Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Maui, HI, USA, July 11-15, 2022

ICCFD11-2022-0702

A Nonlinear Schur Complement Solver for CFD-Based Multidisciplinary Models

Anil Yildirim[a,*], Justin S. Gray[b], and Joaquim R. R. A. Martins[a]
[*] Corresponding author: anily@umich.edu
[a] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109
[b] NASA Glenn Research Center, Cleveland, OH, 44135

Abstract: Simulation-based multidisciplinary models are fundamental building blocks of multidisciplinary design optimization frameworks that involve coupled models. Solving the coupled linear and nonlinear systems that arise from these models is challenging. One common challenge arises when the Jacobian matrices represent a saddle point problem, where a block-diagonal corresponding to a discipline is non-invertible. These problems require a coupled solver algorithm such as the Newton's method instead of the popular block Gauss–Seidel-based methods because of this non-invertible block. However, the coupled Newton's method is challenging to implement and suffers from robustness issues when the initial guess is away from the solution. To address these challenges with saddle point problems, we introduce a nonlinear Schur complement solver suitable for CFD-based multidisciplinary models. The solver leverages the specialized CFD linear and nonlinear solvers. Therefore, it does not require solving a large coupled linear system like the coupled Newton's method. Furthermore, because the solver primarily uses the specialized CFD solvers, it is not subject to the coupled Newton method's robustness issues. We implement the solver in NASA's OpenMDAO framework and demonstrate its effectiveness using a coupled aeropropulsive model. The solver is applicable to a wide range of saddle point problems that arise with simulation-based design optimization formulations. By side-stepping the challenges of these saddle point problems, the solver enables flexible and robust formulation of design optimization problems, and will be an essential component of simulation based design optimization frameworks.

Keywords: Nonlinear solvers, Multidisciplinary design optimization, Saddle point problems.

1 Introduction

To reduce the environmental impact of aviation, new aircraft concepts use advanced technologies, such as boundary layer ingestion or distributed electric propulsion. However, we do not have any prior experience in designing these systems. Multidisciplinary design optimization (MDO) frameworks can help make optimal interdisciplinary trades for these technologies [1].

Multidisciplinary and multiphysics models are at the core of every MDO framework [2, Sec. 13.2]. Keyes et al. [3] provide a review of multiphysics modeling approaches and the computational challenges that arise during this process. Performing design optimization with these multiphysics problems further compounds the challenge. Solving optimization problems with large numbers of design variables and computationally expensive models in a scalable way requires gradient-based optimization [2, Sec. 1.4]. Thus, efficiently computing the derivatives of coupled models is the key to solving such optimization problems [2, Sec. 13.3]. There are many ad-hoc implementations of multiphysics models that include analytic derivatives [4**?** **?** , 5]; however, these frameworks are not extensible to additional disciplines. To tackle emerging design problems such as aeropropulsive design optimization, we need flexible and extensible MDO frameworks that

can robustly solve multidisciplinary models and automatically compute analytic derivatives for the entire coupled model.

One MDO framework developed to address this need is OpenMDAO [6]. OpenMDAO uses the MAUD architecture to automate the analytic derivative computation of coupled models with explicit and implicit model components [7]. The flexibility of OpenMDAO streamlines the creation of multidisciplinary models that can go beyond the ad-hoc implementations of multidisciplinary frameworks in terms of the number of disciplines and model complexity. However, OpenMDAO only provides the basic building blocks of the models, and it is up to the users to ensure the resulting nonlinear and linear systems can be solved successfully.

Solving the linear and nonlinear systems of equations that arise from multidisciplinary models is challenging. The most common solvers for these problems are based on Newton's method and linear and nonlinear block Gauss–Seidel (GS) methods [2]. OpenMDAO provides these core solver algorithms as well as a hierarchical solver structure where groups that contain computational components can be nested to improve solvers' robustness and performance [6].

Despite its flexibility with the hierarchical solver structure, the current solvers in OpenMDAO have difficulties when solving saddle-point problems [8]. In this context, a saddle point problem is a problem where the governing equations for a given discipline or a computational component cannot be solved on its own. This property of such models severely limits the hierarchical and GS-based solvers in OpenMDAO. The only available approach for solving them is to use a fully coupled Newton's method. However, using the fully coupled Newton's method is not always feasible or the most robust option available. Furthermore, despite providing quadratic convergence near the final solution, the Newton's method is not guaranteed to be the most computationally efficient approach due to the requirement of solving a large linear system at each solver iteration.

Saddle point problems like these appear across many different design optimization applications. The most common example with CFD models arises when performing aerodynamic shape optimization of aircraft configurations to minimize drag or some other performance metric that depends on drag, subject to a target lift constraint [9]. These optimization problems are usually formulated with equality constraints; however, the target lift value can also be achieved by varying the angle of attack at each design iteration. In this formulation, the angle of attack variable is included in the model states and the target lift value can be enforced with a *balance equation* with its own residual. This formulation is not commonly used 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 therefore, non-invertible. Other common examples that arise with CFD-based multidisciplinary models include the trim balance for aerostructural optimizations [10], and as we introduce in Sec. 3, the continuation balance for CFD models with powered boundary conditions to model propulsors [11]. These example applications show that saddle point problems appear across a wide range of design optimization applications, even with single disciplinary models. Therefore, addressing this challenge of solving these saddle point problems during design optimization is critical for flexible formulation of simulation based optimization problems.

To demonstrate the shortcomings of the current OpenMDAO solvers with these problems, we consider a generic two-discipline multidisciplinary model. The coupled nonlinear system of this example problem is written as:

$$\mathcal{R}(u) = \begin{bmatrix} \mathcal{R}_1(u_1, u_2) \\ \mathcal{R}_2(u_1, u_2) \end{bmatrix} = 0, \tag{1}$$

where $\mathcal{R}_1$ and $\mathcal{R}_2$ represent the governing equations of the two disciplines and $u_1 \in \mathbb{R}^m$ and $u_2 \in \mathbb{R}^n$ represent the states of each discipline. The Jacobian matrix that contains the partial derivatives of the model's residuals with respect to its states can be written as

$$\frac{\partial \mathcal{R}}{\partial u} = \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & \frac{\partial \mathcal{R}_2}{\partial u_2} \end{bmatrix}. \tag{2}$$

The two most common methods of solving this system of nonlinear equations are Newton's method and block Gauss–Seidel (BGS)-based methods. With the Newton's method, the Jacobian matrix introduced in

Eq. 2 is used in the formulation of a linear system, written as:

$$\frac{\partial \mathcal{R}}{\partial u}\Delta u^i = -\mathcal{R}(u^i), \tag{3}$$

where the superscript i represents the iteration number. Using this formula, the vector Δu^i iteratively updates the state until convergence as

$$u^{i+1} = u^i + \Delta u^i. \tag{4}$$

Even though Newton's method provides good convergence rates near the final solution, it suffers from two main shortcomings. First, the update formulation requires the solution of a linear system that contains the entire state vector of the multidisciplinary model. While solving such systems can be tractable with relatively small models, their solution can get significantly more challenging as the multidisciplinary state vector size increases. Secondly, Newton's method cannot robustly converge to the solution when the initial guess is far from the final solution, and robustly solving the resulting nonlinear systems requires globalization strategies [12].

Because of these shortcomings, the nonlinear BGS-based methods are often preferred over Newton's method for large multidisciplinary models that involve CFD solvers. This is because the nonlinear BGS methods can utilize each discipline's specialized linear and nonlinear solvers. To solve the coupled nonlinear equations, the nonlinear BGS method first solves the first discipline's governing equations and uses the new state of the first discipline to update and solve the second discipline's governing equations. The method then iterates between the two disciplinary models until it converges sufficiently.

