Title

A hybrid architecture for large-scale system design optimization of PDE-based models

Authors

Joshy, Anugrah Jo
Yan, Jiayao
Hwang, John T

Publication Date

2022-01-03

DOI

10.2514/6.2022-1614

Peer reviewed

A hybrid architecture for large-scale system design optimization of PDE-based models

Anugrah Jo Joshy*, Jiayao Yan† and John T. Hwang‡
University of California San Diego, La Jolla, CA, 92093

Large-scale system design optimization has recently started being used in many fields as a tool in the design of new concepts. This has exposed many of its limitations, especially in regard to its usability and computational efficiency. State-of-the-art optimization algorithms can solve problems with tens of thousands of design variables in just hundreds of model evaluations. However, this can be computationally expensive for complex systems with costly simulations. A new, hybrid algorithm called SURF, which unifies the full-space and reduced-space architectures, can accelerate optimization of computational models of engineering systems involving large numbers of implicitly defined state variables. However, the theory and numerical results for SURF were restricted to equality-constrained problems solved using the method of Newton-Lagrange. This paper investigates the extension of SURF to the inequality-constrained optimization setting. We first extend the theory of SURF to bring sequential quadratic programming under the scope of SURF. Based on this extended theory, we propose an implementation of the SURF algorithm for inequality-constrained problems. We then test the hypothesis that this algorithm unifies the full-space and reduced-space algorithms, using a nonlinear topology optimization problem. Finally, we measure the improvement provided by SURF over the reduced-space method, using the same optimization problem. The results from the investigation suggest that the proposed new algorithm unifies the full-space and reduced-space methods for any constrained optimization setting. We also observe a reduction in the total number of linear system solutions required in solving the nonlinear systems within the model. However, we expect an actual improvement in the overall optimization time only with an industrial quality sequential quadratic programming optimizer implemented with SURF since the optimizer used in this study did not scale well with the number of design variables thus inherently favoring the reduced-space approach.

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
$\nabla F(x)$	=	gradient of a function F with respect to a variable x

I. Introduction

LARGE-scale system design optimization (LSDO) is a class of numerical techniques used in solving system design problems that are defined on high dimensional design spaces. Many of the large-scale systems are interdisciplinary, and their study requires multiphysics models and simulations. Design optimization of such large-scale systems are therefore characterized by multiple simulations of computationally expensive models. The optimization of such systems was considered infeasible until not long ago because of the challenges in computing the derivatives of a multidisciplinary

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

model for gradient-based optimization. Modelling multidisciplinary derivatives would require significant human effort and time as there was no general derivative computation method applicable to all multidisciplinary systems. Moreover, a gradient-free approach on a problem with hundreds of design variables is not viable with the computational power of a workstation.

However, the discovery of the unified derivatives equation (UDE) a few years ago impacted LSDO greatly by automating the coupled derivative computation. This automation was made possible through the UDE-based MAUD (modular analysis and unified derivatives) architecture [1] implemented in NASA's OpenMDAO [2], an open-source modeling framework for multidisciplinary analysis and optimization. There has been an explosion in the number of LSDO applications since this implementation, largely due to the improved usability of LSDO algorithms offered by OpenMDAO. The rapid growth of LSDO applications in aircraft wing design [3–19] and wind turbine design [20–34] are some examples.

As the barrier to entry lowered, many researchers from academia and industry started exploring LSDO as a tool in the design of new engineering systems such as CubeSats [35, 36], eVTOL aircraft [37–40], and many others. This has exposed many limitations of current LSDO algorithms, one of them being its inefficiency while optimizing large-scale systems with expensive computational models. The authors previously published a paper introducing a new, hybrid architecture called SURF (strong unification of reduced-space and full-space) [41] for overcoming the efficiency barrier for a subset of LSDO problems with simulations involving highly nonlinear systems. For the problem it considered, SURF was able to speed up the optimization by approximately an order of magnitude, compared to conventional architectures. Optimization problems with large numbers of implicitly calculated state variables benefit greatly from this new architecture; examples include problems involving high-fidelity CFD or nonlinear FEA solutions.

SURF defines an algorithm that unifies two conventional architectures, namely, multidisciplinary feasible (MDF), and simultaneous analysis and design (SAND). SURF can generate any hybrid of the two architectures by just changing nonlinear solver tolerances within the computational model and solving an additional linear system that computes the adjoint vector for the nonlinear systems. SURF is also able to vary the hybrid across optimization iterations. Although these properties give SURF great potential for efficiency improvement, the theoretical equivalence between SURF and its parent algorithms is proved and validated only for equality-constrained problems, and the method of Newton-Lagrange.

This paper has four high-level objectives. The first objective is to extend the theory for the current SURF algorithm from the method of Newton-Lagrange to sequential quadratic programming (SQP) for equality-constrained problems. The next objective is to leverage this theory and suggest a new SQP-based SURF algorithm for general constrained optimization problems. The third objective is to test the hypothesized new algorithm using a nonlinear topology optimization problem. The test verifies that the proposed algorithm unifies the full-space and reduced-space algorithms for an inequality-constrained optimization setting. The final objective is to measure and quantify the improvements offered by the new SURF algorithm over the conventional reduced-space algorithm using the same optimization problem.

We present the paper as follows. In Sec. II, we present current approaches used in LSDO with details on different optimization formulations. In Sec. III, we discuss the SURF algorithm in detail and its extension from the method of Newton-Lagrange to sequential quadratic programming . We then present the new, hypothesized SQP-based SURF algorithm for a general constrained optimization setting. In Sec. IV, we present a topology optimization problem, and the details of its implementation. We then use this problem for a comparison study of the new SURF algorithm with the reduced-space algorithm.

II. Background

A. Current approach in LSDO

Current LSDO methods follow the decades-old approach of a coupled model-optimizer setup where the model is run iteratively using the design variables generated by the optimizer. The optimizer computes the new set of design variables based on the model outputs. The model outputs needed by the optimizer are the values of the objective, constraints, and their derivatives. The model is evaluated iteratively until the convergence criteria for optimality and feasibility are met.

In efficient model implementations, computations of the output functions and their gradients take comparable times. The total optimization time is proportional to the number of times the model is evaluated before a solution is reached. State-of-the-art LSDO methods can optimize systems with tens of thousands of design variables in just hundreds of model evaluations. This makes the optimization of large and expensive models impractical as it might take weeks to reach the optimal design of a system whose simulation takes one hour.

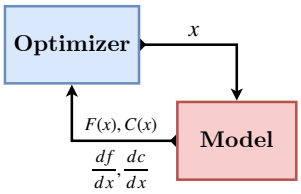


Fig. 1 Current approach [42] . The design variables, objective and constraints are respectively, x, $F(x)$ and $C(x)$.

Considering the current level of efficiency with the state-of-the-art optimization algorithms, a reduction in the optimization time is possible only if we can enable partial model evaluations. Partial model evaluations will bring down the time for each model evaluation, and the solution can be attained in less time when there is no significant increase in the number of model evaluations. This requires new optimizers and modeling frameworks that can facilitate a new paradigm where the solvers in the model could be instructed by the optimizer to partially converge to specified tolerances to speed up optimization.

