Robust Schur-Complement Solvers and Large-Scale Multidisciplinary Design Optimization

M.A. Saja Abdul-Kaiyoom*, Anil Yildirim†, and Joaquim R. R. A. Martins‡
University of Michigan, Ann Arbor, Michigan, United States

Multidisciplinary models are composed of numerous implicit systems that are solved in a coupled manner. A common challenge arises when an individual discipline's Jacobian is non-invertible, which results in the coupled multidisciplinary model representing a saddle-point problem. In general, the small-scale multidisciplinary saddle-point problems are solved with the coupled Newton's method. Nevertheless, coupled solver approaches are difficult to build and have robustness problems. In addition, they necessitate solving large coupled linear systems in large-scale applications. In this work, we use the newly developed nonlinear and linear Schur complement (SC) solvers, appropriate for multidisciplinary models based on computational fluid dynamics (CFD) to solve these saddle-point problems. The SC solvers are not susceptible to the robustness problems of conventional coupled solutions since they leverage specialized CFD solvers. Thanks to these solvers, the conventional constrained optimization, which has a saddle-point system in its Jacobian, can be transformed into unconstrained optimizations. In this work, we demonstrate several applications of Schur-complement-based optimizations of CFD-based saddle-point problems: aerodynamic shape optimization (ASO) of a wing and coupled aeropropulsive design optimization of a podded propulsor. In the ASO cases, the Schur-complement solvers-based optimizations outperform the conventional constrained optimizations. Although the conventional optimization in the coupled aeropropulsive design optimization problem outperforms the SC solvers-based optimization, it requires an effective scaling of the design variables of the boundary conditions and the constraints. SC solvers-based optimizations always provide a feasible design at each design iteration, whereas the conventional approach does not. Furthermore, in large-scale nonlinear saddle-point problems, the nonlinear SC solver offers an alternative way to obtain a feasible solution for a given design instead of carrying out a modest optimization in order to get feasible solutions. In the CFD-based saddle-point applications, SC solvers will play an important role in the future because they are robust and can solve the saddle-point problem using the native solvers of the specialized CFD model.

Nomenclature

A_{ff}	=	fan-face area
C_D	=	drag coefficient
C_L	=	lift coefficient
C_p	=	pressure coefficient
D	=	drag force
F	=	force
$\dot{m}$	=	mass flow rate
M	=	Mach number
p	=	pressure
P_{total}	=	total shaft fan power
r	=	residual function

*Ph. D. Candidate, Department of Aerospace Engineering.
†Post-doctoral Research Fellow, Department of Aerospace Engineering.
‡Professor, Department of Aerospace Engineering.

T	=	temperature
t	=	thickness
$\vec{V}$	=	velocity
v	=	volume
X	=	design variables
x_{shape}	=	shape design variables
x_{twist}	=	twist design variables
α	=	angle of attack
Accronyms	=	
ASO	=	aerodynamic shape optimization
BC	=	boundary condition
CFD	=	computational fluid dynamics
DV	=	design variable
FPR	=	fan pressure ratio
MDO	=	multidisciplinary design optimization
RANS	=	Reynolds-averaged Navier–Stokes
SC	=	Schur-complement
XDSM	=	eXtended Design Structure Matrix
Subscripts and Superscripts	=	
$(\cdot)^*$	=	target value in optimization problems
$(\cdot)^{\text{aero}}$	=	value in the aerodynamic model
$(\cdot)^{\text{prop}}$	=	value in the propulsion model
$(\cdot)_{\text{ff}}$	=	quantity at the fan face
$(\cdot)_{\text{fe}}$	=	quantity at the fan exit
$(\cdot)_{\text{t}}$	=	total quantity
$(\cdot)_{\text{s}}$	=	static quantity

I. Introduction

Linear and nonlinear systems that arise from multidisciplinary models are challenging to solve [1]. A saddle-point problem arises when a computational component or the governing equations for a specific discipline cannot be solved on their own [2]. The structure of these models makes the hierarchical and BGS-based solvers inapplicable to these types of scenarios. The general approach to solve saddle point problems is the fully coupled Newton's method. Nevertheless, the fully coupled Newton's method is not necessarily the most efficient one. Even in cases when Newton's method provides quadratic convergence near the final solution, it may not necessarily be the most computationally efficient approach because it necessitates solving a large linear system at each solver iteration.

The 2×2 block saddle-point system can be used to illustrate this problem in its most general form:

$$\begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{1}$$

In the coupled nonlinear system, the partial derivatives of the residuals of the model with respect to its states are contained in the Jacobian matrix, which is expressed as

$$\begin{bmatrix} \dfrac{\partial r_1}{\partial u_1} & \dfrac{\partial r_1}{\partial u_2} \\ \dfrac{\partial r_2}{\partial u_1} & 0 \end{bmatrix} \begin{bmatrix} \Delta u_1^i \\ \Delta u_2^i \end{bmatrix} = \begin{bmatrix} -r_1(u_1^i, u_2^i) \\ -r_2(u_1^i, u_2^i) \end{bmatrix}, \tag{2}$$

where the zero block is the result of several balance residuals and $\partial r_1 / \partial u_1$ is the coupled Jacobian of one or more sets of governing equations.

Numerous applications of design optimization involve saddle-point problems. Two groups (coupled and decoupled) of algorithms exist for addressing saddle-point problems [3]. One possible solution for solving large-scale saddle-point

problems is the Schur-complement (SC) method, which is a decoupled approach. There have been significant efforts to use SC method for preconditioning approaches [4–14]. However, only a few studies have addressed the saddle-point problems in multidisciplinary models [2, 15]. Abdul-Kaiyoom et al. [16] developed nonlinear Schur complement (NSC) and linear Schur complement (LSC) solvers to overcome the challenges posed by the fully-coupled Newton's approach and to address the problem with the BGS-based solvers. These SC solvers can be used to circumvent this drawback of the BGS-based solvers and avoid the difficulties posed by the fully-coupled Newton's approach. In addition, because the SC solvers leverage specialized disciplinary solvers, they are less prone to robustness issues. In this paper, we demonstrate several applications of SC-based optimizations to overcome the drawbacks of current solvers in saddle-point problems.

Many saddle-point problems arise from computational fluid dynamics (CFD) based multidisciplinary applications. The aerodynamic shape optimization (ASO) of aircraft configurations to minimize drag or some other performance metric that depends on drag, subject to a target lift constraint is a well-known example [17]. Typically, the target lift value is achieved by enforcing the equality constraint in the optimization's formulation. However, it can also be achieved by changing the angle of attack at each design iteration implicitly to achieve the lift target. In this formulation, the target lift value can be imposed using a *balance equation* with its own residual, where the angle of attack variable is included in the model states. Because the balance residual itself does not explicitly depend on the angle of attack value, the diagonal sub-block of the Jacobian matrix is zero and thus non-invertible. As a result, this formulation is not widely used. The trim balance for aerostructural optimizations [18] and the continuation balance for CFD models with powered boundary conditions (BCs) to model propulsors [19] are the other common CFD-based multidisciplinary models that result in saddle point problems.

In the coupled aeropropulsive design problems, the propulsion system effects are incorporated into the CFD model for the coupled aeropropulsive model through the use of powered BCs. With the BC formulation, we satisfy several conservation quantities between the CFD and the propulsion models across the fan through balance residuals to model the coupled aeropropulsive system. In conventional optimizations, these residuals are enforced as constraints in the problem formulation because these residuals are not explicitly dependent on the state values and the diagonal sub-block of the Jacobian matrix is zero and thus non-invertible. It is evident from these examples that saddle-point problems arise in many design optimization applications, in single-disciplinary and multidisciplinary optimizations. Thus, it is crucial to address these saddle-point problems in design optimization to enable the flexible formulation of simulation-based optimization problems.

In this study, we use the newly developed NSC and LSC solvers to circumvent the difficulties posed by the fully-coupled Newton's approach and to address this issue with the BGS-based solvers. The solvers were implemented in OpenMDAO [20]. To automate the analytic derivative computation of coupled models with explicit and implicit model components, OpenMDAO utilizes the MAUD architecture [21]. The NSC solver in forward mode was introduced in coupled aeropropulsive design analysis with powered boundary conditions by Yildirim et al. [2]. Then, the NSC solver was used in reverse mode in an aeroelastic coupling within a trim equilibrium analysis [15]. They showed that the NSC solver successfully converged the saddle-point multidisciplinary problem. In this work, we use both NSC and LSC solvers to perform optimizations and show the benefits of them.

In this paper, we will show several applications of Schur-based ASO and coupled aeropropulsive design optimizations. For different linear and nonlinear solver tolerances, we will compare the SC-based optimization with the default lift-constrained optimization in the ASO of a wing. Similarly, we will compare the SC-based optimization with the default constrained optimization to show the benefits of SC solvers in the coupled aeropropulsive design optimization.

In this paper, we begin by outlining the mathematical formulation of the SC-based solvers in Sec. II. In Section III, we perform two different CFD-based design optimization to demonstrate the effectiveness of SC solvers. Finally, we present the conclusions of this work in Section IV.

II. Schur Complement Solver

A. Nonlinear Schur Complement Solver

The partial derivatives of the residuals with respect to the states are dealt with in the NSC solver There are two ways to compute these partial derivatives: forward mode and reverse mode. The formulation for the solver differs depending on the mode. The further explanation of the solvers is detailed in a previous publication by the current

authors [2, 15, 16].