An example solution process with the nonlinear BGS approach that uses the Newton's method to solve each disciplinary model can be written as follows. First, the guess for u_2 at the i^{th} iteration is used to solve the first discipline's governing equations:

$$\begin{aligned} \frac{\partial \mathcal{R}_1}{\partial u_1}\Delta u_1^j &= -\mathcal{R}_1(u_1^j, u_2^i), \\ u^{j+1} &= u^j + \Delta u^j. \end{aligned} \tag{5}$$

where the update formula is iterated until the j^{th} sub-iterate solves the governing equations of the first discipline and is taken as the $(i+1)^{\text{th}}$ iterate of u_1 with the nonlinear BGS solver:

$$\mathcal{R}_1(u_1^{i+1}, u_2^i) = 0. \tag{6}$$

Then using the new state u_1^{i+1}, the second discipline's governing equations are solved with the same approach:

$$\mathcal{R}_2(u_1^{i+1}, u_2^{i+1}) = 0. \tag{7}$$

This process is iterated until both disciplines' governing equations are satisfied. At the cost of lower convergence rates, the nonlinear BGS approach can be used to solve multidisciplinary models. Despite the convergence rate, a significant advantage of the nonlinear BGS method is that it only requires the linear and nonlinear solutions with individual model components, enabling the use of specialized solvers developed for each component. For example, if one discipline includes a CFD model, the CFD code's specialized linear and nonlinear solvers can be used with the nonlinear BGS method to solve coupled multidisciplinary systems.

Even though the nonlinear BGS method is a popular choice for solving multidisciplinary models, it cannot be used with multidisciplinary models that contain a saddle point problem. This is because the nonlinear BGS solver requires the diagonal sub-blocks of the Jacobian ($\partial\mathcal{R}_1/\partial u_1$ and $\partial\mathcal{R}_2/\partial u_2$) to be invertible. However, many practical multidisciplinary models like the one we detail in Section 3 have one diagonal sub-block in the Jacobian matrix that cannot be inverted. Specifically, with the practical example model we use in this work, the second diagonal sub-block of the Jacobian is a zero matrix. With this property, the resulting Jacobian matrix becomes:

$$\frac{\partial \mathcal{R}}{\partial u} = \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & 0 \end{bmatrix}. \tag{8}$$

Because of this property of the nonlinear system, the nonlinear BGS-based methods cannot solve the resulting

nonlinear systems.

To address this shortcoming with the BGS-based solvers in OpenMDAO while also side-stepping the challenges associated with the fully-coupled Newton's method, we develop a nonlinear Schur complement (NSC) solver. We implement the NSC solver in OpenMDAO and demonstrate its performance with a CFD-based aeropropulsive model. Compared to the available solver algorithms in OpenMDAO, the NSC solver has several advantages. First, while the BGS cannot solve saddle point problems, NSC solves such problems by considering the fully coupled Jacobian during the update process. Secondly, despite considering the fully coupled Jacobian, the NSC solver does not require the solution of the large coupled linear system. The NSC solver accomplishes this by utilizing the specialized linear and nonlinear solvers of the models' sub-components. As a result, the NSC solver does not suffer from the same robustness and computational cost limitations as the fully coupled Newton's method.

In the remainder of this paper, we first introduce the mathematical formulation of the NSC solver in Section 2, and demonstrate its effectiveness with an example CFD-based aeropropulsive model in Section 3. Finally, we provide the conclusions of this work in Section 4.

2 Nonlinear Schur Complement Solver

The goal of the NSC solver is to solve the coupled nonlinear system efficiently without needing to compute a fully coupled Newton update. The solver also leverages existing specialized disciplinary linear and nonlinear solvers while avoiding the inversion of diagonal sub-blocks of the Jacobian. In the following section, we derive the solver algorithm and demonstrate the effectiveness of the solver with simple test problems.

2.1 Solver Derivation

To derive the NSC solver, we again use the example two-discipline system introduction in Section 1 and consider the following coupled Newton update formula for this system:

$$\frac{\partial \mathcal{R}}{\partial u}\Delta u^i = \begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & \frac{\partial \mathcal{R}_2}{\partial u_2} \end{bmatrix} \begin{bmatrix} \Delta u_1^i \\ \Delta u_2^i \end{bmatrix} = \begin{bmatrix} -\mathcal{R}_1(u_1^i, u_2^i) \\ -\mathcal{R}_2(u_1^i, u_2^i) \end{bmatrix} = -\mathcal{R}(u_1^i, u_2^i). \tag{9}$$

This formulation requires the solution of a linear system that includes both disciplines' states. However, we can compute the Schur complement of the coupled Jacobian matrix to compute the updates to the states separately for each discipline. Rearranging the first row of Eq. 9, we get

$$\Delta u_1^i = \left(\frac{\partial \mathcal{R}_1}{\partial u_1}\right)^{-1} \left(-\mathcal{R}_1(u_1^i, u_2^i) - \frac{\partial \mathcal{R}_1}{\partial u_2}\Delta u_2^i\right). \tag{10}$$

Multiplying both sides by $\partial \mathcal{R}_2/\partial u_1$ and substituting in the second row of Eq. 9 yields

$$\left(\frac{\partial \mathcal{R}_2}{\partial u_2} - \frac{\partial \mathcal{R}_2}{\partial u_1}\left(\frac{\partial \mathcal{R}_1}{\partial u_1}\right)^{-1}\frac{\partial \mathcal{R}_1}{\partial u_2}\right)\Delta u_2^i = -\mathcal{R}_2(u_1^i, u_2^i) + \frac{\partial \mathcal{R}_2}{\partial u_1}\left(\frac{\partial \mathcal{R}_1}{\partial u_1}\right)^{-1}\mathcal{R}_1(u_1^i, u_2^i). \tag{11}$$

Using this formulation, we can first solve for Δu_2^i and then substitute this result into Eq. 10 to obtain Δu_1^i.

This formulation uses the Schur complement of the coupled Jacobian matrix to compute the coupled Newton update for the second discipline Δu_2^i, without needing to form and solve the fully coupled Jacobian. This formulation also does not directly invert the diagonal sub-block of the Jacobian; the inverse of $\partial \mathcal{R}_2/\partial u_2$ does not show up in the equation directly, and the final linear system has a solution if the coupled Jacobian itself is invertible. Despite avoiding the solution of one large linear system with all of the states, computing Δu_2^i with this formulation requires $n+1$ linear solutions with the matrix $\partial \mathcal{R}_1/\partial u_1$. This is because the matrix $\partial \mathcal{R}_1/\partial u_2$ has a size of m by n, and therefore the term $(\partial \mathcal{R}_1/\partial u_1)^{-1}\partial \mathcal{R}_1/\partial u_2$ that appears on the left hand side of Eq. 11 requires n linear solutions; one for each column of $\partial \mathcal{R}_1/\partial u_2$. The final additional linear solution arises from the term $(\partial \mathcal{R}_1/\partial u_1)^{-1}\mathcal{R}_1(u_1^i, u_2^i)$ on the right-hand side of Eq. 11.

As detailed in the next section, a typical multidisciplinary model structure that can benefit from this

solver contains a CFD solver in the first discipline. Therefore, the first discipline has many more states than the second one. This factor also affects the cost of linear solutions with each disciplines' Jacobian; the linear solutions with the first discipline's Jacobian is significantly more expensive than the second. As a result, keeping the number of linear solutions with the first discipline as small as possible is beneficial for computational performance. Furthermore, we prefer to leverage the CFD solver's nonlinear solver instead of computing an update to its states using Eq. 10. Combining these two ideas, we can eliminate the additional linear solution required for Eq. 11 by first solving $\mathcal{R}_1 = 0$ with a constant u_2, which is identical to the nonlinear BGS update for the first discipline.

