An automated selection algorithm for nonlinear solvers in MDO

Shamsheer S. Chauhan John T. Hwang Joaquim R. R. A. Martins

Abstract There are two major types of approaches that are used for the multidisciplinary analysis (MDA) of coupled systems: fixed-point-iteration-based approaches and coupled Newton-based approaches. Fixed-point-iteration approaches are easier to implement, but can require a large number of iterations or diverge for strongly coupled problems. On the other hand, coupled-Newton approaches have superior convergence orders, but generally require more effort to implement and have more expensive iterations. Additionally, these two major approaches have many variations, including hybrid approaches where the MDA begins with a fixed-point iteration and then switches to a coupled-Newton approach after a certain number of iterations. However, there is a lack of criteria to govern how to select between these approaches, and when to switch between them in a hybrid approach. This paper compares these approaches and provides an algorithm that can be used to automatically select and switch between them. The proposed algorithm is implemented using OpenMDAO, NASA's open-source framework for multidisciplinary analysis and optimization, and is tested using OpenAeroStruct, an open-source low-fidelity tool for aerostructural optimization. The results show that the proposed algorithm provides a balance of improved robustness and speed.

1 Introduction

Setting up multidisciplinary design optimization (MDO) problems requires making decisions that affect both the speed and robustness of optimizations. Two major decisions are which MDO architecture to use (Martins and Lambe, 2013), and which numerical optimization algorithm to use. For the popular multidisciplinary feasible (MDF) architecture and distributed MDF architectures (Martins and Lambe, 2013), the choice of approach for the multidisciplinary analysis (MDA) is also important.

The purpose of the MDA is to find a solution that satisfies the systems of equations representing the multidisciplinary system, which is a nonlinear system in general. A widely used strategy for the MDA is the nonlinear block Gauss–Seidel (NLBGS) approach, where each discipline is solved sequentially given fixed values of the unknowns from other disciplines (Cervera et al, 1996; Maute et al, 2001; Küttler and Wall, 2008; Joosten et al, 2009; Keyes et al, 2012; Hwang et al, 2014; Gray et al, 2014; Chauhan et al, 2018; Jasa et al, 2018). For example, in a fluid-structure interaction (FSI) problem, given an initial guess for the structural shape, the fluid equations are solved and the tractions on the surfaces are computed. These tractions are then used in the solution of the structural equations to obtain a new deformed shape. This new shape is further used to solve the fluid equations again and the process is repeated until the desired level of convergence is achieved. The block-Jacobi approach is another option, where discipline analyses are solved in parallel. Since the variables are only updated after each iteration instead of being used as soon as they are computed, the block-Jacobi approach requires more iterations than block Gauss–Seidel. These approaches are also commonly referred to as *fixed-point iteration* (FPI) (Haftka et al, 1992), *loosely coupled* (Arian, 1997; Keyes et al, 2012), *partitioned* (Heil, 2004; Keyes et al, 2012), or *block relaxation* schemes (Saad, 2003). Note that when we use the term *relaxation* in the rest of the paper, we mean the use of under- and over-relaxation factors.

When it is possible to compute partial derivatives for the residual equations being solved, approaches based on Newton's method can be used, where the systems of equations for the multiple disciplines are solved simultaneously. This is referred to as the coupled-Newton (CN) approach in this paper. This type of approach is also commonly referred to as *tightly coupled* (Arian, 1997; Keyes et al, 2012) or *monolithic* (Küttler and Wall, 2008; Heil et al, 2008).

FPI approaches are easier to implement with existing solvers and do not require significant modifications or in-depth understanding of the solvers. Additionally, they are less likely to diverge without an initial guess that is sufficiently close to the solution than CN approaches. However, FPI approaches require increasingly large numbers of iterations or diverge as the strength of the coupling between components increases (Haftka et al, 1992; Heil et al, 2008; Chauhan et al, 2018). In this paper, the terms *disciplines* and system *components* are used interchangeably. Also, by one FPI or NLBGS *iteration*, we mean solving every component once.

On the other hand, CN approaches are known for their superior convergence orders and rates. Newton's method exhibits a quadratic order of convergence when sufficiently close to the solution. Due to this property, CN approaches have the potential to reduce computation time over FPI approaches. Additionally, CN approaches can be more robust than FPI approaches as the strength of coupling increases (Heil, 2004; Barcelos et al, 2006; Heil et al, 2008; Chauhan et al, 2018). One disadvantage of CN approaches is that without a good initial guess, the iterations may diverge. Another disadvantage is that CN approaches require setting up and solving one large linear system of equations at each iteration, as opposed to multiple smaller ones in the case of FPI approaches, resulting in more expensive iterations.