1. Forward mode

A typical multidisciplinary model structure that can take advantage of this solver has a CFD solver in the first system (r_1), as will be shown in more detail in the following section. As a result, there are a lot more states in the first system than in the second (r_2). The cost of linear solutions with the Jacobian of each system is likewise influenced by this factor; the linear solutions with the Jacobian of the first system are substantially more expensive than the second. Computational performance is therefore improved by minimizing the number of linear solutions with the first system. This methodology is not limited to solving problems with CFD solvers; any system that has the ability to have several components, each with its unique nonlinear solution, can be the first system. By combining these two concepts, we may solve $r_1 = 0$ with a constant u_2, which is the same as the nonlinear BGS update for the first system. Thus, the Newton update formula for this system becomes

$$\begin{bmatrix} \frac{\partial r_1}{\partial u_1} & \frac{\partial r_1}{\partial u_2} \\ \frac{\partial r_2}{\partial u_1} & \frac{\partial r_2}{\partial u_2} \end{bmatrix} \begin{bmatrix} \Delta u_1^i \\ \Delta u_2^i \end{bmatrix} = \begin{bmatrix} -r_1(u_1', u_2^i) \\ -r_2(u_1', u_2^i) \end{bmatrix} = \begin{bmatrix} 0 \\ -r_2(u_1', u_2^i) \end{bmatrix}, \tag{3}$$

where the intermediate state u_1' satisfies $r_1(u_1', u_2^i) = 0$.

By rearranging the first row of Eq. 3, we get

$$\Delta u_1^i = -\left(\frac{\partial r_1}{\partial u_1}\right)^{-1} \frac{\partial r_1}{\partial u_2} \Delta u_2^i. \tag{4}$$

Then using Eq. 4 in the second row of Eq. 3, Δu_2^i is computed by solving

$$\left(\frac{\partial r_2}{\partial u_2} - \frac{\partial r_2}{\partial u_1}\left(\frac{\partial r_1}{\partial u_1}\right)^{-1} \frac{\partial r_1}{\partial u_2}\right) \Delta u_2^i = -r_2(u_1', u_2^i), \tag{5}$$

which requires n_{u_2} additional linear solutions with the matrix $\partial r_1 / \partial u_1$.

After taking the update Δu_2^i, the final step in this method would be to compute a final update for u_1' to account for the changes in u_2. Δu_1^i can be found from 1st row of Eq. 3 and becomes

$$\frac{\partial r_1}{\partial u_1} \Delta u_1^i = -r_1(u_1', u_2^i) - \frac{\partial r_1}{\partial u_2} \Delta u_2^i = -\frac{\partial r_1}{\partial u_2} \Delta u_2^i. \tag{6}$$

That being said, the right side of this formulation is just a first-order approximation of r_1 at $(u_1', u_2^i + \Delta u_2^i) = (u_1', u_2^{i+1})$. Consequently, we may update the u_2 vector and calculate the actual nonlinear residual $r_1(u_1', u_2^{i+1})$ instead of depending on this first-order approximation. Using the first system's specialized nonlinear solver, we can finally find u_1^{i+1}. We can converge r_1 using this method, which is a necessary first step to get Eq. 3 in the first place, as opposed to utilizing a single nonlinear solver iteration with the first system. To update the state vector u_1 of the first system, the solver solely uses specialized nonlinear solvers of sub-components of the first system. Therefore, it avoids introducing robustness problems that arise from applying Newton's approach in the absence of globalization. The intermediate states of the first system that solves $r_1 = 0$ for given u_2^i can be simply carried over as the u_1 vector that arises from the previous iteration. With this approach, u_1' in each iteration is replaced with u_1^i. As a result, we solve the u_1 using the first system's specialized nonlinear solvers and then update the u_2 using Eq. 5. Then, we update u_1 using the first system's specialized nonlinear solvers for the updated u_2. This process is repeated until convergence criteria are reached in the NSC solver.

2. Reverse mode

The reverse mode formulation for the NSC solver is derived in a manner similar to the forward mode. However, the coupled Jacobian matrix is transposed in reverse mode. As a result, Eq. 3, the coupled Newton system, becomes

$$\begin{aligned}\begin{bmatrix}(\Delta u_1^i)^T & (\Delta u_2^i)^T\end{bmatrix}\begin{bmatrix}\frac{\partial r_1}{\partial u_1}^T & \frac{\partial r_2}{\partial u_1}^T \\ \frac{\partial r_1}{\partial u_2}^T & \frac{\partial r_2}{\partial u_2}^T\end{bmatrix} &= -r^T \\ &= \begin{bmatrix}-r_1(u_1', u_2^i)^T & -r_2(u_1', u_2^i)^T\end{bmatrix} \\ &= \begin{bmatrix}0 & -r_2(u_1', u_2^i)^T\end{bmatrix}.\end{aligned} \tag{7}$$

The SC of the coupled Jacobian matrix is computed to get the updates to the states for the second system, as we covered in Section II.A.1. Rearranging Eq. 7's first column becomes

$$(\Delta u_1^i)^T = -(\Delta u_2^i)^T \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \left(\frac{\partial r_1}{\partial u_2}^T\right). \tag{8}$$

Thus, the second column of Eq. 7 becomes

$$\left((\Delta u_2^i)^T \frac{\partial r_2}{\partial u_2}^T - (\Delta u_2^i)^T \frac{\partial r_1}{\partial u_2}^T \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \frac{\partial r_2}{\partial u_1}^T\right) = -r_2(u_1', u_2^i)^T. \tag{9}$$

Following that, the Δu_2^i can be obtained by taking the transpose of Eq. 9 and solving

$$\left(\frac{\partial r_2}{\partial u_2}^T - \frac{\partial r_1}{\partial u_2}^T \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \frac{\partial r_2}{\partial u_1}^T\right)^T \Delta u_2^i = -r_2(u_1', u_2^i), \tag{10}$$

which also requires n_{u_2} additional linear solutions with the matrix $(\partial r_1/\partial u_1)^T$.

The next step in this technique would be to compute a final update for u_1' to account for the changes in u_2 after taking the update Δu_2^i, which is similar to Eq. 3 in the forward mode. Nevertheless, we update the u_2 vector and calculate the true nonlinear residual $r_1(u_1', u_2^{i+1})$ instead of using the first-order approximation of r_1 at $(u_1', u_2^i + \Delta u_2^i) = (u_1', u_2^{i+1})$. We employ the first system's specialized nonlinear solver, as detailed in Section II.A.1.

B. Linear Schur Complement Solver

Similar to the NSC solver, the LSC solver can function in both forward and reverse directions. In situations where there are more output functions than design variables, the forward mode is effective. Conversely, the reverse mode works well when there are more design variables than output functions [22, Section 6.7.3].

With regard to the explicit and implicit reliance of f on x, we are interested in computing the jacobian $(\mathrm{d}f/\mathrm{d}x)$ in optimizations. x represents the design variables, and f represents the functions of interest. Consequently, f's total derivative with regard to x and the state variables (u) is as follows:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial u}\left(\frac{\partial r}{\partial u}\right)^{-1} \frac{\partial r}{\partial x}, \tag{11}$$

which results in partial derivatives only on the right-hand side [22, Section 6.7.2]. There are two methods to solve the linear system on the right side: the direct method (forward mode) and the adjoint method (reverse mode) [22, Section 6.7.2]. The linear system $(\partial r/\partial u)^{-1}\partial r/\partial x$ is taken to be solved in the forward mode. In the reverse mode, the linear system is solved using $\partial f/\partial u(\partial r/\partial u)^{-1}$. This linear system can be transposed to obtain the linear solution as follows: $(\partial f/\partial u)^T\psi = (\partial r/\partial u)^T$. The vectors known as adjoints are represented by ψ.

1. Forward mode

The linear system in Eq. 11 is solved in the forward mode using $\partial r/\partial x$. Consequently, the forward-mode linear system can be expressed as

$$\frac{\partial r}{\partial u}\phi = \frac{\partial r}{\partial x}. \tag{12}$$

This equation (Eq. 12) can be formulated for two systems, just like the NSC solver. As a result, the linear system turns into

$$\begin{bmatrix} \frac{\partial r_1}{\partial u_1} & \frac{\partial r_1}{\partial u_2} \\ \frac{\partial r_2}{\partial u_1} & \frac{\partial r_2}{\partial u_2} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial r_1}{\partial x} \\ \frac{\partial r_2}{\partial x} \end{bmatrix}. \tag{13}$$

By rearranging the first row of Eq. 13, we get

$$\phi_1 = \left(\frac{\partial r_1}{\partial u_1}\right)^{-1} \left(\frac{\partial r_1}{\partial x} - \frac{\partial r_1}{\partial u_2}\phi_2\right). \tag{14}$$

The result of multiplying both sides by $\partial r_2/\partial u_1$ and substituting in the second row of Eq. 13 is

$$\left(\frac{\partial r_2}{\partial u_2} - \frac{\partial r_2}{\partial u_1}\left(\frac{\partial r_1}{\partial u_1}\right)^{-1}\frac{\partial r_1}{\partial u_2}\right)\phi_2 = \frac{\partial r_2}{\partial x} - \frac{\partial r_2}{\partial u_1}\left(\frac{\partial r_1}{\partial u_1}\right)^{-1}\frac{\partial r_1}{\partial x}. \tag{15}$$

ϕ_2 is first solved using this formulation and then it is substituted into Eq. 14 to get ϕ_1 . The NSC solver in the forward mode has already solved Eq. 5, which has identical linear solutions with $\partial r_1/\partial u_1$ for $\partial r_1/\partial u_2$. Therefore, it simply needs to initialize the NSC solver's solution in the LSC solver. When the linear system in the NSC solver is sufficiently converged, the linear system in the LSC solver can be solved very quickly.