Re-formulating Eq. 9 with this approach yields

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & \frac{\partial \mathcal{R}_2}{\partial u_2} \end{bmatrix} \begin{bmatrix} \Delta u_1^i \\ \Delta u_2^i \end{bmatrix} = \begin{bmatrix} -\mathcal{R}_1(u_1', u_2^i) \\ -\mathcal{R}_2(u_1', u_2^i) \end{bmatrix} = \begin{bmatrix} 0 \\ -\mathcal{R}_2(u_1', u_2^i) \end{bmatrix} \tag{12}$$

where the intermediate state u_1' satisfies $\mathcal{R}_1(u_1', u_2^i) = 0$. Then, Δu_2^i can be obtained by solving

$$\left(\frac{\partial \mathcal{R}_2}{\partial u_2} - \frac{\partial \mathcal{R}_2}{\partial u_1} \left(\frac{\partial \mathcal{R}_1}{\partial u_1} \right)^{-1} \frac{\partial \mathcal{R}_1}{\partial u_2} \right) \Delta u_2^i = -\mathcal{R}_2(u_1', u_2^i), \tag{13}$$

which only requires n additional linear solutions with the matrix $\partial \mathcal{R}_1 / \partial u_1$ instead of $n + 1$.

The last step of this approach 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. This correction can be formulated using Eq. 10 as:

$$\frac{\partial \mathcal{R}_1}{\partial u_1} \Delta u_1^i = -\mathcal{R}_1(u_1', u_2^i) - \frac{\partial \mathcal{R}_1}{\partial u_2} \Delta u_2^i = -\frac{\partial \mathcal{R}_1}{\partial u_2} \Delta u_2^i. \tag{14}$$

However, the right-hand side of this formulation is simply a first-order approximation of $\mathcal{R}_1$ at $(u_1', u_2^i + \Delta u_2^i) = (u_1', u_2^{i+1})$. Therefore, instead of relying on this first order approximation, we can update the u_2 vector and evaluate the true nonlinear residual $\mathcal{R}_1(u_1', u_2^{i+1})$. Finally, we can solve for u_1^{i+1} using the specialized nonlinear solver of the first discipline. With this approach, instead of taking a single nonlinear solver iteration with the first discipline, we can converge $\mathcal{R}_1$, which is a preliminary step required for obtaining Eq. 12 in the first place. Using this connection, we can replace u_1' in each iteration by u_1^i; the intermediate first discipline state that solves $\mathcal{R}_1 = 0$ given u_2^i can simply be carried over as the u_1 vector that results from the previous iteration.

The resulting NSC solver from this derivation is summarized in Algorithm 1. In this formulation, η_{rel} represents the relative nonlinear convergence tolerance. The solver only relies on the specialized nonlinear solver of the first discipline to update the first discipline's state vector u_1. As a result, it does not introduce robustness issues due to using the Newton's method without globalization. Furthermore, the solver does not require the inversion of the diagonal sub-block of the Jacobian ($\partial \mathcal{R}_2 / \partial u_2$) and obtains an update for the second discipline's states that is equivalent to using the coupled Newton's method.

Algorithm 1 The NSC solver.

Given u_2^0, solve $\mathcal{R}_1(u_1^0, u_2^0) = 0$ to obtain u_1^0.
$i = 0$
while $||\mathcal{R}(u_1^i, u_2^i)||_2 > \eta_{\text{rel}} ||\mathcal{R}(u_1^0, u_2^0)||_2$ **do**
  Solve Eq. 13 with u_1^i as u_1' to obtain Δu_2^i.
  $u_2^{i+1} = u_2^i + \Delta u_2^i$
  Given u_2^{i+1}, solve $\mathcal{R}_1(u_1^{i+1}, u_2^{i+1}) = 0$ to obtain u_1^{i+1}.
  $i = i + 1$
end while

2.2 Simple Test Problems

We introduce two simple nonlinear problems to demonstrate the NSC solver's basic principles. The first problem's governing equations can be written as

$$\begin{aligned} \mathcal{R}_1(u_1, u_2) &= 5\frac{u_1}{2} + 20\left(\frac{u_2}{2}\right)^3, \\ \mathcal{R}_2(u_1, u_2) &= \left(\frac{u_1}{2}\right)^3 + \frac{u_2}{2}, \end{aligned} \tag{15}$$

where both disciplines contain a single scalar state, and both residuals depend on both of the states. In contrast, we also introduce another test problem,

$$\begin{aligned} \mathcal{R}_1(u_1, u_2) &= 5\frac{u_1}{2} + 20\left(\frac{u_2}{2}\right)^3, \\ \mathcal{R}_2(u_1, u_2) &= \left(\frac{u_1}{2}\right)^3. \end{aligned} \tag{16}$$

Both test problems contain one solution at $(u_1, u_2) = (0, 0)$. The second test problem differs from the first one introduced in Eq. 15 in a single term in the second residual. This modification removes the dependence of the second residual ($\mathcal{R}_2$) on the second discipline's state (u_2). As we will show next, this modification introduces significant challenges in solving the second problem.

To demonstrate the significance of the change, we compute the Jacobian matrix of both problems. The first problem's Jacobian is:

$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & \frac{\partial \mathcal{R}_2}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2.5 & 30(\frac{u_2}{2})^2 \\ 1.5(\frac{u_1}{2})^2 & 0.5 \end{bmatrix}, \tag{17}$$

whereas the second problem's Jacobian is:

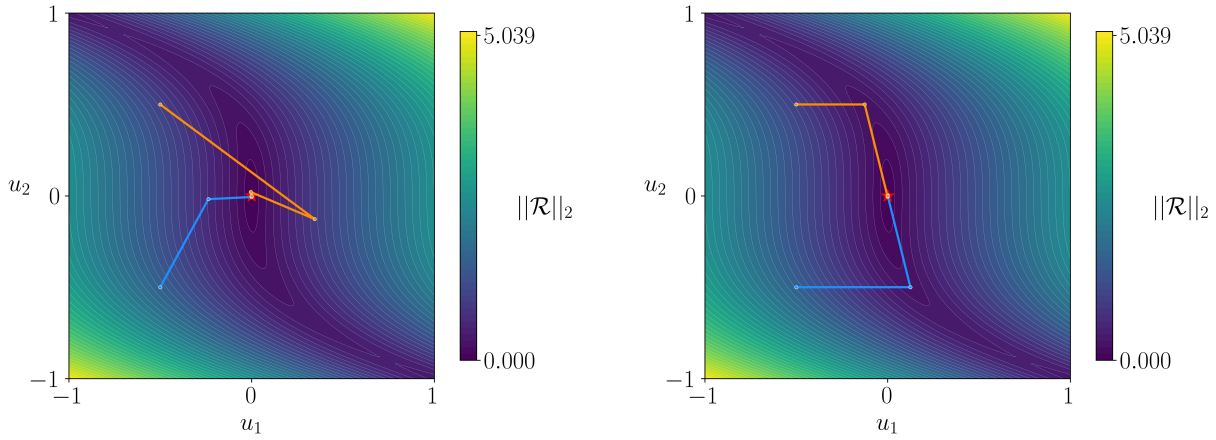
$$\begin{bmatrix} \frac{\partial \mathcal{R}_1}{\partial u_1} & \frac{\partial \mathcal{R}_1}{\partial u_2} \\ \frac{\partial \mathcal{R}_2}{\partial u_1} & \frac{\partial \mathcal{R}_2}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2.5 & 30(\frac{u_2}{2})^2 \\ 1.5(\frac{u_1}{2})^2 & 0 \end{bmatrix}. \tag{18}$$

Due to the removal of the u_2 dependence in $\mathcal{R}_2$ in the second problem, the last entry of the second problem's Jacobian becomes zero. Even though its full Jacobian is invertible in most of the solution domain, the sub-Jacobian of the second discipline ($\partial \mathcal{R}_2 / \partial u_2$) is not invertible. As a result, even though both problems can be solved with the fully-coupled Newton's method, the second problem cannot be solved with the BGS-based solvers.

The NSC solver is developed for saddle point problems with this structure where one of the disciplines cannot be solved independently of the others. Because it does not need to invert the diagonal sub-Jacobian block, on top of being able to solve the first test problem, the NSC solver can also solve the second test problem.

