Convergence in optimality for RS, MFS and SURF when: (a) $\mathcal{R}_B$ is a constraint, (b)$\mathcal{R}_\tau$ is a constraint.

Figure 5 shows the convergence in optimality for the reduced-space, modified full-space and SURF algorithms for two cases. In the first case, $\mathcal{R}_B(B_m, D, L) = 0$ was turned into a constraint for the SURF and MFS methods while in the second case, $\mathcal{R}_\tau(\tau_{EM}, D, L) = 0$ was made a constraint for MFS and SURF. Note that RS shown in both plots are the same, and it follows the standard problem formulation (24) where both residuals are solved exactly. SURF uses the adaptive tolerance selection strategy explained in section IV. In both cases, only one residual is made a constraint, and we do not test MFS or SURF with multiple residuals turned into constraints at a time. Because of the particular nature of these nonlinear equations, a Newton solver is not suitable, and we need bracketing methods for robust solutions. Therefore, a FS approach cannot converge for this problem, and this was confirmed in our tests.

We observe that during early iterations, the MFS and SURF methods are slightly better in terms of computational time compared to the RS for the first case. However, we see that the trend is reversed in the second case. For the first case, we see that MFS is slightly faster than SURF. This can be attributed to the slower convergence of the QP solver within the SQP algorithm when the residuals are only partially converged. This implies that although we gain more time with inexact model evaluations, this could sometimes cost us more in the optimization algorithm. For the second case, we see that SURF is approximately 20 percent faster than MFS. In this case, SURF gains more time from partial model evaluations compared to the additional time incurred for the QP solutions. Therefore, from the 2 cases above, we infer that there exists a tradeoff between model evaluation time and QP solver time for the SURF algorithm, and the tradeoff depends on the nonlinear residual that is included as a constraint.

We note that the motor problem studied here is a relatively small problem to observe significant differences in the computation time. However, a recurring observation from both cases is that MFS and SURF converge more tightly to a better optimal solution than RS. We attribute this to the high sensitivity of the reduced-space method to the total derivatives of the objective and constraints with respect to the vector of design variables x.

B. Wind farm layout optimization

The wind farm layout optimization (WFLO) problem is a highly multimodal problem characterized by multiple local optima. Therefore, these problems are traditionally solved using gradient-free optimizers due to their ability to navigate tough design space. However, gradient-free methods are prohibitively expensive for large-scale wind farm models with hundreds of design variables. For this reason, gradient-based optimization methods are now widely studied for large-scale WFLO. With new strategies such as the Wake Expansion Continuation (WEC) [17], gradient-based approaches are now competitive with gradient-free methods.

We now solve the wind farm layout optimization problem

$$\begin{aligned}
\text{maximize} \quad & AEP \\
\text{with respect to} \quad & \mathbf{x}, \mathbf{y} \in \mathbb{R}^{N_t} \\
\text{subject to} \quad & d_{ij} \geq d_{min} \qquad \text{for } i = 1, 2, ..., N_t - 1 \text{ and } j = i+1, i+2, ..., N_t, \\
& \phi(x_i, y_i) \geq 0 \qquad \text{for } i = 1, 2, ..., N_t \\
\text{where} \quad & \mathcal{R}(\mathbf{x}, \mathbf{y}, \mathbf{w}_{eff}) = 0,
\end{aligned} \tag{25}$$

where AEP is the annual energy production, $\mathbf{x}$ and $\mathbf{y}$ are vectors containing (x_i, y_i) coordinates denoting the location of ith wind turbine, N_t is the number of wind turbines to be positioned in the wind farm, d_{ij} is the distance between ith and jth turbines, $\phi \geq 0$ represents the boundary constraint, $\mathbf{w}_{eff}$ is a vector containing the effective wind speeds at each turbine location, and $\mathcal{R}(\mathbf{x}, \mathbf{y}, \mathbf{w}_{eff}) = 0$ represents the implicit nonlinear system that solves for $\mathbf{w}_{eff}$. The wind farm boundary is represented by $\phi = 0$, and $\phi(x, y) > 0$ implies (x, y) lies in the feasible region.

For our study, we use the open-source NREL 5MW wind turbine [18] model. The probability distribution of wind conditions is discretized into 16 wind direction bins for a constant wind speed of 9.8 m/s, as in the IEA37 case studies. The turbine wakes were formulated using the Bastankhah wake model [19], the turbulence intensity model from [20], and superposition with root sum-of-squares [21]. The spacing constraint enforces a minimum distance d_{min} of 1.8 rotor diameters between any two turbines. We use a simple rectangular wind farm boundary defined by $\mathbf{x}, \mathbf{y} \in [-750, 750]^{N_t}$ with $N_t = 9$.