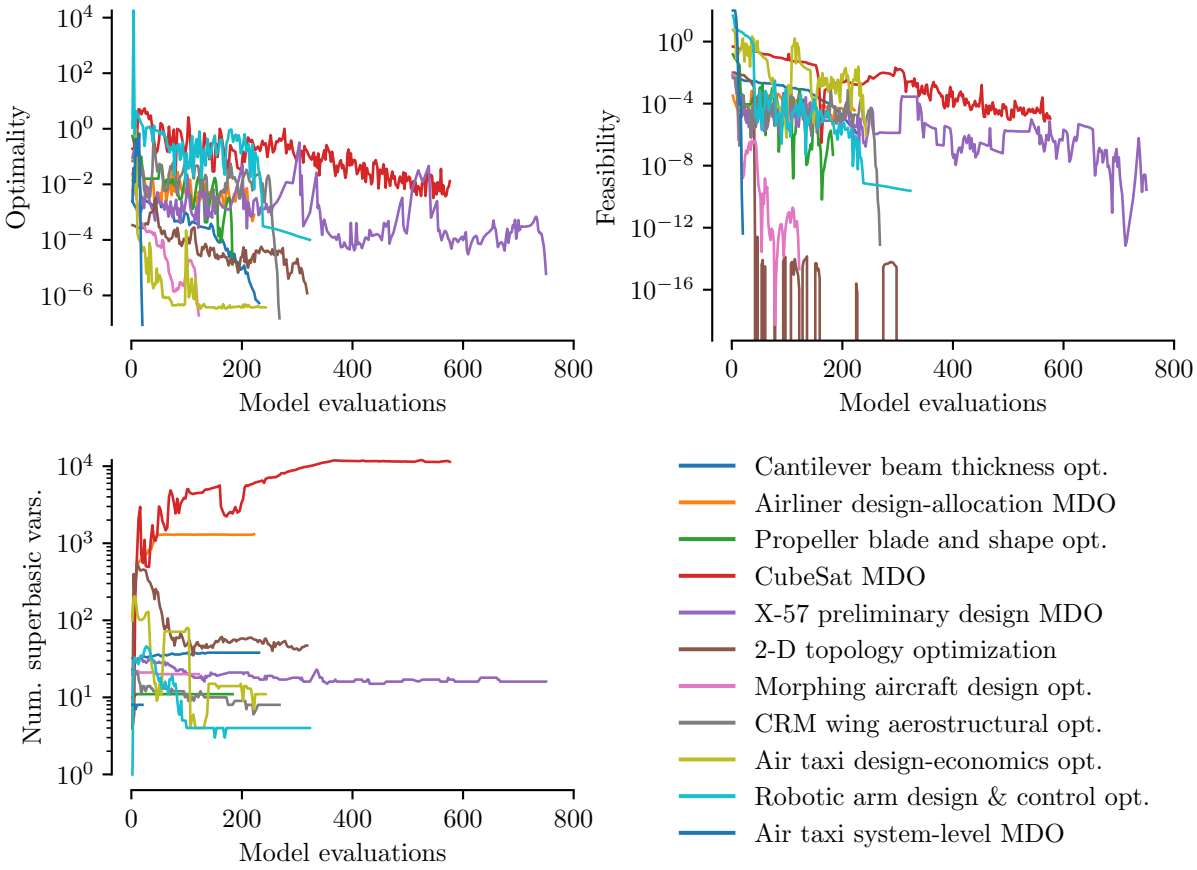


Fig. 2 The state-of-the-art in LSDO [43] . Previous LSDO applications are optimized in hundreds of model evaluations. Superbasic variables are the design variables that are not fixed due to bounds or constraints.

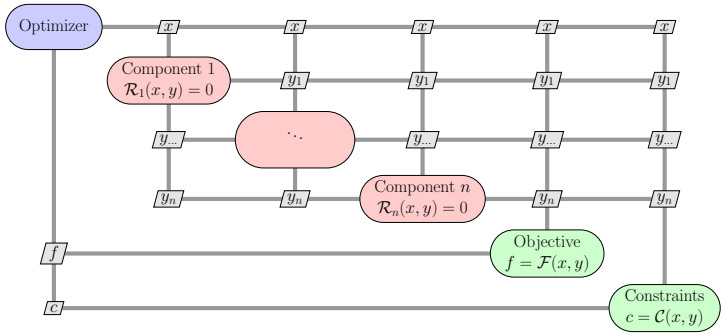
B. Optimization formulations

An optimization formulation or architecture refers to the way in which a system-level optimization problem is mathematically defined. There are multiple formulations possible for any given design problem but we will focus on the reduced-space (RS) formulation and the full-space (FS) formulation, also popularly known as the multidisciplinary

feasible (MDF) architecture and the simultaneous analysis and design (SAND) architecture, respectively. The full-space formulation considers the state variables defined by implicit equations as additional design variables, and the respective implicit equations as additional constraints. In the reduced-space formulation, the implicit equations are solved within the model to compute the state variables.

Reduced-space

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x, \mathcal{Y}(x)) \\ \text{s.t.} \quad & C(x, \mathcal{Y}(x)) \geq 0 \\ \text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0 \end{aligned} \tag{1}$$



Full-space

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

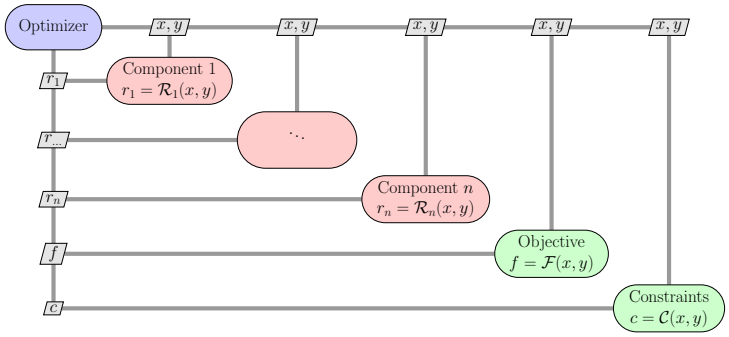


Fig. 3 Reduced-space versus full-space formulation for an inequality-constrained problem in the presence of implicitly computed state variables [42] .

Given the design variables x, the reduced-space formulation solves the residual equations $\mathcal{R}(x, \mathcal{Y}(x)) = 0$ (which defines the implicit state variables y) within the model to obtain the state variables as $y = \mathcal{Y}(x)$. Therefore, vectors in the design space for the reduced-space formulation consists of only x. In contrast, the full-space formulation treats the implicit state variables, y, as additional design variables and thus, the vectors in the design space consists of both x and y. The full-space formulation also treats the residuals $\mathcal{R}(x, y) = 0$ as additional constraints thereby ensuring that the y at a feasible solution will satisfy the residual equations. Each formulation has its own merits:

- The RS formulation results in expensive model evaluations because of the nonlinear solutions required but the optimization problem becomes smaller and simpler with fewer design variables and constraints.
- The FS formulation results in cheaper model evaluations since it does not solve any nonlinear systems but the optimization problem becomes larger and more complex with more design variables and constraints.

The FS formulation has the potential to be more efficient but in practice, optimization convergence issues or a greater increase in the number of model evaluations makes it less robust and negates the benefits from an inexpensive model. Therefore, most LSDO applications consider only a reduced-space approach for solving their problems. However, the RS formulation is very inefficient since it has to ensure tight convergence for the solutions of all nonlinear residuals in each optimization iteration. This requirement is necessary to ensure that the objective and constraints are consistent with their derivatives computed by the model at any x given by the optimizer.

We now look at the optimality conditions and Karush-Kuhn-Tucker (KKT) systems for both formulations in a simplified, equality-constrained setting.