We demonstrate the convergence of both solvers with the first test problem from two initial guesses in Figure 1. In this figure, we plot the convergence path of each solver over the contours of the residual vector L2 norm. Similarly, we plot the convergence of both solvers with the second problem in Figure 2. These results demonstrate that the NSC solver can solve these nonlinear test problems without solving the full Newton update or inverting the second diagonal sub-block of the Jacobian.

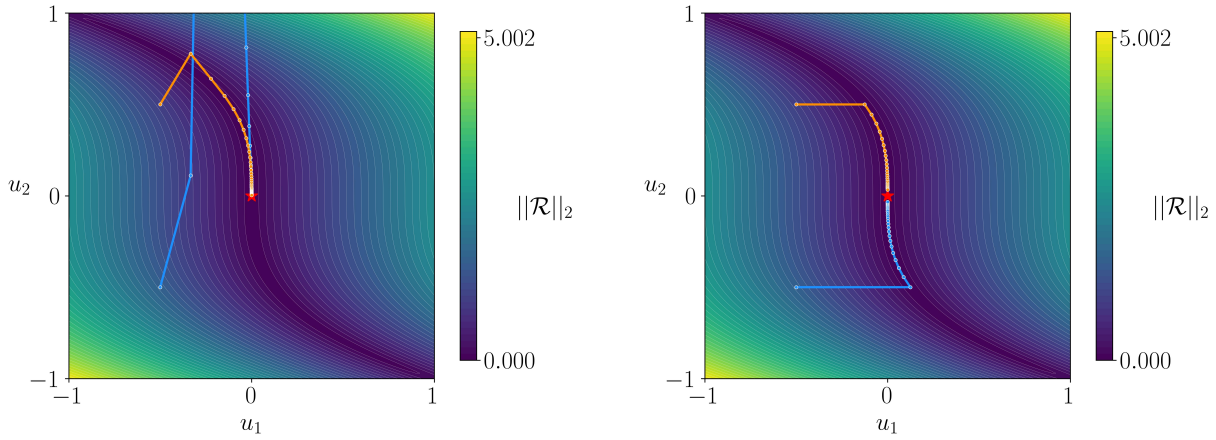
When the first discipline's residuals are converged exactly in its updates, the NSC solver becomes equivalent to the reduced-space Newton's method [6]. Even though these two formulations are equivalent in this special case, their convergence path is not guaranteed to be the same when the underlying nonlinear and linear systems are solved inexactly. In this context, the NSC solver provides more flexibility in the solver approach by formulating the nonlinear system in the full space, rather than the reduced space. Furthermore, the NSC solver's performance can be optimized by adjusting the convergence tolerances for the underlying linear and nonlinear solutions as we discuss in the next section with a practical problem.



(a) Newton solver convergence path.

(b) NSC solver convergence path.

Figure 1: L2 norm of the residual vector and the nonlinear convergence history of the first test problem (Eq. 15) with the Newton and NSC solvers. The solution at $(0, 0)$ is highlighted with a red star.



(a) Newton solver convergence path.

(b) NSC solver convergence path.

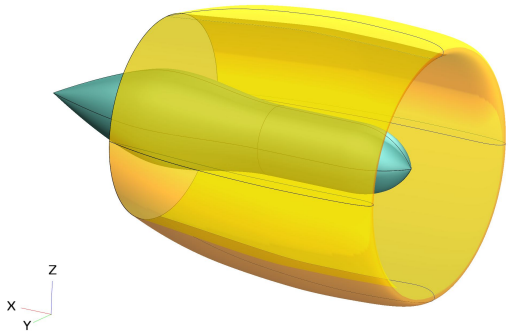
Figure 2: L2 norm of the residual vector and the nonlinear convergence history of the second test problem (Eq. 16) with the Newton and NSC solvers. The solution at $(0, 0)$ is highlighted with a red star.

3 Solver Performance with a CFD-Based Aeropropulsive Model

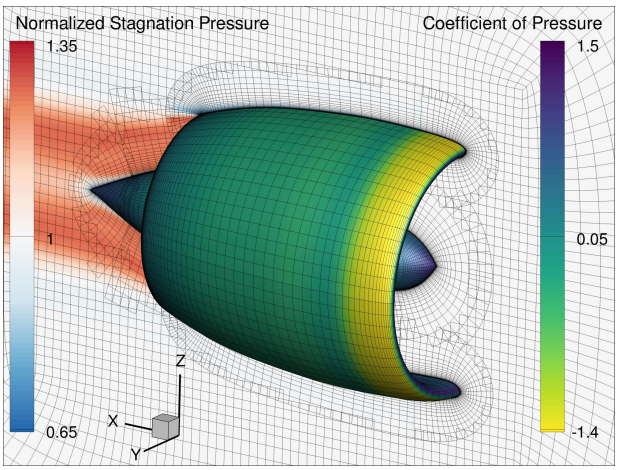
Our primary motivation for developing the NSC solver is to solve nonlinear systems arising from CFD-based aeropropulsive models. In particular, we want to solve the nonlinear problems that result from using boundary conditions (BCs) to model propulsors in CFD models. We first introduce the coupled aeropropulsive model we use as a benchmark case, explain how the NSC solver is integrated into the rest of the solver hierarchy, and study the solver's performance.

3.1 Coupled Aeropropulsive Model

The aeropropulsive model we use in this study is the BC version of the benchmark aeropropulsive model developed in our previous work [11]. The benchmark model is a podded electric fan design shown in Fig. 3, based on the aft-propulsor of NASA's STARC-ABL concept [13]. While this is a simple design, it represents the fundamental challenges of coupling a CFD solver to a propulsion model for a fully coupled aeropropulsive



(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.

Figure 3: Geometry and the CFD model of the podded fan.

model. We only study the BC version of the benchmark model in this work because the nonlinear system resulting from this version results in a zero sub-block in the Jacobian matrix, which prevents using BGS-based solvers. In contrast, the NSC solver does not suffer the same limitations.

The coupled aeropropulsive model uses powered BCs to introduce the effects of the propulsion system in the CFD model. Even though source-term formulations are more accurate for fan simulations [14], the BC approach is also valuable. First, not all CFD solvers support the source-term formulation; therefore, a BC formulation is more widely applicable. Secondly, while the source-term formulations are more accurate for components such as fans, they become computationally restrictive when simulating an engine core, where low-order cycle models can provide sufficient accuracy [15]. As a result, the know-how developed with aeropropulsive design optimization using a BC formulation will be useful for future complete turbofan design optimization problems where the core flow inlet and outlet interfaces are modeled using this BC formulation.

The extended design structure matrix (XDSM) diagram [16] of the multidisciplinary analysis (MDA) formulation is shown in Fig. 4. The coupled aeropropulsive model is implemented using MPhys, a flexible multiphysics analysis and optimization library based on NASA's OpenMDAO framework [6]. With the help of MPhys, the coupled aeropropulsive state vector and functionals are exposed to OpenMDAO, which handles the data transfer between each model and solves the coupled derivative problems for gradient-based optimization. Even though we do not perform design optimizations in this work, we use the available analytic derivatives in the computation of the partial derivatives in the Jacobian matrix for the NSC solver.

The coupled aeropropulsive models consist of three main components: the aerodynamic model, the propulsion model, and the balance formulation used to couple the two models together. In the following sections, we detail each component in these coupled aeropropulsive models and explain the coupled model formulation.