2. Reverse mode

The linear system that is solved in the reverse mode differs from the forward mode to compute the derivatives. The linear system with $\partial f/\partial u$ on the right-hand side of Eq. 11 is solved using the adjoint method (reverse mode). When solving the adjoint term in the reverse mode, it can be shown as

$$\psi^T \equiv \frac{\partial f}{\partial u}\left(\frac{\partial r}{\partial u}\right)^{-1}. \tag{16}$$

By multiplying both sides by $\partial r/\partial u$ and transposing the entire equation, we may obtain the adjoint equation, which is

$$\frac{\partial r}{\partial u}^T \psi = \frac{\partial f}{\partial u}^T, \tag{17}$$

where ψ are the adjoint vectors. Thus, the linear system becomes

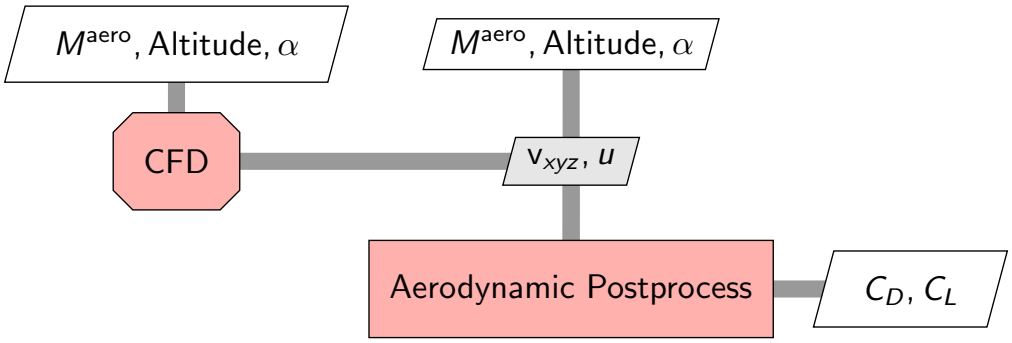
$$\begin{bmatrix} \frac{\partial r_1}{\partial u_1}^T & \frac{\partial r_2}{\partial u_1}^T \\ \frac{\partial r_1}{\partial u_2}^T & \frac{\partial r_2}{\partial u_2}^T \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial u_1}^T \\ \frac{\partial f}{\partial u_2}^T \end{bmatrix}. \tag{18}$$

By rearranging the first row of Eq. 18, we obtain

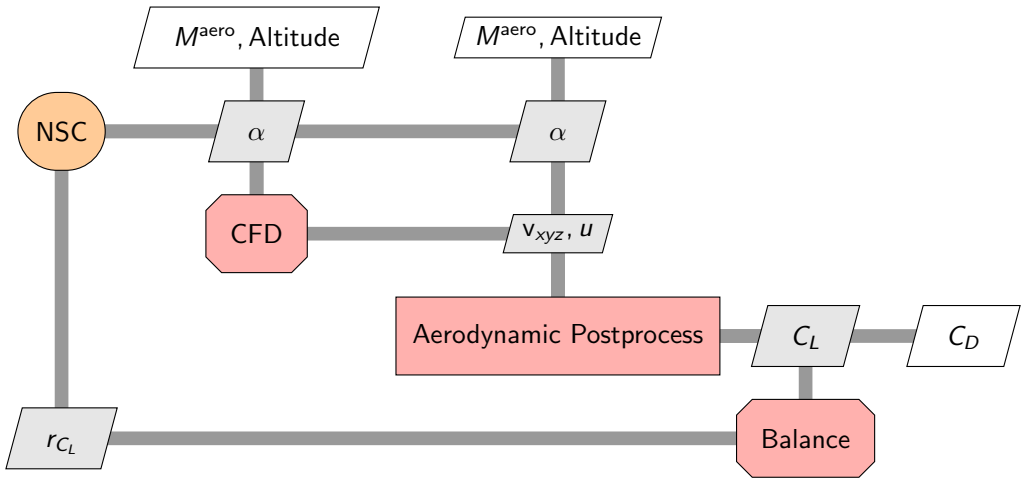
$$\psi_1 = \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \left(\frac{\partial f}{\partial u_1}^T - \frac{\partial r_2}{\partial u_1}^T \Psi_2\right). \tag{19}$$

Multiplying both sides by $(\partial r_1/\partial u_2)^T$ and substituting in the second row of Eq. 18 become

$$\left(\frac{\partial r_2}{\partial u_2}^T - \frac{\partial r_1}{\partial u_2}^T \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \frac{\partial r_2}{\partial u_1}^T\right)\psi_2 = \frac{\partial f}{\partial u_2}^T - \frac{\partial r_1}{\partial u_2}^T \left(\frac{\partial r_1}{\partial u_1}^T\right)^{-1} \frac{\partial f}{\partial u_1}^T. \tag{20}$$



(a) XDSM diagram of the MDA formulation with the conventional constrained optimization problem.



(b) XDSM diagram of the MDA formulation with the SC-based optimization problem.

Fig. 1 XDSM diagrams for the ASO of both the conventional and SC solvers-based optimizations. v_{xyz} are the volume coordinates and u are the state variables from the CFD solver (ADflow).

As with the forward mode, we may obtain ψ_1 by first solving for ψ_2 and then substituting this value into Eq. 19. Similar to the forward mode, the linear solution with $(\partial r_1/\partial u_1)^T$ for $(\partial r_2/\partial u_1)^T$ in the NSC solver can be used as an initial guess in the LSC solver when the NSC solver and LSC solver operate in the reverse mode. This will cause the linear system to converge quickly.

While using both NSC and LSC solvers in the same mode (either forward or reverse) is preferred, it is also possible for them to operate in different modes. This is because the linear solution from the NSC solver can be initialized in the LSC solver, which can quickly converge the linear system in the LSC solver. Four distinct mode combinations arise from this.

III. CFD-Based Application

In this section, we show the benefits of SC solvers for the ASO of a wing and a coupled aeropropulsive optimization of a podded propulsor. Both of these CFD-based applications are detailed in the following sections.

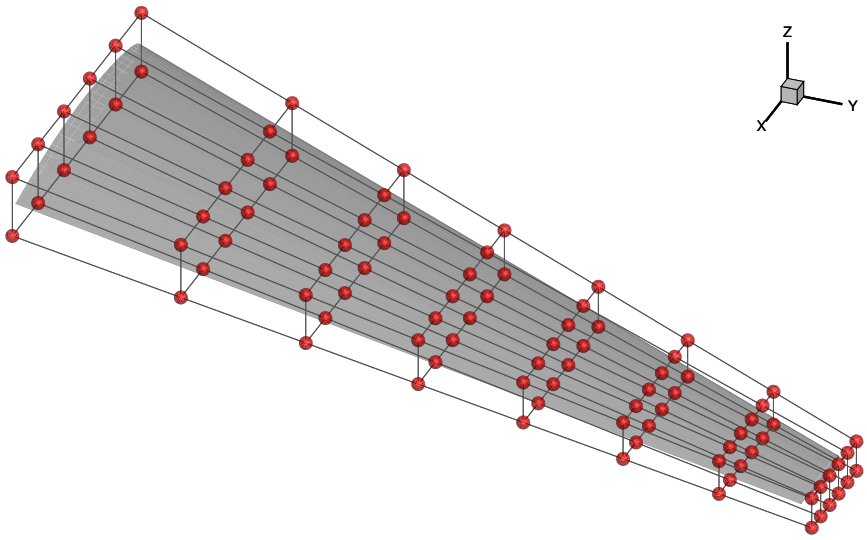


Fig. 2 Wing geometry and FFD control points.

A. Aerodynamic Shape Optimization of a Wing

In this section, we perform an ASO of a wing using SC solvers (NSC and LSC solvers). When dealing with conventional ASO problems, the wing's shape is modified to minimize drag and yet produce the desired lift under cruise conditions. The lift constraint is typically applied as an equality constraint in an ASO. However, we may solve for the lift at each design iteration with the use of SC solvers by enforcing *balance residual*, which eliminates the need to enforce the equality constraint. Figure 1 demonstrates the XDSM diagram for both these approaches. In the SC-based optimization, the angle of attack becomes a state variable. The following sections explain both conventional and SC-based optimizations.

1. Methodology

We use ADflow [23] as a CFD solver, which solves the RANS equations on structured multiblock and overset grids. In all of the results, we use the Spalart–Allmaras (SA) turbulence model [24]. ADflow computes the derivatives needed for gradient-based optimization [25] using an effective adjoint solver and uses a robust approximate Newton–Krylov solver algorithm [26]. Furthermore, we use the geometry parameterization module pyGeo [27] and the volume mesh warping module IDWarp [28]. We use the free form deformation (FFD) scheme [29] implemented in pyGeo [27], as illustrated in Fig. 2, to modify the shape of the wing.

Both of the problems defined in Figure 1 are implemented using the MPhys library, which is constructed using the OpenMDAO framework [20]. This means that in our ASO problems, we only need to compute the partial derivatives of each component. OpenMDAO then employs the modular analysis and unified derivatives (MAUD) architecture [21] to address the coupled derivative problem, especially useful for SC-based optimization.

2. Problem Formulation

The problem formulations of the typical lift-constrained and the residual enforced wing optimizations are shown in Table 1. The angle of attack, which is the coupling variable (u_2 in Eqs. 5, 10, 15, and 20), is not a design variable in the residual enforced wing optimizations. Moreover, there is also no equality constraint for lift. We solve the lift constraint as residuals using the NSC solver at each design iteration:

$$r_{C_L}(\alpha) = C_L - C_L^* = 0, \tag{21}$$

where C_L^* is the target lift coefficient. Table 1 demonstrates both of the problem formulations.

Table 1 The optimization aims to minimize the drag subject to lift and geometric constraints.

	Function/Variable	Description	Quantity
Conventional lift-constrained optimization:			
Minimize	C_d		
By varying	x_{shape}	Shape design variables	96
	x_{twist}	Twist design variables	7
	α	Angle of attack	1
Subject to	C_L=C_L^*	Lift coefficient	1
	$t \geq t_{\min}$	Thickness constraint	200
	$v \geq v_{\min}$	Volume constraint	1
Residual enforced optimization using SC solvers:			
Minimize	C_d		
By varying	x_{shape}	Shape design variables	96
	x_{twist}	Twist design variables	7
Subject to	$t \geq t_{\min}$	Thickness constraint	200
	$v \geq v_{\min}$	Volume constraint	1

For SC-based analyses and optimizations, there are 4 convergence-related parameters for the NSC solver: the relative nonlinear convergence of each CFD simulation (η_{CFD}), the relative linear convergence (η_{lin}) of each solution to obtain the linear solutions in Eq. 5 in forward mode or Eq. 10 in reverse mode, and the relative and absolute nonlinear convergence of the NSC solver (η_{rel} and η_{abs}). We set the absolute tolerance of NSC solver (η_{abs}) to 10^{-6} and the relative tolerance of NSC solver (η_{rel}) to 10^{-8}. In addition, the linear systems in the LSC solver (Eq. 15 in forward mode or Eq. 20 in reverse mode) are also solved to a similar tolerance to the linear systems in the NSC solver. The CFD is converged to a relative tolerance of 10^{-14} in both of the optimization problems in Table 1, and the adjoint linear system is solved to a tolerance of 10^{-14} in conventional lift-constrained optimizations and in Eq. 14 in forward mode or Eq. 19 in reverse mode for SC-based optimizations.