Reduced-space equations

We restrict Eq. (1) to equality contraints and define the Lagrangian $l = \mathcal{L}(x, \lambda)$ as

$$\mathcal{L}(x, \lambda) = \mathcal{F}(x, \mathcal{Y}(x)) + \lambda^T C(x, \mathcal{Y}(x)) \tag{3}$$

Using the method of Newton-Lagrange, we obtain the following first order necessary optimality conditions:

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

The reduced-space KKT system for computing the search direction p_k is given by

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

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

Full-space equations

We restrict Eq. (2) to equality constraints and define the Lagrangian $m = \mathcal{M}(x, y, \psi, \lambda)$ as

$$\mathcal{M}(x, y, \psi, \lambda) = \mathcal{F}(x, y) + \psi^T \mathcal{R}(x, y) + \lambda^T C(x, y) \tag{6}$$

Once again, using the method of Newton-Lagrange, we obtain the following first order necessary optimality conditions:

$$\begin{aligned} \frac{\mathrm{d}m}{\mathrm{d}x} &= \frac{\partial \mathcal{F}}{\partial x} + \psi^T \frac{\partial \mathcal{R}}{\partial x} + \lambda^T \frac{\partial C}{\partial x} = 0 \qquad & \frac{\mathrm{d}m}{\mathrm{d}\psi} &= \mathcal{R}(x, y) = 0 \\ \frac{\mathrm{d}m}{\mathrm{d}y} &= \frac{\partial \mathcal{F}}{\partial y} + \psi^T \frac{\partial \mathcal{R}}{\partial y} + \lambda^T \frac{\partial C}{\partial y} = 0 \qquad & \frac{\mathrm{d}m}{\mathrm{d}\lambda} &= C(x, y) = 0 \end{aligned} \tag{7}$$

The full-space KKT system is then given by

$$\begin{bmatrix} m_{xx} & m_{xy} & {\mathcal{R}_x}^T & {C_x}^T \\ m_{yx} & m_{yy} & {\mathcal{R}_y}^T & {C_y}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ p_k^{(\psi)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k, y_k) \\ -C(x_k, y_k) \end{bmatrix}, \tag{8}$$

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

C. SURF: strong unification of reduced-space and full-space

SURF [42] is an algorithm that unifies the full-space and reduced-space formulations. It has the ability to generate any hybrid of the two formulations. The theory of the SURF algorithm is developed based on the modified full-space (MFS) method; MFS can generate the same sequence of design variable and Lagrange multiplier iterates as the reduced-space method using an underlying full-space KKT system. MFS provides a way to unify the full-space and reduced-space methods. It can then be used to derive the SURF algorithm, which provides access to a continuous spectrum of hybrids of the full-space and reduced-space methods. The existing SURF algorithm is limited to the equality-constrained optimization setting. The goal of the current paper is to extend it to the inequality-constrained optimization setting. To avoid any confusion, it is worth noting that MFS was originally called corrected full-space or CFS method in [42].

1. Modified full-space formulation

The MFS architecture shown in Fig. 4 is essentially a modified version of the original full-space method; it updates 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 design variables, state variables, and the corresponding Lagrange multipliers at the end of the k-th iteration, this method updates y_k to y'_k by solving $\mathcal{R}(x_k, y'_k) = 0$ and then ψ_k to ψ'_k by setting $dm/dy = 0$. The FS KKT system is then solved at the updated point $[x_k, y'_k, \psi'_k, \lambda_k]^T$ to obtain the search direction p_k toward the next iterate. The modified KKT system for the MFS method is then given by

$$\begin{bmatrix} m_{xx} & m_{xy} & {\mathcal{R}_x}^T & {C_x}^T \\ m_{yx} & m_{yy} & {\mathcal{R}_y}^T & {C_y}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ p_k^{(\psi)} \\ p_k^{(\lambda)} \end{bmatrix} = \begin{bmatrix} -m_x \\ 0 \\ 0 \\ -C(x_k, y'_k) \end{bmatrix} \tag{9}$$

$m_y = 0$ and $\mathcal{R}(x_k, y'_k) = 0$ after the two modifications are reflected in the left hand side vector. At this point, it is worth noting that $m_{xx},\ m_{xy},\ m_{yx},\ m_{yy},\ m_x,\ \mathcal{R}_x,\ \mathcal{R}_y,\ C_x$, and C_y are not the same for the FS and MFS methods since the MFS KKT system is computed at $[x_k, y'_k, \psi'_k, \lambda_k]^T$ while the FS KKT system is computed at $[x_k, y_k, \psi_k, \lambda_k]^T$. As a result of this modification of the FS KKT system (8), the solution $p_k^{(x)}$ and $p_k^{(\lambda)}$ from the MFS KKT system (9) is the same as the solution provided by the RS KKT system (5), the detailed proof of which can be found in [42]. Hence, the sequence of iterates generated by the modified full-space method and the reduced-space method are identical.

Modified full-space

$$
\begin{aligned}
\min_{x,y} \quad & \mathcal{F}(x, y) \\
\text{s.t.} \quad & C(x, y) \geq 0 \\
& \mathcal{R}(x, y) = 0 \\
\text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0 \\
& dm/dy = 0
\end{aligned}
\tag{10}
$$

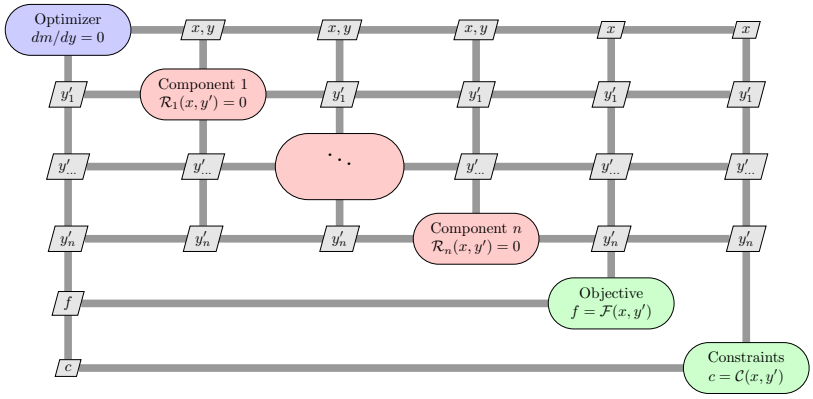


Fig. 4 Modified full-space architecture [42] .

A comparison of the reduced-space, full-space and modified full-space methods [42]:

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**

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

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

2. Strong unification of reduced-space and full-space

The complete unification of the full-space and reduced-space methods results from employing inexact solvers to perform the two modifications in the original MFS method. The algorithm that achieves the complete unification is called SURF (Algorithm 4) and is simply obtained by applying inexact solvers in lines 2 and 3 of the Algorithm 3. The loop that starts at line 1 indicates the start of each outer iteration. Line 4 updates the approximation to the Lagrangian Hessian evaluated at $[x_k, y'_k, \psi'_k, \lambda_k]^T$. Lines 5 and 6 assembles and solves the FS KKT system at the updated point $[x_k, y'_k, \psi'_k, \lambda_k]^T$ using a preconditioner. Lines 7 and 8 compute the step length using an appropriate line search, and apply the step along the search direction p_k. We note at this point that we did not prove that the MFS method was equivalent to the RS method with the addition of a line search into the algorithm. However, the equivalence still holds and it follows from the unification proof presented in [42], the details of which were not explicitly provided.

The SURF algorithm provides a complete unification of the FS and RS methods because it can give any hybrid of the RS and FS methods by just modifying the inexact solver tolerances in lines 2 and 3. Lines 2 and 3 performed with a zero tolerance gives us the exact solutions to $\mathcal{R}(x, y) = 0$, and $dm/dy = 0$, in which case the algorithm is just the same as the MFS method in Algorithm 3. Thus, a zero tolerance on the inexact solvers inside the SURF optimizer is equivalent to the reduced-space method since both MFS and RS generate the same sequence of iterates. Now, if we do not execute lines 2 and 3 in Algorithm 4 (which is equivalent to an infinite tolerance on the solvers), SURF reduces to the Algorithm 2 for the FS method. Therefore, SURF unifies the full-space and reduced-space methods, and can generate one of the two or a hybrid, simply by changing the inexact solver tolerances. The architecture resulting from the SURF algorithm is shown in Fig. 5.