3.1.1 Aerodynamic Model

The aerodynamic analyses are performed with the open-source CFD solver, ADflow[1] [17], which solves the Reynolds-averaged Navier–Stokes (RANS) equations using a finite volume scheme on structured multi-block and overset meshes. In this study, we use the Spalart–Allmaras (SA) turbulence model [18] with the RANS simulations to compute the flow field around the propulsor, as shown in Fig. 3b. ADflow is well

[1] `https://github.com/mdolab/adflow`, accessed June 2022

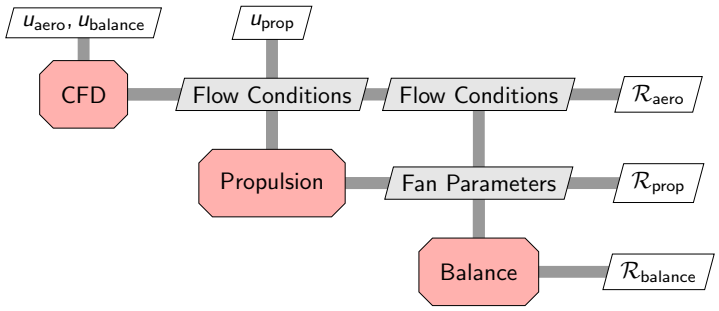


Figure 4: XDSM diagram of the podded fan aeropropulsive model [11].

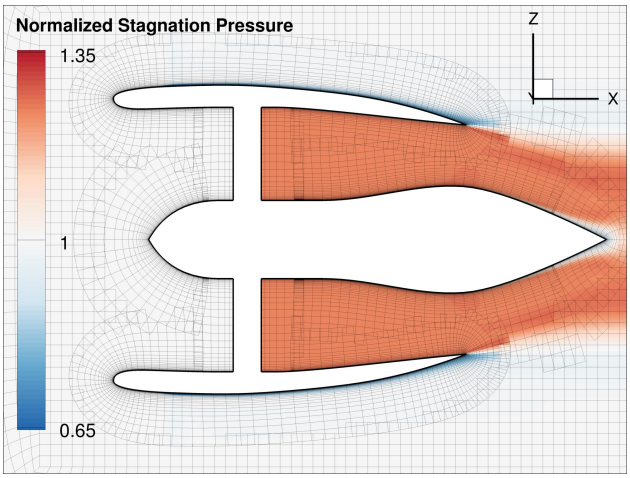


Figure 5: The CFD model uses subsonic outflow BCs for the fan face and subsonic inflow BCs for the fan exit. The mass, momentum, and energy conservation across the fan are satisfied by adjusting the BC values on the fan face and exit.

suited for gradient-based optimization because of its Python interface [17] and efficient adjoint solver implementation [19]. Due to the challenging nature of converging CFD simulations with overset meshes and fan models, we use the approximate Newton–Krylov (ANK) solver implemented in ADflow [20]. The CFD mesh contains about 600 thousand volume cells. Figure 5 shows the normalized stagnation pressure contours on the symmetry plane.

We prescribe subsonic outflow BCs at the fan face by setting the static pressure at the fan face ($p_{\text{s,ff}}$). At the fan exit, subsonic inflow conditions are prescribed by setting the total pressure ($p_{\text{t,fe}}$) and total temperature ($T_{\text{t,fe}}$) at the fan exit. These three variables are inputs to the CFD model.

The 3-D CFD solutions need to be averaged at the fan face and exit to obtain inputs for the propulsion model introduced in the next section. These parameters are included in the outputs of the CFD model in Fig. 4. We use an area-based averaging for static and total pressure terms, and for the total temperature integration, we use a mass-based averaging. The mass flow rate, area, pressure, and momentum forces on the fan face and exit are computed directly using the integration from the CFD solver without any averaging. The details of the flow transfer method to couple the CFD model to the propulsion model is explained in our previous work [11], which is based on the force-based averaging scheme developed by Gray et al. [21].

3.1.2 Propulsion Model

To estimate the power required by the fan, we use thermodynamic cycle analyses built with the pyCycle library [22]. The propulsion model consists of a simple compressor with an assumed polytropic efficiency

of 0.97, based on the NASA N+3 technology level reference propulsion system [23]. The compressor model computes the total power required to operate the fan using the mass and area averaged inputs from the CFD model. We then use the total power required by the propulsor ($\mathcal{P}_{\text{total}}$) to prescribe the increase in the total temperature across the fan in the CFD model, as explained in the next section.

3.1.3 Aeropropulsive Coupling

To obtain a coupled aeropropulsive solution, we must satisfy the mass, momentum, and energy conservation across the fan. Conservation of these quantities are achieved by adjusting the static pressure at the fan face ($p_{\text{s,ff}}$), and the total pressure ($p_{\text{t,fe}}$) and total temperature ($T_{\text{t,fe}}$) at the fan exit.

The balance formulation requires the amount of force applied by the fan (F_{fan}) to be specified as an input. This force applied by the fan is different from the net thrust; the fan force is how much the fan pushes on the flow, whereas the net thrust of the propulsor also includes the integration of pressure and viscous forces on the propulsor walls. In this context, the force applied by the fan is equal to the difference in momentum of the flow between the fan face and exit BCs. The momentum and energy added to the flow by the fan appears in the balance equations, which are formulated as

$$\begin{aligned}
\mathcal{R}_{\text{mass}}\left(p_{\text{s,ff}}, p_{\text{t,fe}}, T_{\text{t,fe}}\right) &= \dot{m}_{\text{ff}} - \dot{m}_{\text{fe}} = 0, \\
\mathcal{R}_{\text{momentum}}\left(p_{\text{s,ff}}, p_{\text{t,fe}}, T_{\text{t,fe}}\right) &= F_{\text{momentum,ff}} + F_{\text{pressure,ff}} - F_{\text{momentum,fe}} - F_{\text{pressure,fe}} + F_{\text{fan}} = 0, \\
\mathcal{R}_{\text{energy}}\left(p_{\text{s,ff}}, p_{\text{t,fe}}, T_{\text{t,fe}}\right) &= \left(T_{\text{t,ff}}\dot{m}_{\text{ff}} - T_{\text{t,fe}}\dot{m}_{\text{fe}}\right) c_p + \mathcal{P}_{\text{total}} = 0,
\end{aligned} \tag{19}$$

where the subscripts "ff" and "fe" represent the values evaluated at the fan face and fan exit, respectively. In this formulation, all of the terms except for the force applied by the fan (F_{fan}) and the total power required for the fan ($\mathcal{P}_{\text{total}}$) are obtained by integrating the flow field at the fan face and fan exit BCs in the CFD domain. Finally, the isobaric-specific heat (c_p) of air is set to 1.0045 kJ/kgK°.

When this three-by-three system of equations is satisfied, the mass flow at the fan face is equal to the mass flow at the fan exit, the momentum going into the fan face, plus the fan force is equal to the momentum coming out of the fan exit. Finally, the rate of energy flow into the fan face plus the total power of the fan is equal to the rate of energy that comes out of the fan exit. This total power term ($\mathcal{P}_{\text{total}}$) is computed by the propulsion model introduced in Section 3.1.2.

In previous work [11], we did not converge the residuals listed in Eq. 19 at each analysis. Instead, we formulated the optimization problems such that the three BC parameters are design variables, and the three residual equations are equality constraints in the optimization problems. This is because we could not solve these residual equations given the solver capabilities before developing the NSC solver. Adding these balance equations as equality constraints in the optimization formulations introduce the requirement of computing the total derivatives of these residuals with respect to design variables in every design iteration. This is not computationally efficient because the resulting linear systems to compute the derivatives to be used in optimization need to be solved accurately, meanwhile we do not need to solve the resulting linear systems with the NSC solver accurately as we demonstrate in Sec. 3.3. Furthermore, using the NSC solver is more robust compared to including the balance residuals as equality constraints because the NSC solver ensures that the flow through the fan is balanced at each design iteration. On the other hand, the flow through the fan starts at an imbalanced state where the equality constraints are not satisfied in the early iterations with the optimization problem formulation based on equality constraints. This imbalance can introduce undesirable features in the flow field such as separation that negatively affect CFD solver's accuracy and performance. Because of these factors, it is more beneficial to use the NSC solver instead of the equality constraint based approach in previous work [11].