We now apply optimization on the wind farm model using a variation of the new SURF algorithm and the traditional RS algorithm. The SURF algorithm applied here does not use the adaptive hybrid selection criteria presented in Sec. IV but instead uses FS model evaluations in line 6 and exact model evaluations in line 3 of Algorithm 4. This strategy consistently performs better than a RS approach for the WFLO problem (25).

Figure 6 shows a comparison of the RS and SURF methods applied to the WFLO problem. Figure 6(a) compares the convergence of the two methods while Fig. 6(b) compares the optimized results from both methods. Since WFLOP is multimodal, different optimization algorithms often converge to different local minimas. Depending on the minima it converges to, one algorithm can be significantly faster than the other, and the pattern could be random with randomized initial guesses. Therefore, while comparing convergence behaviors, it is important to ensure that all methods converge to the same solution. Figure 6(b) is thus necessary to confirm that both our methods converged to the same solution.

In Fig. 6(a), we see that SURF is approximately 25 per cent faster compared to RS when both methods converged to the same optimality of 10^{-5}. Figure 6(b) shows that the optimized results from both RS and SURF are identical. Altogether, this means that when both methods converge to the same local optima, SURF is faster than the traditional RS method.

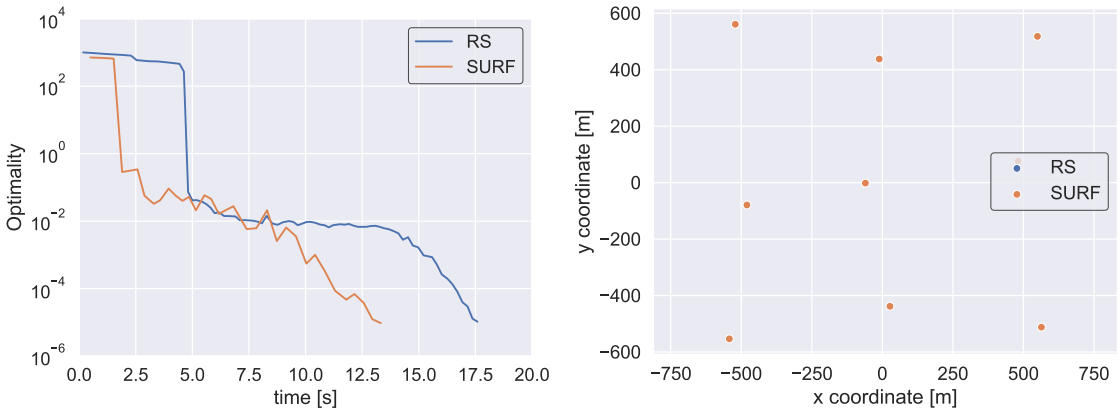


Fig. 6 (a) Convergence in optimality, and (b) optimized turbine locations for RS and SURF.

VI. Conclusion

In this paper, we presented a hybrid architecture for multidisciplinary design optimization (MDO) that unifies two conventional architectures, namely, the reduced-space architecture, and the full-space architecture. The goal of the paper was to extend the previously proposed SURF algorithm that achieves this unification, however, for a simplified, equality-constrained setting.

We reviewed the traditional reduced-space and full-space formulations, and the preliminary unification in Sec. III. The preliminary unification results are based on the modified full-space method that exhibits the behavior of a reduced-space method. These results are applicable for equality-constrained optimization problems, however, they neglect many practical components of an optimization algorithm.

In Sec. IV, we presented new theoretical results that prove the equivalence of the reduced-space and modified full-space methods in presence of line searches and quasi-Newton Hessian approximations. Furthermore, we provide additional results that extend this equivalence for general, inequality-constrained optimization problems solved using sequential quadratic programming (SQP). The SURF algorithm can then be obtained by applying inexact tolerances to certain solvers in the modified full-space algorithm. A method for adaptively selecting nonlinear solver tolerances for the SURF algorithm is also derived in this section.

In Sec. IV, we solved two design optimization optimization problems using the reduced-space and SURF algorithms based on SQP. SURF converges approximately 25 percent faster compared to reduced-space in one of the problems. In the other problem, SURF was able to converge more tightly than the reduced-space to provide a better solution.