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 $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - C_y^T \lambda_k$
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 $\left[x_{k+1}, y_{k+1}, \psi_{k+1}, \lambda_{k+1}\right]^T = \left[x_k, y'_k, \psi'_k, \lambda_k\right]^T + \alpha_k p_k$
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 (8), 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

$$A_k = \begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & C_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & C_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix}, b_k = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x_k, y'_k) \\ -C(x_k, y'_k) \end{bmatrix}, M_1 = \begin{bmatrix} 0 & 0 & \mathcal{I} & 0 \\ 0 & \mathcal{I} & 0 & 0 \\ \mathcal{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{I} \end{bmatrix}, M_{2,k} = \begin{bmatrix} \mathcal{R}_y & 0 & 0 & 0 \\ m_{yy} & \mathcal{R}_y^T & 0 & 0 \\ m_{xy} & \mathcal{R}_x^T & \mathcal{I} & 0 \\ C_y & 0 & 0 & \mathcal{I} \end{bmatrix} \tag{11}$$

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

SURF

$$
\begin{aligned}
\min_{x,y} \quad & \mathcal{F}(x,y) \\
\text{s.t.} \quad & C(x,y) \geq 0 \\
& \mathcal{R}(x,y) = 0 \\
\text{with} \quad & \mathcal{R}(x,y') \approx 0 \\
& dm/dy \approx 0
\end{aligned}
\tag{12}
$$

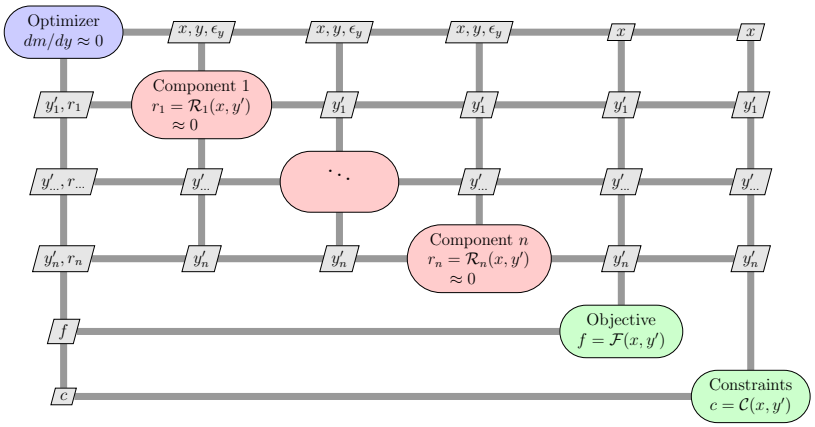


Fig. 5 SURF architecture[42] . Model solves the residual equations to tolerances ϵ_y provided by the optimizer.

III. Methodology

The current SURF algorithm unifies the RS and FS methods only for equality-constrained problems that are solved using the method of Newton-Lagrange. However, most of the practical optimization problems cannot be formulated as an equality-constrained problem due to the presence of inequality constraints. Moreover, the method of Newton-Lagrange is not well-suited for large-scale problems, even if they are equality-constrained.

As a first step toward broadening the range of applicability of SURF, we extend the SURF algorithm to make it compatible with the sequential quadratic programming (SQP) method for an equality-constrained optimization setting. This has two advantages. First, SQP methods represent one of the most successful class of methods for solving large-scale optimization problems (equality- or inequality-constrained) which means that SURF could now be used solve any equality-constrained optimization problem. Second, there is the possibility of a natural extension of SURF from equality-constrained problems to inequality-constrained problems through the SQP method since SQP methods handle both equality- and inequality-constrained problems in similar ways.

A. Extension of the SURF algorithm to an equality-constrained SQP method

We saw earlier in Sec. II.C.1 that the MFS KKT system (9) yields the same $p_k^{(x)}$ and $p_k^{(\lambda)}$ values as the RS KKT system (5). Here we show that the solution provided by the MFS KKT system and a modified full-space quadratic programmming (QP) subproblem are the same. Altogether, this would imply that the $p_k^{(x)}$ and $p_k^{(\lambda)}$ iterates generated by the modified full-space SQP algorithm and the reduced-space algorithm are the same.

The SQP algorithm for an equality-constrained problem solves a sequence of equality-constrained quadratic programming (QP) subproblems to find the search direction toward the next iterate. Each QP subproblem minimizes a quadratic approximation of the objective subject to linearized constraints. The quadratic approximation is based on the Hessian of the Lagrangian instead of the Hessian of the objective.

Considering the full-space problem (2) with only equality constraints, the QP subproblem at the $(k+1)$st iteration takes the form

$$
\begin{aligned}
\min_{v} \quad & \nabla\mathcal{F}(v_k)^T (v - v_k) + (v - v_k)^T H_k (v - v_k) && v = [x, y]^T, \; v_k = [x_k, y_k]^T, \\
\text{s.t.} \quad & C(v_k) + C_v(v_k)(v - v_k) = 0 && \text{where} \quad C_v = [C_x, C_y], \; \mathcal{R}_v = [\mathcal{R}_x, \mathcal{R}_y], \quad \text{and} \\
& \mathcal{R}(v_k) + \mathcal{R}_v(v_k)(v - v_k) = 0 && H_k \;\; \text{is the Lagrangian Hessian at } v_k, \psi_k, \lambda_k
\end{aligned}
\tag{13}
$$

The solution to the QP subproblem 13 is then equivalent to the solution of the following linear system:

$$
\begin{bmatrix}
m_{xx} & m_{xy} & \mathcal{R}_x{}^T & C_x{}^T \\
m_{yx} & m_{yy} & \mathcal{R}_y{}^T & C_y{}^T \\
\mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\
C_x & C_y & 0 & 0
\end{bmatrix}
\begin{bmatrix}
p_k^{(x)} \\ p_k^{(y)} \\ \psi_{k+1} \\ \lambda_{k+1}
\end{bmatrix}
=
\begin{bmatrix}
-\nabla_x \mathcal{F}(x_k, y_k) \\ -\nabla_y \mathcal{F}(x_k, y_k) \\ -\mathcal{R}(x_k, y_k) \\ -C(x_k, y_k)
\end{bmatrix}
\tag{14}
$$

By rearranging the terms in the first two block rows of the above system after substituting $\psi_{k+1} = \psi_k + p_k^{(\psi)}$ and $\lambda_{k+1} = \lambda_k + p_k^{(\lambda)}$, we can show that the solution from this system is same as the solution from the FS KKT system (8), i.e., $p_k^{(x)}$, $p_k^{(y)}$, $p_k^{(\psi)}$, and $p_k^{(\lambda)}$ are identical for both the systems.

Now considering the reduced-space problem (1) with only equality constraints, the QP subproblem at the $(k+1)$st iteration takes the form

$$\begin{aligned} \min_{x} \quad & \nabla F(x_k)^T (x - x_k) + (x - x_k)^T H_k (x - x_k) \qquad \text{where} \quad H_k \ \text{ is the Lagrangian Hessian at } x_k, \lambda_k \\ \text{s.t.} \quad & c(x_k) + c_x(x_k)(x - x_k) = 0 \end{aligned} \tag{15}$$

The solution to the QP subproblem 15 is then equivalent to the solution of the following linear system:

$$\begin{bmatrix} l_{xx} & {c_x}^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla_x F(x_k) \\ -c(x_k, y_k) \end{bmatrix} \tag{16}$$

By rearranging the terms in the first block row of the above system after substituting $\lambda_{k+1} = \lambda_k + p_k^{(\lambda)}$, we can show that the solution from this system is same as the solution from the RS KKT system (5), i.e., $p_k^{(x)}$, and $p_k^{(\lambda)}$ are identical for both the systems.