Without the NSC solver, one method to solve these additional balance equations with the CFD model is to use a fully-coupled Newton's method where the three balance residuals are coupled to the large nonlinear system of the CFD model. However, this approach requires the solution of a large linear system that includes the updates to the CFD model as well as the BC variables. Due to the large size and the stiffness of the resulting linear system, this approach requires a highly customized linear solver that can couple additional states to the specialized linear solver in the CFD code.

Another commonly used approach to solve CFD-based multidisciplinary models is the nonlinear BGS method. However, this approach cannot be used with these balance equations because the states ($p_{\text{s,ff}}$, $p_{\text{t,fe}}$,

and $T_{\mathrm{t,fe}}$) do not directly appear in the balance residuals explicitly. As a result, the sub-block of the global Jacobian is a three-by-three zero matrix. Even though the coupled Jacobian is an invertible system, this sub-block is non-invertible, preventing the use of BGS-based solvers. Due to these limitations, we did not solve the balance equations at each design iteration in previous work. The NSC solver we introduced in this work is developed to address the challenges that arise directly. The following section explains how we use the NSC solver to converge the coupled aeropropulsive model.

3.2 Hierarchical Solver Structure

The aeropropulsive model introduced in the previous section results in three groups of states and governing equations that must be solved to obtain a balanced simulation. The vector of states and residuals of this system can be written as:

$$u = \begin{bmatrix} u_{\mathrm{aero}} \\ u_{\mathrm{prop}} \\ u_{\mathrm{balance}} \end{bmatrix}, \; \mathcal{R} = \begin{bmatrix} \mathcal{R}_{\mathrm{aero}} \\ \mathcal{R}_{\mathrm{prop}} \\ \mathcal{R}_{\mathrm{balance}} \end{bmatrix}. \tag{20}$$

In this notation, u_{aero} represents the states of the CFD solver to model the aerodynamics, u_{prop} represents the states of the propulsion model, and finally, u_{balance} represents the states for the balance formulation listed in Eq. 19.

We group the states for the aerodynamic and propulsion disciplines to obtain the combined state for the aeropropulsive discipline to re-formulate the nonlinear system to more closely resemble the two-discipline example introduced earlier. This results in

$$u = \begin{bmatrix} u_{\mathrm{aeroprop}} \\ u_{\mathrm{balance}} \end{bmatrix}, \; \mathcal{R} = \begin{bmatrix} \mathcal{R}_{\mathrm{aeroprop}} \\ \mathcal{R}_{\mathrm{balance}} \end{bmatrix}, \tag{21}$$

where $u_{\mathrm{aeroprop}} = [u_{\mathrm{aero}} \; u_{\mathrm{prop}}]^{\mathrm{T}}$, $\mathcal{R}_{\mathrm{aeroprop}} = [\mathcal{R}_{\mathrm{aero}} \; \mathcal{R}_{\mathrm{prop}}]^{\mathrm{T}}$ and the entries of the Jacobian matrix can be formulated as:

$$\begin{bmatrix} \frac{\partial \mathcal{R}_{\mathrm{aero}}}{\partial u_{\mathrm{aero}}} & \frac{\partial \mathcal{R}_{\mathrm{aero}}}{\partial u_{\mathrm{prop}}} & \frac{\partial \mathcal{R}_{\mathrm{aero}}}{\partial u_{\mathrm{balance}}} \\ \frac{\partial \mathcal{R}_{\mathrm{prop}}}{\partial u_{\mathrm{aero}}} & \frac{\partial \mathcal{R}_{\mathrm{prop}}}{\partial u_{\mathrm{prop}}} & \frac{\partial \mathcal{R}_{\mathrm{prop}}}{\partial u_{\mathrm{balance}}} \\ \frac{\partial \mathcal{R}_{\mathrm{balance}}}{\partial u_{\mathrm{aero}}} & \frac{\partial \mathcal{R}_{\mathrm{balance}}}{\partial u_{\mathrm{prop}}} & \frac{\partial \mathcal{R}_{\mathrm{balance}}}{\partial u_{\mathrm{balance}}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{R}_{\mathrm{aeroprop}}}{\partial u_{\mathrm{aeroprop}}} & \frac{\partial \mathcal{R}_{\mathrm{aeroprop}}}{\partial u_{\mathrm{balance}}} \\ \frac{\partial \mathcal{R}_{\mathrm{balance}}}{\partial u_{\mathrm{aeroprop}}} & \frac{\partial \mathcal{R}_{\mathrm{balance}}}{\partial u_{\mathrm{balance}}} \end{bmatrix}. \tag{22}$$

Using this rearrangement, the "aeroprop" group represents the first discipline, while the "balance" group represents the second discipline in Algorithm 1.

The NSC solver involves solving the nonlinear system for the first discipline using its specialized nonlinear solver. To do this with the aeropropulsive model, we use the solver hierarchy of the OpenMDAO model shown in Fig. 6. In this XDSM diagram, the NSC solver provides the current values of u_{balance} using the update formula in Eq. 13. Using these values, we first converge the CFD solver using the ANK solver, a highly specialized nonlinear solver in ADflow [20]. Then, using the averaged quantities from the CFD solution, we solve the governing equations of the propulsion system using the Newton solver in OpenMDAO. Finally, all of these intermediate states are used to compute the balance residuals, and we compute an update to the balance states using Eq. 13 until the desired relative nonlinear convergence is achieved.

3.3 Performance Benchmarks with the Solver

The NSC solver has three convergence related parameters: the relative nonlinear convergence of each CFD simulation (η_{CFD}), the relative linear convergence of each solution to obtain the left hand side in Eq. 13 (η_{lin}), and the relative nonlinear convergence of the overall solver (η_{rel}). For all of the cases in this section, we set η_{rel} to 10^{-8}. This is equivalent to converging the CFD nonlinear residuals by 12 orders of magnitude while converging the balance residuals by 6 orders of magnitude with respect to their initial values.

In Table 1, we present the convergence metrics with several combinations of the η_{CFD} and η_{lin} parameters. The η_{CFD} parameter controls how much we reduce the nonlinear residual norm of the CFD model using the ANK solver as shown in Fig. 6. A lower convergence tolerance satisfies the solution of the first discipline in Algorithm 1 more tightly, at the cost of extra computational effort. Similarly, η_{lin} controls how tightly each linear system is solved to compute the left-hand side of Eq. 13. We always converge the first CFD

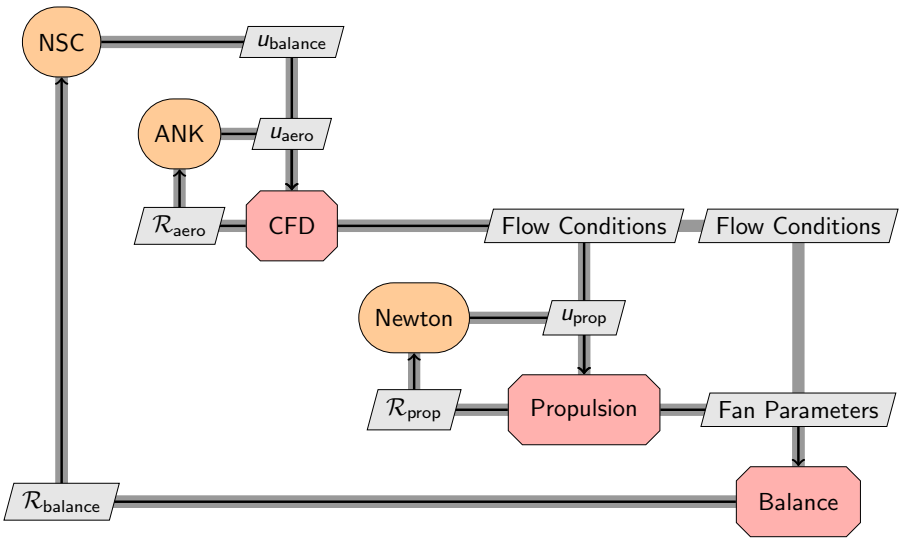


Figure 6: XDSM diagram of the nonlinear solver hierarchy.

nonlinear solution by 10^{-4} because converging this system more loosely causes the propulsion model to fail. The subsequent nonlinear solutions of the CFD model use the specified η_{CFD} value for each case. For all of the cases in this section, we use a single Broadwell node with 28 cores in the Pleiades supercomputer at the NASA Ames Research Center.