To generate practical designs, we include geometric constraints which include thickness and volume constraints. We choose a Mach number (M) of 0.8 and an altitude of 10, 000 meters. We use the sparse nonlinear optimizer (SNOPT) [30], a sequential quadratic programming (SQP) algorithm, through pyOptSparse interface [31].

3. Results

First, we perform a simple nonlinear analysis of the problem in Fig. 1b to solve for $C_L = 0.5$ using the NSC solver in reverse mode. We set the relative convergence of the nonlinear and linear residuals in the NSC solver to $\eta_{\text{CFD}} = 10^{-4}$

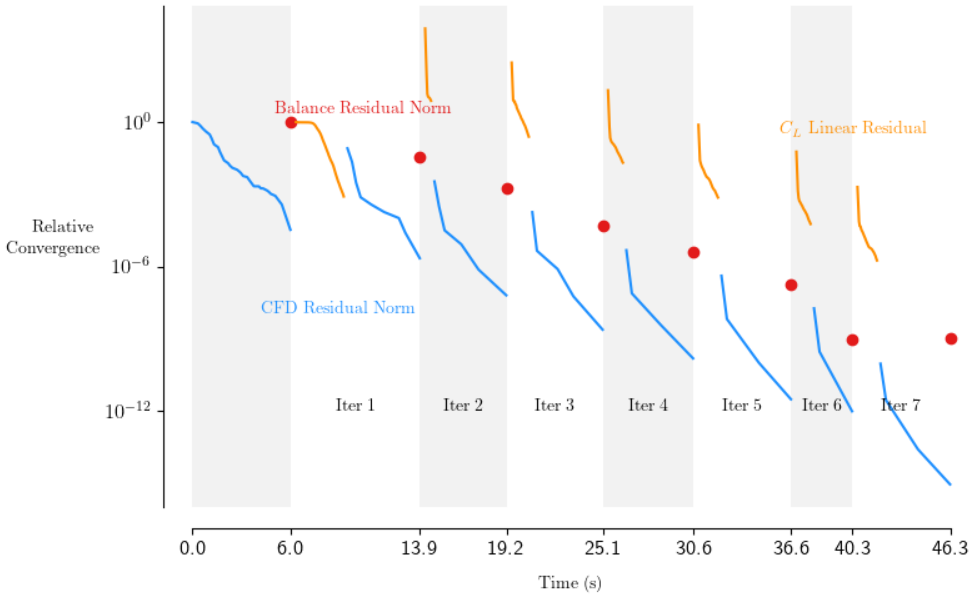


Fig. 3 MDA of C_L residual enforced case using the NSC solver.

and $\eta_{\text{lin}} = 10^{-3}$. Fig. 3 demonstrates the cost of each stage of the solver. We partially converge the CFD residuals prior to the initial NSC solver iteration, as delineated in Sec. II.A. Then, we solve the linear system needed for the update formula in Eq. 5 in forward mode or Eq. 10 in reverse mode for each iteration of the NSC solver. We restart this linear system from the previous solutions. We converge to a tolerance of η_{lin} with respect to the initial linear residual norm at the beginning of each solution. The CFD model is then reconverged by ANK solvers for the updated balance variable using the NSC solver. Approximately 15 orders of magnitude are reduced in the CFD model residuals and more than 6 orders of magnitude are reduced in the balance residuals when the solution procedure is repeated until a η_{abs} value of 10^{-6} or a η_{rel} value of 10^{-8} is obtained. The NSC solver offers an option to obtain a feasible solution for a given design instead of carrying out a modest optimization to get feasible solutions.

To demonstrate the efficacy of the SC solvers, we perform several optimizations. We only consider reverse mode for optimizations because the number of design variables is higher than the cost functions. Two distinct optimizations were carried out for the conventional wing optimization scenario in Table 1. In the conventional C_L constrained optimization, the scaling of the C_L constraint severely affects the performance. As a result, in this case, we carry out two optimizations: one in which we scale the C_L constraint and the other not. In the residual enforced cases, we choose different combinations of η_{CFD} and η_{lin} tolerances to accelerate convergence of the optimization. The optimality and feasibility criteria were set to 10^{-16}. All the optimizations were allowed to converge to the smallest possible optimality and feasibility tolerances. Table 2 explains the summary of optimizations. Figure 4 demonstrates C_D convergence with respect to the major iteration and time. In the conventional C_L constrained optimization, the scale of the C_L constraint has a major impact on the optimization cost. When the C_L constraint is scaled down, the equality constraint at the beginning of the optimization does not influence the merit function greatly, giving the optimizer more latitude to modify the design variables in an effort to reach the optimum. However, when this constraint is not scaled adequately, the cost of the optimization is more than twice as high as most of the optimization based on SC solvers. For a given η_{CFD}, when the tolerance for the linear system (η_{lin}) is higher, the optimization ends abruptly and reaches a less accurate solution. When the η_{lin} is higher (10^{-1} or 10^{-2}), the cost of the optimization is also higher. For the tolerances listed in Table 2, Two SC solver-based optimizations perform better than the conventional optimizations, including the scaled one. Conventional optimization scenarios required more iterations and took longer to converge than those two optimizations utilizing SC solvers in Table 2.

For this problem, we were able to identify the C_L constraint's scalability due to the problem's simplicity and our

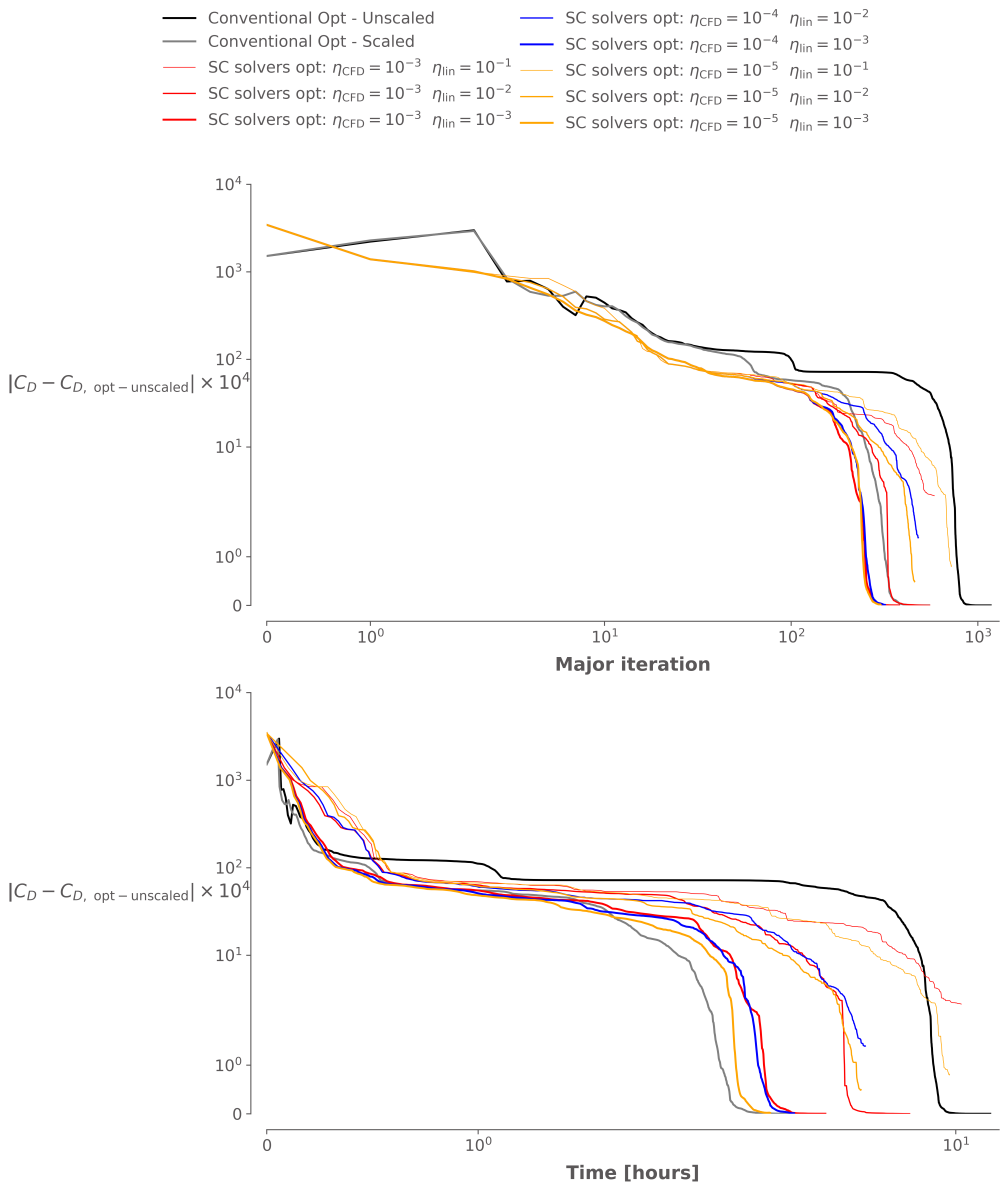


Fig. 4 C_D history with respect to the major iteration and time.

experience with it. It is extremely challenging to choose the right scaling numbers with more general and complex optimization problems. Nonetheless, the scalability problem is not encountered by the optimization that utilizes SC solvers. In addition, SC solvers-based optimization always provides a feasible design, whereas conventional optimization does not.