The aforementioned two observations show the equivalence between the method of Newton-Lagrange and sequential quadratic programming for equality-constrained problems formulated using a full-space or reduced-space approach. Now modifying the QP subproblem in a way similar to the MFS method, we get

$$\begin{aligned} \min_{v} \quad & \nabla \mathcal{F}(v'_k)^T (v - v'_k) + (v - v'_k)^T H'_k (v - v'_k) && v = [x, y]^T, \ v'_k = [x_k, y'_k]^T, \\ \text{s.t.} \quad & C(v'_k) + C_v(v'_k)(v - v'_k) = 0 && \text{where} \quad C_v = [C_x, C_y], \ \mathcal{R}_v = [\mathcal{R}_x, \mathcal{R}_y], \ \text{ and} \\ & \mathcal{R}_v(v'_k)(v - v'_k) = 0 && H'_k \ \text{ is the Lagrangian Hessian at } v'_k, \psi'_k, \lambda_k \end{aligned} \tag{17}$$

The solution to the modified QP subproblem 17 is then equivalent to the solution of the following linear system:

$$\begin{bmatrix} m_{xx} & m_{xy} & {\mathcal{R}_x}^T & {C_x}^T \\ m_{yx} & m_{yy} & {\mathcal{R}_y}^T & {C_y}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k^{(x)} \\ p_k^{(y)} \\ \psi_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{F}(x_k, y'_k) \\ -\nabla_y \mathcal{F}(x_k, y'_k) \\ 0 \\ -C(x_k, y'_k) \end{bmatrix} \tag{18}$$

Again, by rearranging the terms in the first two block rows of the above system by making substitutions as before, we can show that the solution from this system is same as the solution from the MFS KKT system (9) i.e., $p_k^{(x)}$, $p_k^{(y)}$, $p_k^{(\psi)}$, and $p_k^{(\lambda)}$ are identical for both the systems.

The above observation is essentially the mathematical proof that an SQP algorithm with a modified QP subproblem 17 in each iteration generates the same sequence of iterates as the MFS algorithm. This proof along with the original equivalence between MFS and RS methods proved in [42] imply that the modified SQP algorithm is in fact equivalent to the RS method. Based on this result, we can extend the SURF algorithm to an SQP method for equality-constrained problems. The detailed algorithm for SURF with sequential quadratic programming is shown in Algorithm. 5. We see that the new algorithm follows the original SURF algorithm very closely. One advantageous exception is that there is no requirement to compute the preconditioning matrices as in the original SURF algorithm for solving the KKT system. This is because we now solve the QP subproblem iteratively using a QP solver instead of solving the linear KKT system.

This SURF algorithm for SQP has all the benefits of the original SURF algorithm. In addition, this unification extends the applicability of SURF to any equality-constrained optimization problem whether small- or large-scale. The new algorithm also opens up an avenue for extending the theoretical unification provided by SURF to an inequality-constrained setting using the SQP method. This stems from the fact that SQP methods for equality- and inequality-constrained problems are very similar in theory. In the next subsection, we propose a SURF algorithm for inequality-constrained problems based on Algorithm. 5.

Algorithm 5 SURF with SQP (sequential quadratic programming)
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 $\mathcal{R}_y^T \psi'_k = -\mathcal{F}_y^T - C_y^T \lambda_k$
4: Update the approximation to the Lagrangian Hessian, H_k, at $[x_k, y'_k, \psi'_k, \lambda_k]^T$
5: Solve the modified QP subproblem (17)
6: Compute α_k via a line search
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: $H_k = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix}$ is the Hessian of the Lagrangian, p_k is the search direction and α_k is the step size.

B. Extension of the SURF algorithm to inequality-constrained problems

The SQP algorithm for an inequality-constrained optimization problem solves a sequence of inequality-constrained quadratic programming (QP) subproblems to find the search direction toward the next iterate. The QP subproblems are formulated using the approximations as mentioned in previous subsection for equality-constrained problems.

For the reduced-space problem (1), the QP subproblem at the $(k+1)$st iteration takes the form

$$\begin{aligned} \min_x \quad & \nabla F(x_k)^T (x - x_k) + (x - x_k)^T H_k (x - x_k) \qquad \text{where} \quad H_k \ \text{ is the Lagrangian Hessian at } x_k, \lambda_k \\ \text{s.t.} \quad & C(x_k) + C_x(x_k)(x - x_k) \geq 0 \end{aligned} \tag{19}$$

For the full-space problem (2), the QP subproblem at the $(k+1)$st iteration takes the form

$$\begin{aligned} \min_v \quad & \nabla \mathcal{F}(v_k)^T (v - v_k) + (v - v_k)^T H_k (v - v_k) && v = [x, y]^T, \ v_k = [x_k, y_k]^T, \\ \text{s.t.} \quad & C(v_k) + C_v(v_k)(v - v_k) \geq 0 && \text{where} \quad C_v = [C_x, C_y], \ \mathcal{R}_v = [\mathcal{R}_x, \mathcal{R}_y], \ \text{ and} \\ & \mathcal{R}(v_k) + \mathcal{R}_v(v_k)(v - v_k) = 0 && H_k \ \text{ is the Lagrangian Hessian at } v_k, \psi_k, \lambda_k \end{aligned} \tag{20}$$

Our hypothesis is that the modified QP subproblem of the form

$$\begin{aligned} \min_v \quad & \nabla \mathcal{F}(v'_k)^T (v - v'_k) + (v - v'_k)^T H'_k (v - v'_k) && v = [x, y]^T, \ v'_k = [x_k, y'_k]^T, \\ \text{s.t.} \quad & C(v'_k) + C_v(v'_k)(v - v'_k) \geq 0 && \text{where} \quad C_v = [C_x, C_y], \ \mathcal{R}_v = [\mathcal{R}_x, \mathcal{R}_y], \ \text{ and} \\ & \mathcal{R}_v(v'_k)(v - v'_k) = 0 && H'_k \ \text{ is the Lagrangian Hessian at } v'_k, \psi'_k, \lambda_k \end{aligned} \tag{21}$$

yields the same results as its reduced-space counterpart, i.e., $p_k^{(x)}$ and $p_k^{(\lambda)}$ are identical for both the methods. This means that the iterates generated by the reduced-space SQP method and the modified full-space SQP method are the same in the context of inequality-constrained optimization. The SURF algorithm for inequality-constrained SQP will then be the same as Algorithm. 5 that was presented earlier for the equality-constrained SQP method.

However, we do not have a proven theoretical equivalence for this hypothesis but our intuition suggests that the hypothesis must be true based on the result from the previous subsection. To prove the equivalence of reduced-space and full-space in the case of equality-constrained SQP, we used the original equivalence for the method of Newton-Lagrange. However, the absence of an intermediate method such as the Newton-Lagrange in the case of inequality-constrained SQP suggests that a promising approach to proving the equivalence would be to leverage the equivalence in the case of SQP-based SURF for equality-constrained problems to naturally extend SURF to inequality-constrained problems. In the next section, we perform investigations to verify this hypothesis by applying Algorithm. 5 on an inequality-constrained topology optimization problem.