SURF unifies the reduced-space and full-space architectures for a practical optimization setting using SQP solvers. We presented one tolerance selection criterion for nonlinear solvers but future work could look into more adaptive tolerance selection strategies including an update strategy for the linear systems when inexact nonlinear tolerances are applied. The numerical studies provided in this paper offer initial insights, however, extensive testing with different problems is required to mature the algorithms presented, and to discover other potential benefits of this new paradigm. The discussions in the paper were focused on SQP algorithms, however, interior point methods are also equally competitive for large-scale optimization. Studies on extending the SURF algorithm for interior point methods is another potential direction for future research.

Acknowledgments

The first author would like to thank Luca Scotzniovsky for his help with the low-fidelity motor model. This material is based upon work supported by the National Science Foundation under Grant No. 1917142.

References

[1] Joshy, A. J., and Hwang, J. T., "Unifying Monolithic Architectures for Large-Scale System Design Optimization," *AIAA Journal*, 2021, pp. 1–11. doi:https://doi.org/10.2514/1.J059954.

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

[3] Ho, Y.-C., and Pepyne, D. L., "Simple explanation of the no-free-lunch theorem and its implications," *Journal of optimization theory and applications*, Vol. 115, 2002, pp. 549–570.

[4] Hwang, J. T., and Martins, J. R., "A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 44, No. 4, 2018, p. 37. doi:https://doi.org/10.1145/3182393.

[5] Gray, J. S., Hwang, J. T., Martins, J. R., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104. doi:https://doi.org/10.1007/s00158-019-02211-z.

[6] Gandarillas, V., Joshy, A. J., Sperry, M. Z., Ivanov, A. K., and Hwang, J. T., "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis," *Structural and Multidisciplinary Optimization (under review)*, 2022.

[7] Joshy, A. J., Yan, J., and Hwang, J. T., "A hybrid architecture for large-scale system design optimization of PDE-based models," *AIAA SCITECH 2022 Forum*, 2022, p. 1614.

[8] Martins, J. R., and Lambe, A. B., "Multidisciplinary design optimization: a survey of architectures," *AIAA journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

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

[10] Biegler, L. T., Ghattas, O., Heinkenschloss, M., and van Bloemen Waanders, B., "Large-scale PDE-constrained optimization: an introduction," *Large-Scale PDE-Constrained Optimization*, Springer, 2003, pp. 3–13.

[11] Arora, J., and Wang, Q., "Review of formulations for structural and mechanical system optimization," *Structural and Multidisciplinary Optimization*, Vol. 30, No. 4, 2005, pp. 251–272.

[12] Haftka, R. T., "Simultaneous analysis and design," *AIAA journal*, Vol. 23, No. 7, 1985, pp. 1099–1103.

[13] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131. doi:https://doi.org/10.1137/S0036144504446096.

[14] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S., "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, Vol. 12, No. 4, 2020, pp. 637–672. doi:10.1007/s12532-020-00179-2, URL https://doi.org/10.1007/s12532-020-00179-2.

[15] Cheng, Z., Zhao, S., Scotzniovsky, L., Rodriguez, G., Mi, C., and Hwang, J. T., "A Differentiable Method for Low-Fidelity Analysis of Permanent-Magnet Synchronous Motor," *AIAA SCITECH 2023 Forum*, 2023, p. 1091.

[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.

[17] Thomas, J. J., McOmber, S., and Ning, A., "Wake expansion continuation: Multi-modality reduction in the wind farm layout optimization problem," *Wind Energy*, Vol. 25, No. 4, 2022, pp. 678–699. doi:https://doi.org/10.1002/we.2692, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2692.

[18] Jonkman, J., Butterfield, S., Musial, W., and Scott, G., "Definition of a 5-MW Reference Wind Turbine for Offshore System Development," 2009. doi:10.2172/947422, URL https://www.osti.gov/biblio/947422.

[19] Bastankhah, M., and Porté-Agel, F., "Experimental and theoretical study of wind turbine wakes in yawed conditions," *Journal of Fluid Mechanics*, Vol. 806, 2016, p. 506–541. doi:10.1017/jfm.2016.595.

[20] Crespo, A., and Herna´ndez, J., "Turbulence characteristics in wind-turbine wakes," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 61, No. 1, 1996, pp. 71–85. doi:https://doi.org/10.1016/0167-6105(95)00033-X, URL https://www.sciencedirect.com/science/article/pii/016761059500033X.

[21] Katic, I., Højstrup, J., and Jensen, N. O., "A simple model for cluster efficiency," *European wind energy association conference and exhibition*, Vol. 1, A. Raguzzi Rome, Italy, 1986, pp. 407–410.