Table 1: Convergence metrics of each combination of η_{CFD} and η_{lin} studied.

Case	η_{CFD}	η_{lin}	n_{iter}	Final η_{balance}	Total Time (s)	% Time in CFD	% Time in Linear solver	% Time in Remainder	orders of mag. converged per sec.
1	10^{-2}	10^{-1}	6	2.90×10^{-7}	681.0	32.6	24.5	42.9	9.60×10^{-3}
2	10^{-3}	10^{-1}	5	9.24×10^{-7}	609.5	38.1	22.1	39.8	9.90×10^{-3}
3	10^{-4}	10^{-1}	5	1.72×10^{-6}	617.1	39.0	21.7	39.3	9.34×10^{-3}
4	10^{-4}	10^{-2}	5	1.83×10^{-7}	744.4	32.4	34.8	32.8	9.05×10^{-3}
5	10^{-4}	10^{-3}	5	1.79×10^{-7}	838.4	28.7	42.2	29.1	8.05×10^{-3}

For each case in Table 1, we report the number of NSC solver iterations (n_{iter}) required to converge to the target relative tolerance, the relative convergence of the balance residual norm, total time, and percentages of the total time spent in the CFD nonlinear and linear solutions. We also list the percentage of time spent outside these nonlinear and linear solutions. This portion of the cost accounts for all the data transfer in the multidisciplinary model and the partial matrix-vector multiplications required for Eq. 13. Finally, we provide the orders of magnitude converged per second metric to provide a normalized convergence rate that accounts for the different final residual values and the number of NSC solver iterations.

Cases 1 through 3 vary η_{CFD} from 10^{-2} to 10^{-4} for an η_{lin} value of 10^{-1}. As the η_{CFD} value is decreased, the percentage of total time spent in the CFD nonlinear solver increases, and the relative cost of the remaining operations is reduced. These results show that the optimal convergence rate is attained around an η_{CFD} value of 10^{-3}; however, the overall converge rate is not very sensitive to this parameter.

Cases 3 through 5 vary the η_{lin} parameter from 10^{-1} to 10^{-3} for an η_{CFD} value of 10^{-4}. Similarly, as the η_{lin} value decreases, the computational cost of the linear solutions increases as expected. Compared to the first three cases, these results show that the nonlinear convergence rate per time is more sensitive to the changes in linear solver tolerance; the nonlinear convergence rate increases as the η_{lin} is increased.

In this aeropropulsive model, we need to solve 3 linear systems that correspond to the columns of $\partial\mathcal{R}_{\text{aeroprop}}/\partial u_{\text{balance}}$. These columns correspond to the partial derivatives of the aeropropulsive residuals with respect to the three balance states: static pressure at the fan face ($p_{\text{s,ff}}$), total pressure at the fan exit ($p_{\text{t,fe}}$), and the total temperature at the fan exit ($T_{\text{t,fe}}$). Solving these linear systems more tightly results in

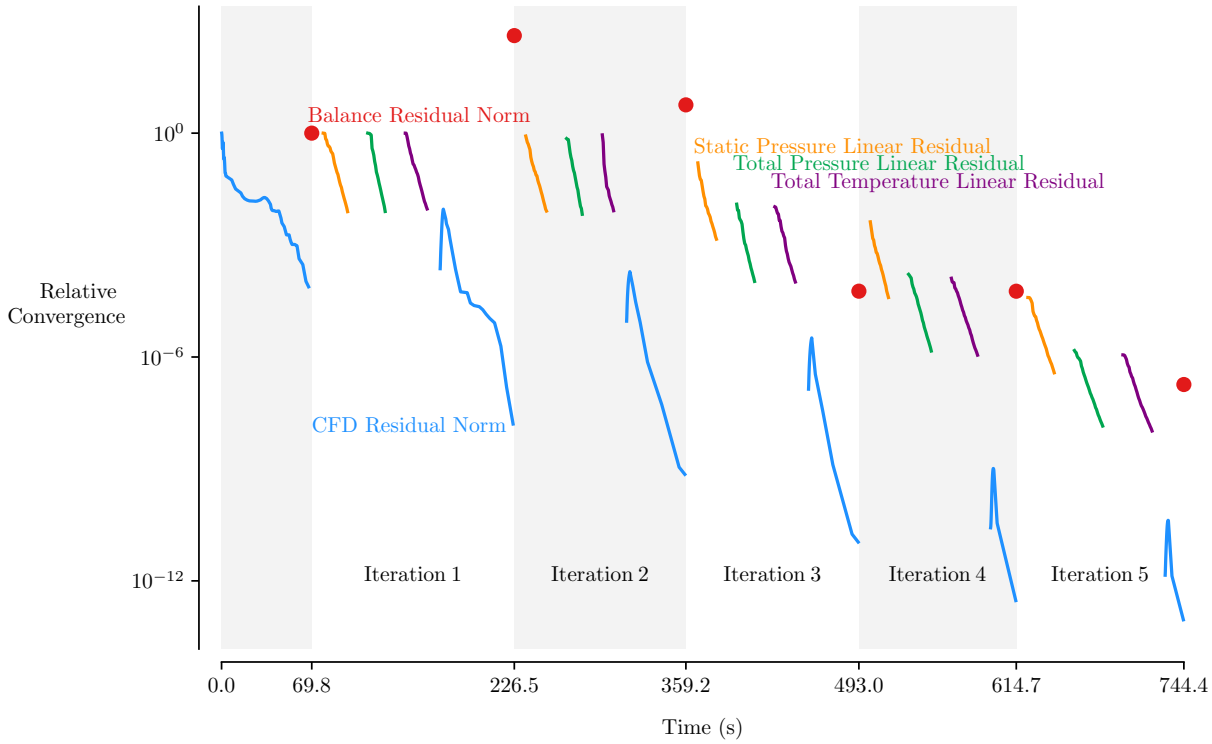


Figure 7: Relative convergence of various components of the NSC solver with respect to time from the initial CFD solution. The blue lines represent the nonlinear residual norm of the CFD model, while the orange, green, and purple lines show the linear residual for the three linear systems solved for the Schur update (Eq. 13). The red dots represent the residual norm of the balance equations.

a more accurate computation of the linear system in Eq. 13 at the cost of extra computational effort. Even though both η_{CFD} and η_{lin} can be set to very small values to closely follow the derivation of the NSC solver, exact solutions of these systems result in high computational costs.

To demonstrate the cost of each stage of the solver, we plot the relative convergence of the nonlinear and linear residuals with the NSC solver with $\eta_{\text{CFD}} = 10^{-4}$ and $\eta_{\text{lin}} = 10^{-2}$ (Case 4) in Fig. 7. As outlined in Algorithm 1, we first converge the CFD residuals partially before the first NSC solver iteration. Then during each NSC solver iteration, we first solve the three linear systems required for the update formula in Eq. 13. These linear systems are restarted from the previous solutions. They are converged to a tolerance of η_{lin} relative to the initial linear residual norm at the beginning of each solution. We then update the balance variables using the NSC solver and the ANK solver to re-converge the CFD model. The solution process is repeated until an η_{rel} value of 10^{-8} is reached, which results in about 12 orders of magnitude reduction in the CFD model residuals and more than 6 orders of magnitude reduction in the balance residuals.