IV. Numerical results

A. Topology optimization problem

We perform topology optimization on a hyperelastic cantilever beam whose domain is modeled using uniform quadrilateral elements. The problem uses a Saint–Venant Kirchhoff model whose elastic energy density is given by

$$\psi(\boldsymbol{u}) = \frac{1}{2} S : E \,, \tag{22}$$

where u is the deformation vector, $E = \frac{1}{2}(C - \mathrm{I})$ is the Green–Lagrange strain, S is the stress tensor defined as

$$S = 2\mu E + \lambda \mathrm{tr}(E)\mathrm{I} \,, \tag{23}$$

where $C = F^\top F$ is the Right Cauchy–Green tensor, λ and μ are Lamé parameters, F is the deformation gradient expressed as $F = \mathrm{I} + \nabla u$; and I is the identity matrix.

We model the beam in FEniCS [44]. The DOLFIN library in FEniCS solves the FEA problem for displacements under a traction load on the right face of the beam. The goal of the optimization is to minimize the compliance of the beam subject to the constraint of an upper bound on the average density. The design variables x are the densities of the elements inside the discretized mesh and the implicit state variables u are the displacements. The optimization problem is formulated mathematically as

$$\begin{aligned} \min_{0 \le x \le \rho_{\max}} \quad & F^T u \\ \text{s.t.} \quad & \rho_{\text{avg}}(x) \le 0.4\rho_{\max} \end{aligned} \qquad \text{with} \quad K(x,u)u - F = 0, \tag{24}$$

where x is the density distribution, u is the displacement vector, F is the force vector, $\rho_{\text{avg}}(\cdot)$ is the average density function for the domain, $\rho_{\max}$ is the maximum density of the material, and K is the function that computes the nonlinear stiffness matrix.

We build the model of the problem in Python using the code from the ATOMiCS [45] (automated topology optimization using OpenMDAO and FEniCS) package, which contains a suite of topology optimization problems. As the models in the ATOMiCS package are written in the OpenMDAO framework, which only supports a conventional optimization paradigm, we remove the dependency on OpenMDAO by constructing a similar software framework that replaces OpenMDAO and interfaced with ATOMiCS to facilitate an intrusive paradigm for the SQP-based SURF optimizer.

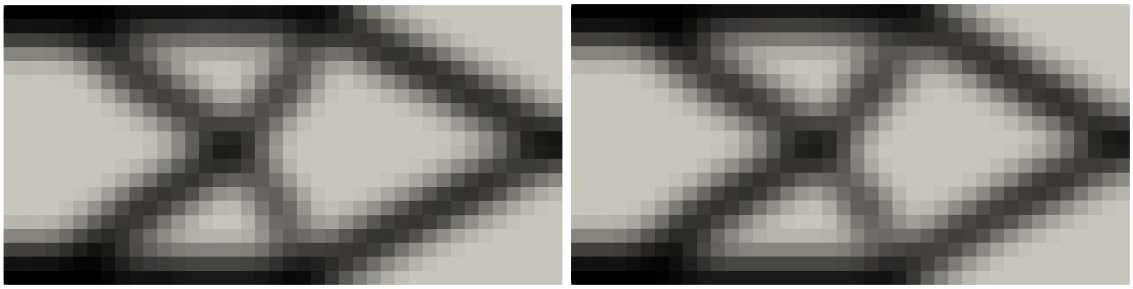
Using the theory from SNOPT [46] optimizer, we develop a basic SQP optimizer for solving general inequality-constrained problems. SNOPT is one of the most successful and widely-used general-purpose optimizers for solving large-scale design optimization problems. In order to solve the intermediate QP subproblems, we use an open-source QP solver called OSQP [47]; OSQP stands for operator splitting solver for quadratic programs.

We now apply the SQP-based reduced space, modified full-space and SURF algorithms for solving this optimization problem. We again note that the MFS and SURF algorithms proposed in Sec. III.B and used here are not theoretically validated, and the goal of this study is to investigate and gain insights into the proposed architecture. In the next subsection, we compare and study the numerical results from the application of different optimization algorithms on this problem.

B. Numerical study

We begin this subsection by referring back to Algorithm. 5 for SURF based on SQP. We see that running this algorithm without performing the inexact solutions in lines 2 and 3 is identical to a full-space SQP algorithm. An equivalence of SURF and FS follows directly from this observation. The first goal of this subsection is to verify that SURF performed with exact solutions in lines 2 and 3 is similar (not equivalent) to the reduced-space method. We specifically mention that our goal is not to see identical solutions from SURF and RS even if there is a theoretical equivalence exists between both. This is because the approximations of Hessians using the BFGS updates and the inaccuracies in the line search algorithms lead to significant deviations from the theory and can affect both algorithms differently.

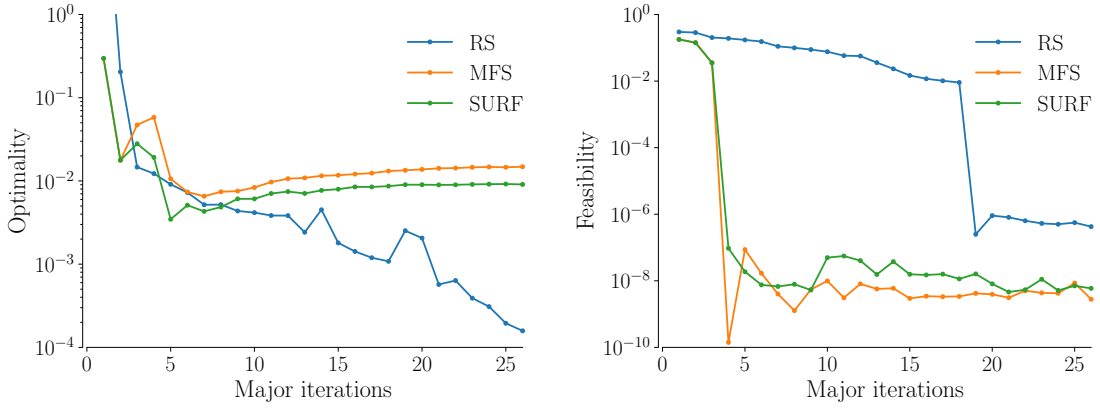
We first run the reduced-space algorithm and the SURF algorithm on the optimization problem. SURF algorithm is run with a very tight tolerance of 1e-16 on the absolute norm of the residual. SURF is now equivalent to the MFS algorithm for SQP. Our goal is to observe the solutions and verify that we get a similar solution from both algorithms.



(a) (b)

Fig. 6 Optimized structures. (a) RS. (b) MFS.

Fig. 6 shows the solutions when we run both the algorithms for 25 major iterations, the point where both the algorithms were sufficiently converged. We acknowledge here that we do not have a good metric to assess the optimality and feasibility for the SURF algorithm. The results for convergence shown in Fig. 7 is based on using the same optimality and feasibility criteria for SURF as in conventional reduced space. However, this plot suggests that the feasibility criteria favor the SURF algorithm, while the optimality criteria are disadvantageous for SURF. The reason for such a discrepancy can be attributed to the addition of a large number of residuals as constraints to the SURF algorithm and solving it to higher tolerances than other constraints in the problem. This affects the optimality and feasibility computed using the conventional approach.



(a) (b)

Fig. 7 Optimization convergence for RS, MFS and SURF. (a) Optimality. (b) Feasibility.