In this application, we showed that the SC solvers-based optimization can outperform the conventional constrained optimizations while avoiding the challenges of finding a good scaling value for constraints. It illustrates how balanced residual problems in multidisciplinary design optimization can be substituted with SC solvers.

Table 2 The optimization aims to minimize the drag subject to lift and geometric constraints.

Cases	Time	Scaling for C_L constraint	Optimal C_D	Optimal α
Conventional opt - unscaled	12 hours and 22 minutes	1	0.01832313	3.864°
Conventional opt - scaled	3 hours and 57 minutes	10^{-4}	0.01832313	3.864°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-3}$ $\eta_{\text{lin}} = 10^{-1}$	10 hours and 24 minutes	–	0.01832589	3.350°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-3}$ $\eta_{\text{lin}} = 10^{-2}$	7 hours and 39 minutes	–	0.01832313	3.858°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-3}$ $\eta_{\text{lin}} = 10^{-3}$	4 hours and 37 minutes	–	0.01832313	3.863°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-4}$ $\eta_{\text{lin}} = 10^{-2}$	5 hours and 52 minutes	–	0.01832454	3.501°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-4}$ $\eta_{\text{lin}} = 10^{-3}$	3 hours and 49 minutes	–	0.01832314	3.865°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-1}$	9 hours and 42 minutes	–	0.01832395	3.623°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-2}$	5 hours and 43 minutes	–	0.01832363	3.661°
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-3}$	3 hours and 17 minutes	–	0.01832315	3.851°

B. Coupled Aeropropulsive Design Optimization of a Podded Propulsor

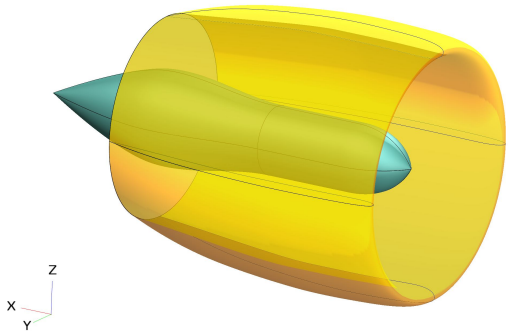
Our primary motivation for applying the SC solvers is to solve nonlinear systems arising from CFD-based models. In particular, we want to solve the nonlinear problems that result from using boundary conditions to model propulsors in CFD models. We also use the coupled aeropropulsive model as a benchmark case, explaining how the SC solvers are integrated into the rest of the solver hierarchy, and studying the solvers' performance.

1. Coupled Aeropropulsive Model

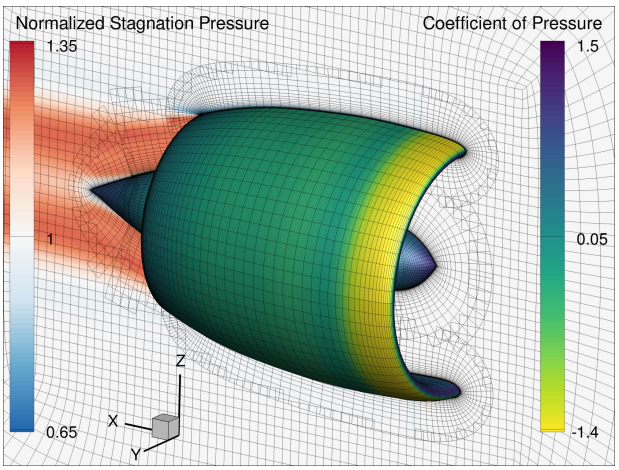
We use the BC version of the benchmark aeropropulsive model created in our earlier work [19] as the aeropropulsive model in this study. The benchmark model is a podded electric fan design that is based on the aft-propulsor of NASA's STARC-ABL concept [32], as illustrated in Fig. 5. Despite being a straightforward design, it illustrates the main difficulties in coupling a CFD solver to a propulsion model for a fully coupled aeropropulsive model. In this work, we exclusively investigate the BC version of the benchmark model since the nonlinear system arising from this version causes the Jacobian matrix to have a zero sub-block, making it impossible to use BGS-based solvers. The SC solvers, however, have no limitations in the same way. Powered BCs are used in the coupled aeropropulsive model to introduce the propulsion system effects into the CFD model. The BC technique is useful even though source-term formulations are more accurate for fan simulations [33]. The source-term formulation is not supported by all CFD solvers; for this reason, a BC formulation is more broadly applicable. Furthermore, although source-term formulations yield higher accuracy for parts like fans, they become computationally constrained when simulating an engine core, for which low-order cycle models can yield adequate accuracy [34]. Therefore, the BC version is important in coupled aeropropulsive problems and we use SC solvers to solve and optimize these problems.

2. Boundary Conditions

We match many coupling parameters within the BC model to maintain consistency between the aerodynamics and propulsion models, using the methodology proposed by Lamkin et al. [35]. The mass flow rate ($r_{\dot{m}}$), area (r_A), and velocity (r_V) between the propulsion models and the CFD must be satisfied across the fan using the BC version. Furthermore, the total net thrust requirement (r_F) is also computed in this residual evaluation. The static pressure at the fan face ($p^{\text{aero}}_{\text{s,ff}}$), and the total pressure ($p^{\text{aero}}_{\text{t,fe}}$), total temperature ($T^{\text{aero}}_{\text{t,fe}}$), and mach number ($M^{\text{prop}}_{\text{fe}}$) at the fan exit are tuned to achieve the conservation of these quantities. The conservation between the propulsion and aerodynamic



(a) Podded electric fan geometry in OpenVSP. The core and the nacelle are modeled using two separate components.



(b) CFD model of the podded fan. The CFD simulations use an overset mesh that contains a symmetry plane about the centerline of the propulsor. The contours on the symmetry plane show the normalized stagnation pressure values, and contours on the propulsor surfaces show the coefficient of pressure values.

Fig. 5 Geometry and the CFD model of the podded fan.

models is expressed as

$$\begin{aligned} r_{\dot{m}}\left(p^{\text{aero}}_{\text{s,ff}}, p^{\text{aero}}_{\text{t,fe}}, T^{\text{aero}}_{\text{t,fe}}, M^{\text{prop}}_{\text{fe}}\right) &= \dot{m}^{\text{aero}}_{\text{fe}} - \dot{m}^{\text{prop}}_{\text{fe}} = 0, \\ r_{A}\left(p^{\text{aero}}_{\text{s,ff}}, p^{\text{aero}}_{\text{t,fe}}, T^{\text{aero}}_{\text{t,fe}}, M^{\text{prop}}_{\text{fe}}\right) &= A^{\text{aero}}_{\text{fe}} - A^{\text{prop}}_{\text{fe}} = 0, \\ r_{V}\left(p^{\text{aero}}_{\text{s,ff}}, p^{\text{aero}}_{\text{t,fe}}, T^{\text{aero}}_{\text{t,fe}}, M^{\text{prop}}_{\text{fe}}\right) &= V^{\text{aero}}_{\text{fe}} - V^{\text{prop}}_{\text{fe}} = 0. \end{aligned} \tag{22}$$

The total net thrust requirement is also given as

$$\begin{aligned} r_{F}\left(p^{\text{aero}}_{\text{s,ff}}, p^{\text{aero}}_{\text{t,fe}}, T^{\text{aero}}_{\text{t,fe}}, M^{\text{prop}}_{\text{fe}}\right) = &\left(\dot{m}^{\text{aero}}_{\text{fe}} V^{\text{aero}}_{\text{fe}} + p^{\text{aero}}_{\text{s,fe}} A^{\text{aero}}_{\text{fe}}\right) \\ &- \left(\dot{m}^{\text{aero}}_{\text{ff}} V^{\text{aero}}_{\text{ff}} + p^{\text{aero}}_{\text{s,ff}} A^{\text{aero}}_{\text{ff}}\right) \\ &- D_{\text{total}} - F_{\text{target}} = 0, \end{aligned} \tag{23}$$

where F_{target} refers the target net thrust requirement. The fan model in the BC version between the propulsion and aerodynamic models is valid when the three residuals in Eq. 22 are met. These conservation residuals are enforced as equality constraints in the conventional coupled aeropropulsive optimization. Thanks to SC solvers, these balance residuals can be solved to achieve a feasible design at each design iteration.

3. Problem Formulation

In the coupled aeropropulsive optimization, we minimize the total shaft power (P_{total}) with respect to the constraints. We use 1-D thermodynamic cycle models developed with the pyCycle library [36] to estimate the total shaft power needed by the fan. Using a polytropic efficiency assumption and a tabular performance map, pyCycle sizes the compressor in the design mode. Using the FPR value obtained from the CFD simulations, we modify the fan's adiabatic efficiency until the compressor reaches the desired polytropic efficiency of 0.97. This number is taken from the NASA N+3 technology level reference propulsion system [37]. We recover the fully coupled system when the optimizer fulfills the conservation residuals for the BC version in Eq. 22.

In these types of typical optimization problems, these conservation quantities are constrained to achieve the feasible optimum. However, thanks to SC solvers, these conservation quantities are enforced as balance equations in a residual form to satisfy them at each design iteration. Figure 6 demonstrates these two different coupling problems for the BC version of the coupled aeropropulsive model. Table 3 shows the conventional coupled aeropropulsive optimization and SC-based optimization problem formulations. $p^{\text{aero}}_{\text{s,ff}}$, $p^{\text{aero}}_{\text{t,fe}}$, and $T^{\text{aero}}_{\text{t,fe}}$, which are the coupling variables (u_2 in Eqs. 5, 10, 15, and 20), are not a design variable in the SC solvers-based case. In addition, there are no conservation residual constraints, $r_{\dot{m}}$, r_A, and r_V. We solve them using SC solvers.

FPR constraint is imposed in both problem formulations. We also enforce geometric constraints to ensure practical designs. Thickness constraints for nacelle are included in the problem formulation to maintain the initial thicknesses. We use the same packages as described in Section III.A.1. However, the geometry of the podded propulsor is parameterized using OpenVSP [38], which is demonstrated in Figure 5a. We choose a Mach number of 0.785 and an altitude of 36, 000 feet.