The computational cost of data transfer across the model and the matrix-vector products in Eq. 13 represent a significant portion of the overall solver cost, as shown in Table 1. This effect can also be seen in Fig. 7. Here, the lines represent the times spent in the nonlinear and linear solvers of the CFD model. The gaps between these lines represent the time spent transferring the data across models and computing the Jacobian-vector products with the OpenMDAO model components. These operations can be responsible for up to 42.9% of the total computational cost among the cases we listed in Table 1.

Given these performance results, one option to further accelerate the solver is to avoid solving the linear systems past a given absolute convergence or using a low-rank update to the Schur complement or its inverse. The low-rank update approach is very similar to the methods introduced by Broyden [24]. Another option for reducing the cost of the Schur complement computation is to use an approximated Jacobian for the CFD model, similar to the method introduced in previous work [20].

4 Conclusions

In this work, we introduced an NSC solver developed for a particular class of multidisciplinary models where a given discipline cannot be solved independently. With this class of models, the non-invertible block diagonal of the Jacobian matrix prevents the use of BGS-based methods. One valid option for solving these systems is to use a fully coupled Newton's method; however, this approach also has shortcomings due to the challenging nature of solving a large linear system and lack of robustness.

To address these shortcomings, we developed an NSC solver that computes the update to the non-invertible discipline by using the Schur complement of the fully coupled Jacobian. We then solve the other disciplines using their own specialized linear and nonlinear solvers. This approach enables us to rely on each disciplinary models' specialized linear and nonlinear solvers while obtaining Newton-like convergence rates for the non-invertible discipline.

We demonstrated the effectiveness of the solver using a benchmark aeropropulsive model, where a propulsion model is coupled to a CFD solver through powered BCs. The solver can successfully converge the balance equations and satisfies the continuity of mass, momentum, and energy across the BCs in the CFD domain. The results demonstrate the favorable convergence rates achieved without formulating a fully coupled Newton solver. The flexibility of the NSC solver removes the requirement of formulating these balance problems as nonlinearly constrained optimization problems. Due to this, the solver will be an essential part of simulation-based design optimization frameworks.

5 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. We thank Tim Brooks for reviewing an early version of this paper. The first author acknowledges Andrew Lamkin for providing support in the OpenMDAO implementation of the solver. The second author acknowledges Gaetan Kenway for early discussions on the topic of Schur complement based solver approaches.

References

[1] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–2075, September 2013. doi:10.2514/1.J051895.

[2] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2021. ISBN 9781108833417. doi:10.1017/9781108980647. URL `https://mdobook.github.io`.

[3] David E. Keyes, Lois C. McInnes, Carol Woodward, William Gropp, Eric Myra, Michael Pernice, John Bell, Jed Brown, Alain Clo, Jeffrey Connors, Emil Constantinescu, Don Estep, Kate Evans, Charbel Farhat, Ammar Hakim, Glenn Hammond, Glen Hansen, Judith Hill, Tobin Isaac, Xiangmin Jiao, Kirk Jordan, Dinesh Kaushik, Efthimios Kaxiras, Alice Koniges, Kihwan Lee, Aaron Lott, Qiming Lu, John Magerlein, Reed Maxwell, Michael McCourt, Miriam Mehl, Roger Pawlowski, Amanda P Randles, Daniel Reynolds, Beatrice Riviere, Ulrich Rude, Tim Scheibe, John Shadid, Brendan Sheehan, Mark Shephard, Andrew Siegel, Barry Smith, Xianzhu Tang, Cian Wilson, and Barbara Wohlmuth. Multiphysics simulations: Challenges and opportunities. *International Journal of High Performance Computing Applications*, 27(1):4–83, February 2013. doi:10.1177/1094342012468181.

[4] Gaetan K. W. Kenway, Graeme J. Kennedy, and Joaquim R. R. A. Martins. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. *AIAA Journal*, 52(5):935–951, May 2014. doi:10.2514/1.J052255.

[5] Enrico Fabiano and Dimitri Mavriplis. Adjoint-based aeroacoustic design-optimization of flexible rotors in forward flight. *Journal of the American Helicopter Society*, 62(4):1–17, 2017.

[6] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, April 2019. doi:10.1007/s00158-019-02211-z.

[7] John T. Hwang and Joaquim R. R. A. Martins. A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives. *ACM Transactions on Mathematical Software*, 44(4):Article 37, June 2018. doi:10.1145/3182393.

[8] Michele Benzi, Gene H. Golub, and Jorg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005. doi:10.1017/S0962492904000212.

[9] Joaquim R. R. A. Martins. Perspectives on aerodynamic design optimization. In *AIAA SciTech Forum*, Orlando, FL, January 2020. AIAA. doi:10.2514/6.2020-0043.

[10] Gaetan K. W. Kenway and Joaquim R. R. A. Martins. Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. *Journal of Aircraft*, 51(1):144–160, January 2014. doi:10.2514/1.C032150.

[11] Anil Yildirim, Justin S. Gray, Charles A. Mader, and Joaquim R. R. A. Martins. Coupled aeropropulsive design optimization of a podded electric propulsor. In *AIAA Aviation Forum*, August 2021. doi:10.2514/6.2021-3032.

[12] E. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2003. ISBN 978-0-89871-544-6. doi:10.1137/1.9780898719154.

[13] Jason Felder Welstead. Overview of the NASA STARC-ABL (rev. B) advanced concept, March 2017. URL `https://ntrs.nasa.gov/citations/20170005612`.

[14] David K. Hall and Michael Lieu. Propulsor models for computational analysis of aircraft aerodynamic performance with boundary layer ingestion. In *Proceedings of the AIAA SciTech Forum*. American Institute of Aeronautics and Astronautics, January 2021. ISBN 978-1-62410-609-5. doi:10.2514/6.2021-0991.

[15] Alejandro M. Briones, Andrew W. Caswell, and Brent A. Rankin. Fully coupled turbojet engine computational fluid dynamics simulations and cycle analyses along the equilibrium running line. *Journal of Engineering for Gas Turbines and Power*, 143(6):061019, June 2021. ISSN 0742-4795, 1528-8919. doi:10.1115/1.4049410.

[16] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, August 2012. doi:10.1007/s00158-012-0763-y.

[17] Charles A. Mader, Gaetan K. W. Kenway, Anil Yildirim, and Joaquim R. R. A. Martins. ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, 17(9):508–527, September 2020. doi:10.2514/1.I010796.

[18] Philippe Spalart and Steven Allmaras. A One-Equation Turbulence Model for Aerodynamic Flows. *La Recherche Aerospatiale*, 1:5–21, 1994.

[19] Gaetan K. W. Kenway, Charles A. Mader, Ping He, and Joaquim R. R. A. Martins. Effective adjoint approaches for computational fluid dynamics. *Progress in Aerospace Sciences*, 110:100542, October 2019. doi:10.1016/j.paerosci.2019.05.002.

[20] Anil Yildirim, Gaetan K. W. Kenway, Charles A. Mader, and Joaquim R. R. A. Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397:108741, November 2019. ISSN 0021-9991. doi:10.1016/j.jcp.2019.06.018.

[21] Justin S. Gray, Charles A. Mader, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Coupled aeropropulsive design optimization of a three-dimensional BLI propulsor considering inlet distortion. *Journal of Aircraft*, 57(6):1014–1025, November 2020. doi:10.2514/1.C035845.

[22] Eric S. Hendricks and Justin S. Gray. pyCycle: A tool for efficient optimization of gas turbine engine cycles. *Aerospace*, 6(87), August 2019. doi:10.3390/aerospace6080087.

[23] Scott M. Jones, William J. Haller, and Michael T. Tong. An N+3 technology level reference propulsion system. Technical Report NASA/TM—2017-219501, NASA Glenn Research Center, 2017. URL `https://ntrs.nasa.gov/citations/20170005426`.

[24] Charles G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92):577–593, October 1965. doi:10.1090/S0025-5718-1965-0198670-6.