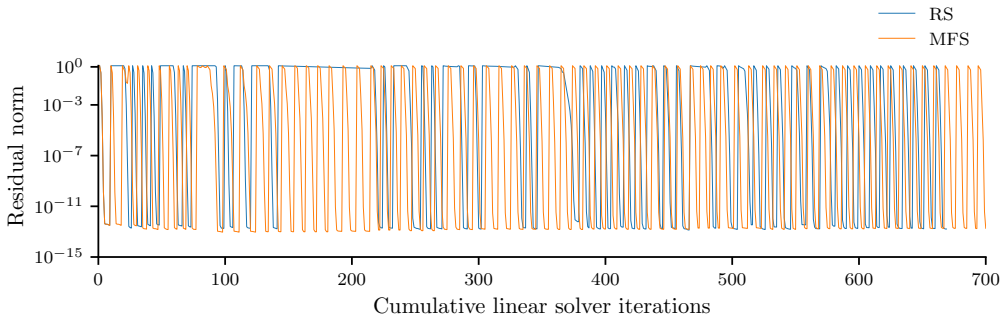


Fig. 8 Convergence with linear solver iterations for RS and MFS.

Fig. 8 shows the convergence of the nonlinear solvers with linear solver iterations till the solution is sufficiently converged. We observe from Fig. 6 and Fig. 8 that both RS and MFS converged to the same solution in roughly the same number of linear iterations, which is approximately 700. We also note here that both RS and MFS converged sufficiently at the 25th iteration from observing the optimized structures during optimization. As mentioned earlier, this is not captured accurately by the optimality and feasibility figures seen from Fig. 7. Altogether, these results point in the direction that our proposed SQP-based SURF algorithm is equivalent to the reduced-space SQP algorithm when we set the highest tolerances on the nonlinear solvers.

Fig. 9 Optimized structure using SURF.

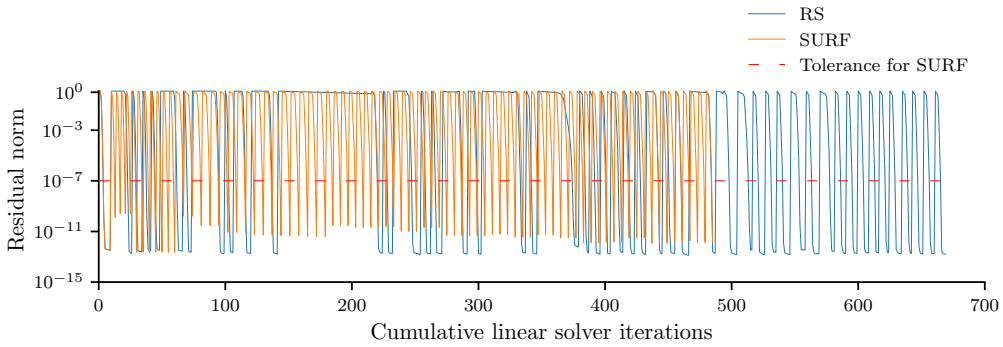


Fig. 10 Convergence with linear solver iterations for RS and SURF.

The other goal of this study was to measure the improvement in the total number of linear iterations needed for the convergence of the nonlinear solvers. To observe this, we need to relax the tolerances on the nonlinear solvers when using the proposed SQP algorithm. Therefore, we now perform line 1 in Algorithm. 5 inexactly to see if we can get any benefit from using a hybrid algorithm. For this study, we set a fixed tolerance of 1e-7 on the nonlinear solvers and run the hybrid SURF algorithm. The algorithm converged to the solution in Fig. 9 in again 25 iterations, the convergence of optimization is shown in Fig. 7. However, from Fig. 10, we see approximately 30 percent reduction, compared to reduced-space, in the total linear iterations required for solving the nonlinear systems. This can be seen as empirical evidence for the hypothesis that the proposed algorithm is not only a hybrid algorithm in a general-constrained optimization setting but also can provide practical benefits when applied to optimization problems that contain implicit state variables that are expensive to compute.

We conclude by noting that we applied a pure full-space SQP algorithm on this problem, and it was unsuccessful in converging to a reasonable solution which again strengthens the reason why FS is not widely used in the optimization of large and complex systems. We also note that there was no improvement in the overall optimization time with SURF since the optimization problem becomes larger and more complex with the addition of new design variables and constraints, and the basic optimizer developed for this study did not scale well with the number of design variables. However, we expect to see an improvement in the optimization times with a good quality optimizer like SNOPT (sparse nonlinear optimizer) or IPOPT (interior point optimizer) when implemented with the proposed SURF algorithm.

V. Conclusion

In this paper, we proposed and investigated a new hybrid architecture that extends the existing SURF architecture to make it compatible with general inequality-constrained problems. We first proved the theoretical equivalence between SURF and RS for SQP methods in an equality-constrained setting. Leveraging this, we proposed that the same algorithm could work for inequality-constrained problems. We then investigated the equivalence of the proposed hybrid algorithm with its parent algorithm using a topology optmization problem. The results strongly suggest that the proposed SURF algorithm is hybrid of reduced-space and full-space algorithms for an SQP method. It also shows that a hybrid architecture can be highly beneficial in case of optimization problems with large number of implicitly computed state variables.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1917142.

References

[1] 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.

[2] 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.

[3] Jasa, J. P., Hwang, J. T., and Martins, J. R. R. A., "Open-source coupled aerostructural optimization using Python," *Structural and Multidisciplinary Optimization*, Vol. 57, No. 4, 2018, pp. 1815–1827. doi:10.1007/s00158-018-1912-8, URL `https://doi.org/10.1007/s00158-018-1912-8`.

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

[5] Jasa, J. P., Mader, C. A., and Martins, J. R. R. A., "Trajectory Optimization of Supersonic Air Vehicle with Thermal Fuel Management System," *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, 2018. doi:10.2514/6.2018-3884.

[6] Friedman, S., Ghoreishi, S. F., and Allaire, D. L., "Quantifying the impact of different model discrepancy formulations in coupled multidisciplinary systems," *19th AIAA non-deterministic approaches conference*, 2017, p. 1950. doi:10.2514/6.2017-1950.

[7] Chaudhuri, A., Lam, R., and Willcox, K., "Multifidelity Uncertainty Propagation via Adaptive Surrogates in Coupled Multidisciplinary Systems," *AIAA Journal*, 2017, pp. 235–249. doi:10.2514/1.J055678.

[8] Palar, P. S., and Shimoyama, K., "Polynomial-chaos-kriging-assisted efficient global optimization," *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, IEEE, 2017, pp. 1–8. doi:10.1109/SSCI.2017.8280831.

[9] Cook, L. W., Jarrett, J. P., and Willcox, K. E., "Extending Horsetail Matching for Optimization Under Probabilistic, Interval, and Mixed Uncertainties," *AIAA Journal*, 2017, pp. 849–861. doi:10.2514/1.J056371.

[10] Cook, L. W., Jarrett, J. P., and Willcox, K. E., "Horsetail matching for optimization under probabilistic, interval and mixed uncertainties," *19th AIAA non-deterministic approaches conference*, 2017, p. 0590. doi:10.2514/6.2017-0590.

[11] Bons, N. P., He, X., Mader, C. A., and Martins, J. R. R. A., "Multimodality in Aerodynamic Wing Design Optimization," *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Denver, CO, 2017. doi:10.2514/6.2017-3753.

[12] Lam, R., Poloczek, M., Frazier, P., and Willcox, K. E., "Advances in Bayesian Optimization with Applications in Aerospace Engineering," *2018 AIAA Non-Deterministic Approaches Conference*, 2018, p. 1656. doi:10.2514/6.2018-1656.

[13] Tracey, B. D., and Wolpert, D., "Upgrading from Gaussian Processes to Student's T Processes," *2018 AIAA Non-Deterministic Approaches Conference*, 2018, p. 1659. doi:10.2514/6.2018-1659.

[14] Cook, L. W., "Effective Formulations of Optimization Under Uncertainty for Aerospace Design," Ph.D. thesis, University of Cambridge, 2018. doi:10.17863/CAM.23427.