For SC-based analyses and optimizations, we set the absolute tolerance of NSC solver (η_{abs}) to 10^{-4} and the relative tolerance of NSC solver (η_{rel}) to 10^{-6}. The CFD is converged to a relative tolerance of 10^{-12} in both of the optimization problems in Table 3, and the adjoint linear system is solved to a tolerance of 10^{-12} in conventional residual enforced optimizations and in Eq. 14 in forward mode or Eq. 19 in reverse mode for SC-based optimizations.

4. Results

We first analyze a simple nonlinear analysis of the problem shown in Fig. 6b using the NSC solver in reverse mode. We set the relative convergence of the nonlinear and linear residuals in the NSC solver to $\eta_{\text{CFD}} = 10^{-4}$ and $\eta_{\text{lin}} = 10^{-3}$ to demonstrate the cost of each stage of the solver in Fig 7. The NSC solver successfully converges the BC residuals and finds the required BC conditions. We partially converge the CFD residuals prior to the initial NSC solver iteration, as delineated in Sec. II.A. Similar to the case in Section III.A.3, we solve the linear system needed for the update formula in Eq. 5 in forward mode or Eq. 10 in reverse mode for each iteration of the NSC solver. Approximately 12 orders of magnitude are reduced in the CFD model residuals and more than 6 orders of magnitude are reduced in the balance residuals when the solution procedure is repeated until a η_{abs} value of 10^{-4} or a η_{rel} value of 10^{-6} is obtained. Instead of performing a modest optimization to achieve a feasible solution for the given design, the NSC solver offers an alternative to achieve a feasible nonlinear solution.

We perform two sets of optimizations, as detailed in Table 3. In this study, we only consider reverse mode for optimizations because the number of design variables is higher than the cost functions. For the conventional BC-constrained optimization, we perform 3 optimizations for different scaling of constraints and BC design variables. For the SC solvers-based optimization, we choose different combinations of η_{CFD} and η_{lin} tolerances to accelerate the convergence of the optimization. Figure 8 demonstrate the convergence of P_{total} for these optimizations. Table 4 summarizes these optimizations. In all of the optimizations, the major step limit parameter in SNOPT is set to 0.1 except for the $\eta_{\text{CFD}} = 10^{-5}$ and $\eta_{\text{lin}} = 10^{-2}$ case in SC solvers-based optimizations, where it is set to 0.01. In all of the cases, the optimality criteria and the feasibility criteria were set to 10^{-6} and 10^{-8}, respectively. The scaled conventional constrained optimizations and all of the SC solvers-based optimizations satisfied the optimality criteria except for the $\eta_{\text{CFD}} = 10^{-5}$ and $\eta_{\text{lin}} = 10^{-5}$ and $\eta_{\text{CFD}} = 10^{-6}$ and $\eta_{\text{lin}} = 10^{-2}$ cases in SC solvers-based optimization. These two SC solvers-based cases converged to the optimality tolerance in the order of 10^{-6}. The scaled conventional constrained optimizations converged to the feasibility tolerance successfully. SC solvers-based optimizations converged to the feasibility tolerance of at least in the order of 10^{-7}. The unscaled case in the conventional constrained optimization problem failed to converge after 13 hours, which is the time limit set for the optimization.

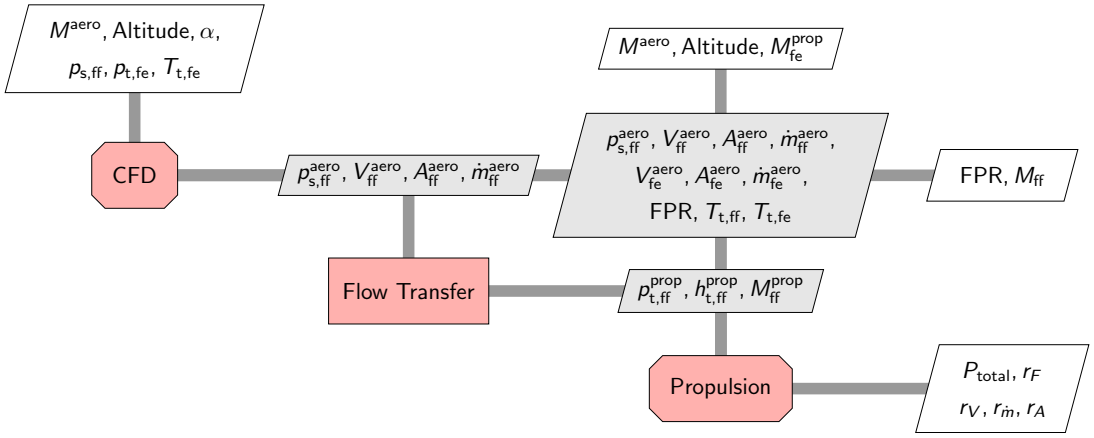
Based on our experiences from the previous studies [19, 39, 40], we scaled the BC design variables in scaled-1 case and both the BC design variables and the BC constraints in scaled-2 case. The scaled optimizations in the conventional constrained problem and the SC solvers-based optimizations converged to the same optimal point. In terms of cost, the scaled optimizations outperform the SC solver cases. To achieve this performance, however, finding an appropriate scaling through trial and error is needed. SC solvers are not limited by these scalability problems. Therefore, SC solvers are still a potential option for using them in these types of saddle-point problems. When the η_{lin} is 10^{-2} in each η_{CFD} tolerances, the cost of the optimization is higher because of the numerical errors.

In this application, we showed that the SC-based optimization outperforms the unscaled conventional optimization. Although the scaled cases are more efficient than the SC solvers-based optimizations, SC solvers provide an alternative to

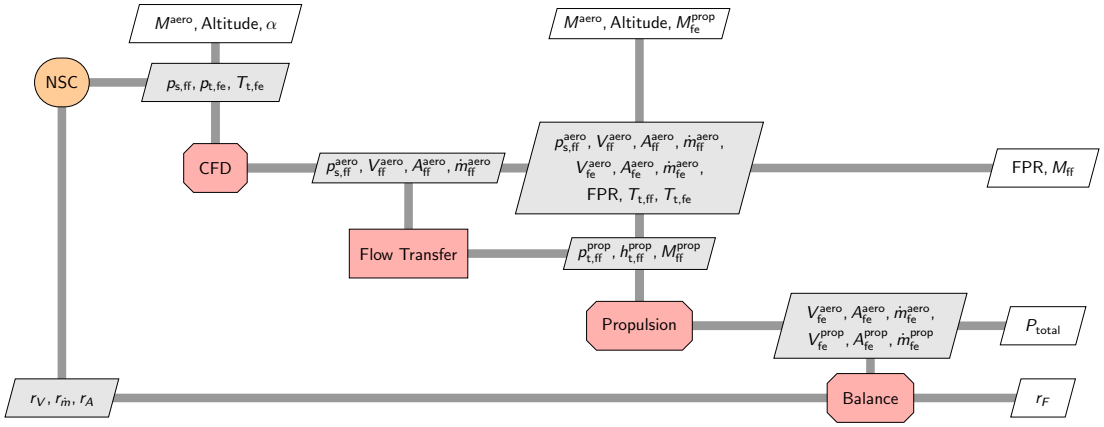
Table 3 Single-point optimization problem definition with the BC version. The differences compared to SC solvers-based optimizations in conventional optimizations are highlighted in red.

	Variable/Function	Description	Quantity
Conventional constrained optimization:			
minimize	P_{total}	Power required for the fan	1
By varying	$P^{\text{aero}}_{\text{s,ff}}$	Static pressure at fan face	1
	$P^{\text{aero}}_{\text{t,fe}}$	Total pressure at fan exit	1
	$T^{\text{aero}}_{\text{t,fe}}$	Total temperature at fan exit	1
	$M^{\text{prop}}_{\text{fe}}$	Mach number at fan exit	1
	x_{plug}	Plug shape	2
	x_{nacelle}	Nacelle shape	15
		Total	21
Subject to	$r_F = 0$	Net thrust residual	1
	$\text{FPR} = \text{FPR}^*$	Target FPR	1
	$M_{\text{ff}} \leq 0.6$	Upper limit of fan face Mach number	1
	$r_{\dot{m}} = 0$	Conservation of $\dot{m}$ between "aero" and "prop" models across the fan	1
	$r_A = 0$	Conservation of A between "aero" and "prop" models across the fan	1
	$r_V = 0$	Conservation of V between "aero" and "prop" models across the fan	1
	$0.99 \leq g_{\text{geo}} \leq 3.0$	Geometric thickness constraints	14
		Total	20
Residual enforced optimization using SC solvers:			
minimize	P_{total}	Power required for the fan	1
By varying	$M^{\text{prop}}_{\text{fe}}$	Mach number at fan exit	1
	x_{plug}	Plug shape	2
	x_{nacelle}	Nacelle shape	15
		Total	18
Subject to	$r_F = 0$	Net thrust residual	1
	$\text{FPR} = \text{FPR}^*$	Target FPR	1
	$M_{\text{ff}} \leq 0.6$	Upper limit of fan face Mach number	1
	$0.99 \leq g_{\text{geo}} \leq 3.0$	Geometric thickness constraints	14
		Total	17

achieve a feasible design at each iteration. Therefore, it illustrates how balanced residual problems in multidisciplinary design optimization can be substituted with SC solvers.



(a) XDSM diagram of the MDA formulation with the conventional constrained optimization problem.



(b) XDSM diagram of the MDA formulation with the SC-based optimization problem.

Fig. 6 XDSM diagram for the coupled aeropropulsive design optimization.