[15] Baptista, R., and Poloczek, M., "Bayesian Optimization of Combinatorial Structures," *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 462–471. URL http://proceedings.mlr.press/v80/baptista18a.html.

[16] Peherstorfer, B., Beran, P. S., and Willcox, K. E., "Multifidelity Monte Carlo estimation for large-scale uncertainty propagation," *2018 AIAA Non-Deterministic Approaches Conference*, 2018, p. 1660. doi:10.2514/6.2018-1660.

[17] Chauhan, S. S., and Martins, J. R. R. A., "Low-Fidelity Aerostructural Optimization of Aircraft Wings with a Simplified Wingbox Model Using OpenAeroStruct," *Proceedings of the 6th International Conference on Engineering Optimization, EngOpt 2018*, Springer, Lisbon, Portugal, 2018, pp. 418–431. doi:10.1007/978-3-319-97773-7_38.

[18] Chaudhuri, A., Jasa, J., Martins, J. R. R. A., and Willcox, K., "Multifidelity Optimization Under Uncertainty for a Tailless Aircraft," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference; AIAA SciTech Forum*, Orlando, FL, 2018. doi:10.2514/6.2018-1658.

[19] Mader, C. A., Kenway, G. K., Yildirim, A., and Martins, J. R., "ADflow: An Open-Source Computational Fluid Dynamics Solver for Aerodynamic and Multidisciplinary Optimization," *Journal of Aerospace Information Systems*, 2020, pp. 1–20.

[20] Ning, A., and Petch, D., "Integrated Design of Downwind Land-Based Wind Turbines Using Analytic Gradients," *Wing Energy*, Vol. 19, No. 12, 2016, p. 2137–2152. doi:10.1002/we.1972.

[21] Barrett, R., and Ning, A., "Integrated Free-Form Method for Aerostructural Optimization of Wind Turbine Blades," *Wind Energy*, Vol. 21, No. 8, 2018, pp. 663–675. doi:10.1002/we.2186.

[22] Zahle, F., Tibaldi, C., Pavese, C., McWilliam, M. K., Blasques, J. P. A. A., and Hansen, M. H., "Design of an Aeroelastically Tailored 10 MW Wind Turbine Rotor," *Journal of Physics: Conference Series*, Vol. 753, No. 6, 2016, p. 062008. URL http://stacks.iop.org/1742-6596/753/i=6/a=062008.

[23] Zahle, F., Sørensen, N. N., McWilliam, M. K., and Barlas, A., "Computational fluid dynamics-based surrogate optimization of a wind turbine blade tip extension for maximising energy production," *Journal of Physics: Conference Series*, Vol. 1037, The Science of Making Torque from Wind, Milano, Italy, 2018. doi:10.1088/1742-6596/1037/4/042013.

[24] McWilliam, M. K., Zahle, F., Dicholkar, A., Verelst, D., and Kim, T., "Optimal Aero-Elastic Design of a Rotor with Bend-Twist Coupling," *Journal of Physics: Conference Series*, Vol. 1037, No. 4, 2018, p. 042009. URL http://stacks.iop.org/1742-6596/1037/i=4/a=042009.

[25] Graf, P., Dykes, K., Damiani, R., Jonkman, J., and Veers, P., "Adaptive stratified importance sampling: hybridization of extrapolation and importance sampling Monte Carlo methods for estimation of wind turbine extreme loads," *Wind Energy Science (Online)*, Vol. 3, No. 2, 2018. doi:10.5194/wes-3-475-2018.

[26] Dykes, K., Damiani, R., Roberts, O., and Lantz, E., "Analysis of Ideal Towers for Tall Wind Applications," *2018 Wind Energy Symposium*, AIAA, 2018. doi:10.2514/6.2018-0999.

[27] Stanley, A. P. J., and Ning, A., "Coupled Wind Turbine Design and Layout Optimization with Non-Homogeneous Wind Turbines," *Wind Energy Science Discussions*, Vol. 2018, 2018, pp. 1–24. doi:10.5194/wes-2018-54, URL https://www.wind-energ-sci-discuss.net/wes-2018-54/.

[28] Thomas, J. J., Gebraad, P. M., and Ning, A., "Improving the FLORIS wind plant model for compatibility with gradient-based optimization," *Wind Engineering*, Vol. 41, No. 5, 2017, pp. 313–329.

[29] Barter, G. E., "Starting with WISDEM (TM): The Short Course," Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2020.

[30] Hegseth, J. M., Bachynski, E. E., and Martins, J. R., "Integrated design optimization of spar floating wind turbines," *Marine Structures*, Vol. 72, 2020, p. 102771.

[31] Nandagopal, R. A., and Narasimalu, S., "Multi-objective optimization of hydrofoil geometry used in horizontal axis tidal turbine blade designed for operation in tropical conditions of South East Asia," *Renewable Energy*, Vol. 146, 2020, pp. 166–180.

[32] Stanley, A. P., Ning, A., and Dykes, K., "Optimization of turbine design in wind farms with multiple hub heights, using exact analytic gradients and structural constraints," *Wind Energy*, Vol. 22, No. 5, 2019, pp. 605–619.

[33] Herrema, A. J., Kiendl, J., and Hsu, M.-C., "A framework for isogeometric-analysis-based optimization of wind turbine blade structures," *Wind Energy*, Vol. 22, No. 2, 2019, pp. 153–170.

[34] Satkauskas, I., Gaertner, E., Bortolotti, P., Barter, G., and Graf, P. A., "Wind Turbine Rotor Design Optimization Using Importance Sampling," *AIAA Scitech 2020 Forum*, 2020, p. 1953.

[35] Hwang, J. T., Lee, D. Y., Cutler, J. W., and Martins, J. R., "Large-scale multidisciplinary optimization of a small satellite's design and operation," *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, 2014, pp. 1648–1663.

[36] Gray, J. S., Hearn, T. A., Moore, K. T., Hwang, J., Martins, J., and Ning, A., "Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO," *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2014, p. 2042.

[37] Ha, T. H., Lee, K., and Hwang, J. T., "Large-scale design-economics optimization of eVTOL concepts for urban air mobility," *AIAA Scitech 2019 Forum*, 2019, p. 1218.

[38] Ha, T. H., Lee, K., and Hwang, J. T., "Large-scale multidisciplinary optimization under uncertainty for electric vertical takeoff and landing aircraft," *AIAA Scitech 2020 Forum*, 2020, p. 0904.

[39] Ha, T., "System-level performance investigation of eVTOL concepts using large-scale DEP-focused design optimizations," Ph.D. thesis, UC San Diego, 2020.

[40] Chauhan, S. S., and Martins, J. R., "Tilt-wing eVTOL takeoff trajectory optimization," *Journal of Aircraft*, Vol. 57, No. 1, 2020, pp. 93–112.

[41] Joshy, A. J., and Hwang, J. T., "A new architecture for large-scale system design optimization," *AIAA AVIATION 2020 FORUM*, 2020, p. 3125. doi:https://doi.org/10.2514/6.2020-3125.

[42] 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.

[43] Hwang, J. T., Jain, A. V., and Ha, T. H., "Large-scale multidisciplinary design optimization–review and recommendations," *AIAA Aviation 2019 Forum*, 2019, p. 3106. doi:https://doi.org/10.2514/6.2019-3106.

[44] Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N., "The FEniCS Project Version 1.5," *Archive of Numerical Software*, Vol. 3, No. 100, 2015. doi:10.11588/ans.2015.100.20553.

[45] Yan, J., Xiang, R., Kamensky, D., Tolley, M. T., and Hwang, J. T., "Topology optimization with automated derivative computation," *Structural and Multidisciplinary Optimization (Accepted)*, 2021.

[46] 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.

[47] 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`.