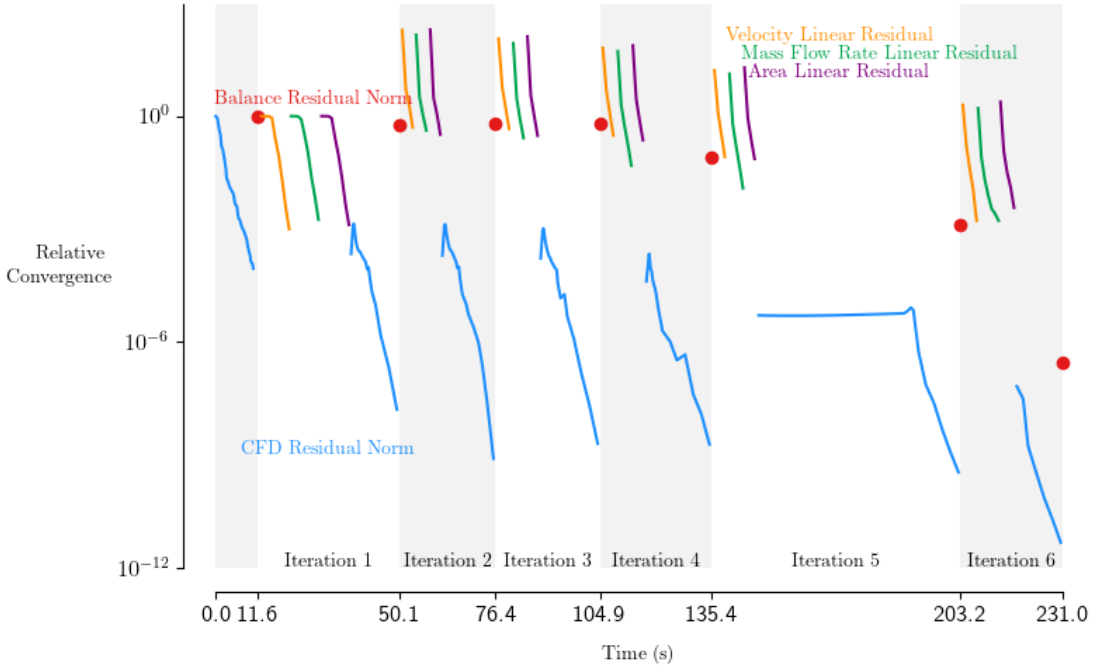


Fig. 7 MDA of residual enforced coupled aeropropulsive case using the NSC solver in rev mode.

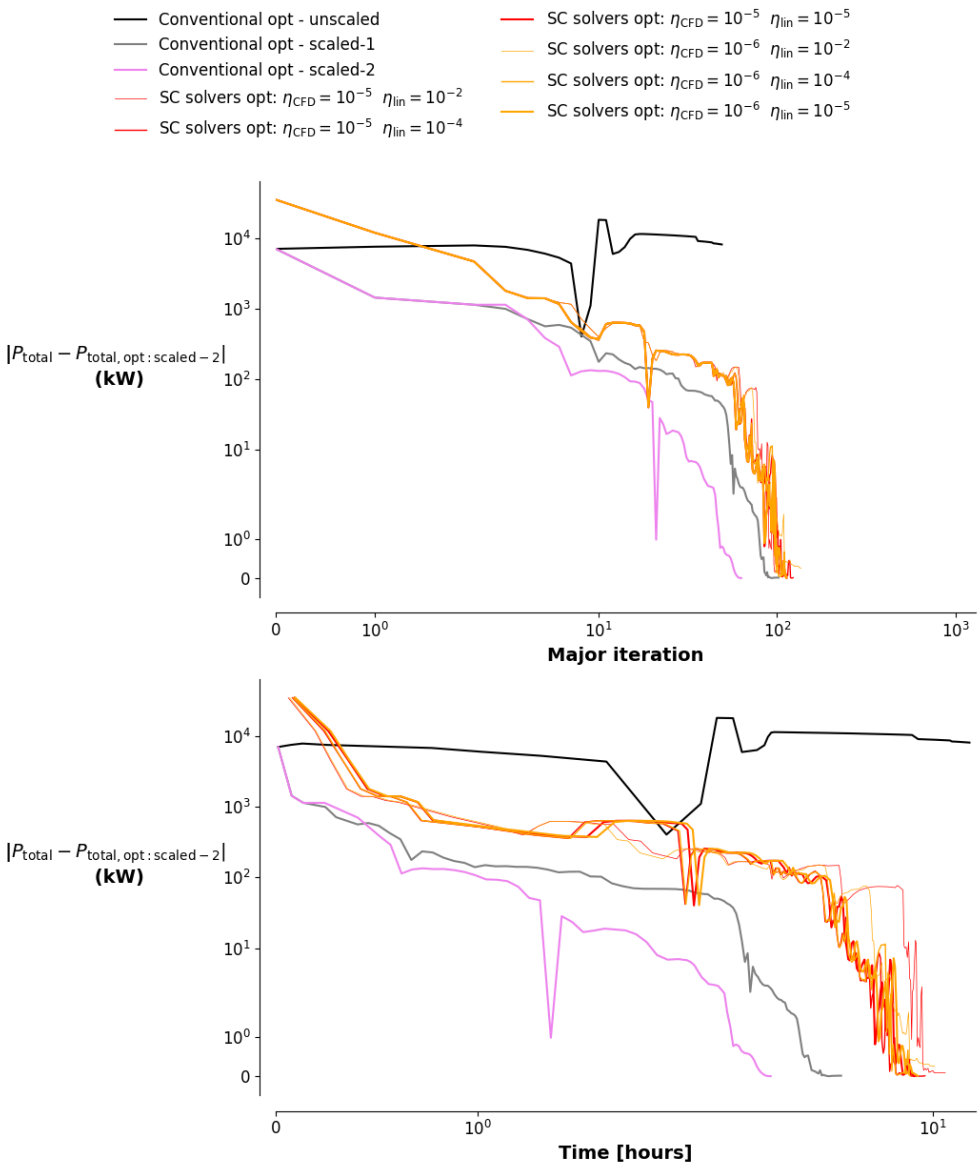


Fig. 8 P_{total} **convergence history with respect to the scaled-2 case's optimal** P_{total}**.**

Table 4 The optimization aims to minimize the total shaft power (P_{total}) subject to constraints.

Cases	Time	Scaling $r_{\dot{m}}, r_A, r_V$	Scaling $P^{\text{aero}}_{\text{s,ff}}, P^{\text{aero}}_{\text{t,fe}}, T^{\text{aero}}_{\text{t,fe}}$	Optimal $P^{\text{aero}}_{\text{s,ff}}$ (Pa), $P^{\text{aero}}_{\text{t,fe}}$ (Pa), $T^{\text{aero}}_{\text{t,fe}}$ ($^{\text{o}}$K)	P_{total}
Conventional opt : unscaled	13 hours and 02 minutes	1, 1, 1	1, 1, 1	Failed	Failed
Conventional opt : scaled-1	5 hours and 38 minutes	1, 1, 1	10^4, 10^4, 10^2	26760.0, 44357.6, 263.6	3577.495
Conventional opt : scaled-2	3 hours and 37 minutes	10^2, 10^1, 10^2	10^4, 10^4, 10^2	26760.1, 44357.9, 263.5	3577.496
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-2}$	11 hours and 27 minutes	–	–	26759.9, 44357.4, 263.6	3577.505
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-4}$	9 hours and 36 minutes	–	–	26760.0, 44357.6, 263.6	3577.495
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-5}$ $\eta_{\text{lin}} = 10^{-5}$	9 hours and 11 minutes	–	–	26760.1, 44357.8, 263.5	3577.488
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-6}$ $\eta_{\text{lin}} = 10^{-2}$	10 hours and 52 minutes	–	–	26759.9, 44357.6, 263.6	3577.517
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-6}$ $\eta_{\text{lin}} = 10^{-4}$	8 hours and 42 minutes	–	–	26760.0, 44357.6, 263.6	3577.496
SC solvers-based opt: $\eta_{\text{CFD}} = 10^{-6}$ $\eta_{\text{lin}} = 10^{-5}$	9 hours and 24 minutes	–	–	26760.1, 44357.8, 263.5	3577.488

IV. Conclusions

In this work, we present the results of applying new SC-based solvers on computational fluid dynamics-based saddle-point problems. Using BGS-based techniques is not possible with this type of model due to the non-invertible block diagonal of the Jacobian matrix. Using a fully coupled Newton's technique is one viable solution for these systems, but it has drawbacks as well because it is not robust and solving a large linear system is difficult. Similar challenges exist in linear solvers, which cannot solve a saddle-point linear system. To overcome these limitations, we applied SC solvers, which use the SC of the fully coupled Jacobian to compute the update to the non-invertible system. We present several applications to demonstrate an alternative approach to handle saddle-point optimization.

First, we presented a benchmark aerodynamic shape optimization of a wing to show the solvers' performance. At every design iteration, the nonlinear SC solver can successfully solve the balancing equation and achieve the target lift. The linear system is also effectively solved by the linear SC solver. The SC solvers-based optimizations outperform the conventional lift-constrained optimization. It is no longer necessary to formulate these balance problems as non-linearly constrained optimization problems thanks to the flexibility of SC solvers. We demonstrated how SC solvers allow the equality constraints in conventional optimization to be converted into residual forms. Moreover, we also showed how the SC-based optimizations are cost-effective when compared to the conventional lift-constrained optimization. In addition, we also demonstrated a fully coupled aeropropulsive design optimization using SC solvers. The nonlinear SC solver successfully converges the balancing equations and achieves the desired boundary conditions at each design iteration. The linear system is also successfully solved by the linear SC solver. Although the well-scaled conventional constrained optimization outperforms SC solvers-based optimizations, it provides an alternative approach to achieve a feasible design at every iteration.

When compared to conventional optimizations, the SC solver-based optimization proved to be more cost-effective in the aerodynamic shape optimization benchmark case we provided. It was not the most effective in the coupled aeropropulsive optimization. However, a well-scaled problem formulation is required in the case of conventional coupled aeropropulsive optimization with the balance residual constraints. SC solvers-based optimization always provides a feasible solution at each design iteration while conventional optimization does not. SC solvers can be a problem-dependent method in terms of cost-effectiveness. The solvers will therefore be a crucial component of simulation-based design optimization frameworks.

Acknowledgments

This work was funded by the NASA Advanced Air Transport Technology (AATT) and Transformational Tools and Technologies (TTT) projects. Computational resources supporting this work were provided by the NASA High-End Computing (HEC) Program through the NASA Advanced Supercomputing (NAS) Division at Ames Research Center.

Copyright Statement

References

[1] Keyes, D. E., McInnes, L. C., Woodward, C., Gropp, W., Myra, E., Pernice, M., Bell, J., Brown, J., Clo, A., Connors, J., Constantinescu, E., Estep, D., Evans, K., Farhat, C., Hakim, A., Hammond, G., Hansen, G., Hill, J., Isaac, T., Jiao, X., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Randles, A. P., Reynolds, D., Riviere, B., Rude, U., Scheibe, T., Shadid, J., Sheehan, B., Shephard, M., Siegel, A., Smith, B., Tang, X., Wilson, C., and Wohlmuth, B., "Multiphysics simulations: Challenges and opportunities," *International Journal of High Performance Computing Applications*, Vol. 27, No. 1, 2013, pp. 4–83. doi:10.1177/1094342012468181.

[2] Yildirim, A., Gray, J. S., and Martins, J. R. R. A., "A Nonlinear Schur Complement Solver for CFD-Based Multidisciplinary

Models," *Eleventh International Conference on Computational Fluid Dynamics*, 2022. URL https://www.iccfd.org/iccfd11/assets/pdf/papers/ICCFD11_Paper-0702.pdf, iCCFD11-0702.

[3] Benzi, M., Golub, G. H., and Liesen, J., "Numerical Solution of Saddle Point Problems," *Acta Numerica*, Vol. 14, 2005, pp. 1–137. doi:10.1017/S0962492904000212.

[4] Dassi, F., and Scacchi, S., "Parallel block preconditioners for three-dimensional virtual element discretizations of saddle-point problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 372, 2020, p. 113424. doi:https://doi.org/10.1016/j.cma.2020.113424, URL https://www.sciencedirect.com/science/article/pii/S0045782520306095.

[5] Franceschini, A., Castelletto, N., and Ferronato, M., "Block preconditioning for fault/fracture mechanics saddle-point problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 344, 2019, pp. 376–401. doi:https://doi.org/10.1016/j.cma.2018.09.039, URL https://www.sciencedirect.com/science/article/pii/S0045782518304924.

[6] Meier, A., Bänsch, E., and Frank, F., "Schur preconditioning of the Stokes equations in channel-dominated domains," *Computer Methods in Applied Mechanics and Engineering*, Vol. 398, 2022, p. 115264. doi:https://doi.org/10.1016/j.cma.2022.115264, URL https://www.sciencedirect.com/science/article/pii/S0045782522003917.

[7] Dorostkar, A., Neytcheva, M., and Serra-Capizzano, S., "Spectral analysis of coupled PDEs and of their Schur complements via Generalized Locally Toeplitz sequences in 2D," *Computer Methods in Applied Mechanics and Engineering*, Vol. 309, 2016, pp. 74–105. doi:https://doi.org/10.1016/j.cma.2016.05.042, URL https://www.sciencedirect.com/science/article/pii/S0045782516304911.

[8] Benner, P., Dolgov, S., Onwunta, A., and Stoll, M., "Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data," *Computer Methods in Applied Mechanics and Engineering*, Vol. 304, 2016, pp. 26–54. doi:https://doi.org/10.1016/j.cma.2016.02.004, URL https://www.sciencedirect.com/science/article/pii/S0045782516300263.

[9] Little, L., Saad, Y., and Smoch, L., "Block LU Preconditioners for Symmetric and Nonsymmetric Saddle Point Problems," *SIAM Journal on Scientific Computing*, Vol. 25, No. 2, 2003, pp. 729–748. doi:10.1137/S1064827502405513, URL https://doi.org/10.1137/S1064827502405513.

[10] Murphy, M. F., Golub, G. H., and Wathen, A. J., "A Note on Preconditioning for Indefinite Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 21, No. 6, 2000, pp. 1969–1972. doi:10.1137/S1064827599355153, URL https://doi.org/10.1137/S1064827599355153.

[11] He, X., Vuik, C., and Klaij, C. M., "Combining the Augmented Lagrangian Preconditioner with the Simple Schur Complement Approximation," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. A1362–A1385. doi:10.1137/17M1144775, URL https://doi.org/10.1137/17M1144775.

[12] Kraus, J., "Additive Schur Complement Approximation and Application to Multilevel Preconditioning," *SIAM Journal on Scientific Computing*, Vol. 34, No. 6, 2012, pp. A2872–A2895. doi:10.1137/110845082, URL https://doi.org/10.1137/110845082.

[13] Dillon, G., Kalantzis, V., Xi, Y., and Saad, Y., "A Hierarchical Low Rank Schur Complement Preconditioner for Indefinite Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 40, No. 4, 2018, pp. A2234–A2252. doi:10.1137/17M1143320, URL https://doi.org/10.1137/17M1143320.

[14] Dener, A., and Hicken, J. E., "Matrix–Free Algorithm for the Optimization of Multidisciplinary Systems," *Structural and Multidisciplinary Optimization*, Vol. 56, 2017, pp. 1429–1446. doi:10.1007/s00158-017-1734-0.

[15] Backhaus, T., Abdul-Kaiyoom, M. A. S., Yildirim, A., Stueck, A., and Martins, J. R. R. A., "Advancing Modularity and Framework Integration Level for Scalable High-Fidelity MDO," *AIAA AVIATION Forum*, San Diego, CA, 2023. doi:10.2514/6.2023-3315.

[16] Abdul-Kaiyoom, M. A. S., Yildirim, A., , and Martins, J. R. R. A., "Nonlinear and Linear Schur Complement Solvers for Optimization of Saddle-Point Systems (Review)," *Optimization and Engineering*, 2024.

[17] Martins, J. R. R. A., "Perspectives on Aerodynamic Design Optimization," *AIAA SciTech Forum*, AIAA, Orlando, FL, 2020. doi:10.2514/6.2020-0043.

[18] Kenway, G. K. W., and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, 2014, pp. 144–160. doi:10.2514/1.C032150.

[19] Yildirim, A., Gray, J. S., Mader, C. A., and Martins, J. R. R. A., "Coupled Aeropropulsive Design Optimization of a Podded Electric Propulsor," *AIAA Aviation Forum*, 2021. doi:10.2514/6.2021-3032.

[20] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., 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:10.1007/s00158-019-02211-z.

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

[22] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, Cambridge University Press, Cambridge, UK, 2022. doi:10.1017/9781108980647, URL `https://mdobook.github.io`.

[23] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 508–527. doi:10.2514/1.I010796.

[24] Spalart, P., and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992. doi:10.2514/6.1992-439.

[25] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. doi:10.1016/j.paerosci.2019.05.002.

[26] Yildirim, A., Kenway, G. K. W., Mader, C. A., and Martins, J. R. R. A., "A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations," *Journal of Computational Physics*, Vol. 397, 2019, p. 108741. doi:10.1016/j.jcp.2019.06.018.

[27] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, 2010. doi:10.2514/6.2010-9231.

[28] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 59, No. 4, 2021, pp. 1151–1168. doi:10.2514/1.J059491.

[29] Sederberg, T. W., and Parry, S. R., "Free-form Deformation of Solid Geometric Models," *SIGGRAPH Comput. Graph.*, Vol. 20, No. 4, 1986, pp. 151–160. doi:10.1145/15886.15903.

[30] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[31] Wu, N., Kenway, G., Mader, C. A., Jasa, J., and Martins, J. R. R. A., "pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, Vol. 5, No. 54, 2020, p. 2564. doi:10.21105/joss.02564.

[32] Welstead, J. F., "Overview of the NASA STARC-ABL (Rev. B) Advanced Concept," , March 2017. URL `https://ntrs.nasa.gov/citations/20170005612`.

[33] Hall, D. K., and Lieu, M., "Propulsor Models for Computational Analysis of Aircraft Aerodynamic Performance with Boundary Layer Ingestion," *Proceedings of the AIAA SciTech Forum*, American Institute of Aeronautics and Astronautics, 2021. doi:10.2514/6.2021-0991.

[34] Briones, A. M., Caswell, A. W., and Rankin, B. A., "Fully Coupled Turbojet Engine Computational Fluid Dynamics Simulations and Cycle Analyses Along the Equilibrium Running Line," *Journal of Engineering for Gas Turbines and Power*, Vol. 143, No. 6, 2021, p. 061019. doi:10.1115/1.4049410.

[35] Lamkin, A. H. R., Yildirim, A., Martins, J. R. R. A., and Wukie, N. A., "Advancements in Coupled Aeropropulsive Design Optimization for High-Bypass Turbofan Engines," *AIAA Aviation Forum*, San Diego, CA, 2023. doi:10.2514/6.2023-3591.

[36] Hendricks, E. S., and Gray, J. S., "pyCycle: A Tool for Efficient Optimization of Gas Turbine Engine Cycles," *Aerospace*, Vol. 6, No. 87, 2019. doi:10.3390/aerospace6080087.

[37] Jones, S. M., Haller, W. J., and Tong, M. T., "An N+3 Technology Level Reference Propulsion System," Tech. Rep. NASA/TM—2017-219501, NASA Glenn Research Center, 2017. URL https://ntrs.nasa.gov/citations/20170005426.

[38] McDonald, R. A., and Gloudemans, J. R., "Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design," *AIAA Scitech*, San Diego, CA, 2022. doi:10.2514/6.2022-0004.

[39] Abdul-Kaiyoom, M. A. S., Yildirim, A., , and Martins, J. R. R. A., "Coupled Aeropropulsive Design Optimization of an Over-Wing Nacelle Configuration," *AIAA SciTech Forum*, 2023. doi:10.2514/6.2023-0327.

[40] Abdul-Kaiyoom, M. A. S., Yildirim, A., and Martins, J. R. R. A., "RANS-Based Multipoint Aeropropulsive Design Optimization of an Over-Wing Nacelle Configuration," *AIAA Aviation Forum*, San Diego, CA, 2023. doi:10.2514/6.2023-3588.