There have been various studies in FSI that show the promise of CN approaches (Heil, 2004; Heil et al, 2008; Fernández and Moubachir, 2005; Bazilevs et al, 2006; Sheldon et al, 2014). Fernández and Moubachir (2005) compare NLBGS and CN approaches for the transient flow in a thin elastic vessel that is representative of blood flow in large arteries. Their results show that the CN approach is almost twice as fast as the NLBGS approach (with Aitken's acceleration) for the solution of their test problems. Heil et al (2008) compare NLBGS and CN approaches using the problem of flow in a collapsible channel with an adjustable parameter used to control the strength of the FSI. In their test problems, the approaches are competitive when the coupling is weak. However, when the coupling is strong, the CN approach outperforms the NLBGS approach. Additionally, the CN approach is shown to be more robust and efficient for unsteady problems, while the NLBGS approach diverges rapidly unless strong under-relaxation is applied. Sheldon et al (2014) compare NLBGS and CN approaches using an unsteady flow benchmark (Turek and Hron, 2006). Their results show that the NLBGS approach (with Aitken's acceleration) requires over four times more CPU time than the CN approach.

In the related field of aerostructural optimization, Maute et al (2001) developed an NLBGS method with relaxation for the solution of high fidelity aerostructural systems. Later, to improve robustness and efficiency, Barcelos et al (2006) proposed a Schur–Newton–Krylov method for the solution of coupled aerostructural systems. Kenway et al (2014) addressed shortcomings of prior research by developing a more advanced flow solver and a new parallel structural solver. Additionally, they benchmarked NLBGS (with Aitken's acceleration) and tightly coupled Newton–Krylov approaches for a high-fidelity aerostructural aircraft wing model. Their results show that both methods require similar aerostructural solution times, but the coupled Newton–Krylov method requires less time with certain preconditioning settings.

In an earlier conference paper (Chauhan et al, 2018), we compared NLBGS and CN approaches to study the factors that impact their relative performance. We observed that several factors including the cost of assembling the linear systems involved in the different approaches, the efficiency of the linear solvers used, and the strength of coupling in the problem affect the relative performance of the approaches. Therefore, it is difficult to determine beforehand which approach will be faster for a particular problem.

Since partial derivatives are required for a CN approach, the nature of the problem and the methods used to compute or estimate these derivatives impacts the cost of each CN iteration. For example, this cost may be negligible if simple analytic expressions are available for the derivatives. However, if computing these derivatives is computationally expensive, the cost may greatly impact the performance. NLBGS approaches may also require assembling linear systems for individual components (e.g., finite element analysis and computational fluid dynamics). Since NLBGS approaches tend to require more iterations, the cost of setting up these systems can add up and adversely impact performance. For a linear discipline (e.g., a linear finite element model), the coefficient matrix of the linear system only needs to be set up once during the MDA and this may not have a significant cost. However, for a nonlinear component (e.g., a nonlinear finite element model), the linear system needs to be set up at each iteration and this may have a large impact on the overall performance.

The efficiencies of the solvers used to solve the linear systems involved in the different approaches also play a major role in the relative performance of the different approaches. With CN approaches, larger linear systems need to be solved than with NLBGS approaches. Several options are available for solving linear systems including different variations of direct solvers and different variations of iterative solvers, which in turn have multiple options for preconditioning. Direct solvers are easy to use and efficient for small problems. However, direct solvers do not scale well with problem size and iterative solvers may be the only practical alternative for large problems and CN approaches (Trefethen and Bau III, 1997; Chauhan et al, 2017). The number of iterations required by iterative linear solvers, which impacts cost, depends on the location of the eigenvalues and the conditioning of the linear system (for GMRES, the number of iterations is related to the location of the eigenvalues of the linear system's coefficient matrix and the condition number of the eigenvector matrix for the linear system's coefficient matrix (Trefethen and Bau III, 1997)). This means that a preconditioning strategy is usually required with an iterative linear solver. With multiple options available for preconditioning, this means that the computational performance depends on the effectiveness and cost of the preconditioning strategy. In practice, another factor that impacts the relative performance of the different approaches is the user's particular implementation of the solvers, including the programming languages and frameworks used.

Coupling strength is another important factor that impacts the relative performance of the MDA approaches. Researchers have observed that as the strength of coupling between the components of a multidisciplinary system increases, the number of iterations required by an NLBGS approach also increases, improving the relative performance of CN approaches (Haftka et al, 1992; Heil et al, 2008; Kenway et al, 2014; Chauhan et al, 2018). However, since the performance of CN approaches with iterative solvers depends on the conditioning of the linear system, there is still some dependence of the performance on the coupling strength because of its impact on the conditioning. Additionally, changing the coupling strength in a system (for example, by changing the sweep angle of a wing or the material properties of a wing structure) may also change the proximity of the initial guess to the solution, further impacting the performance of CN approaches.

In addition to computational time, we should also consider robustness when comparing and selecting numerical solution approaches. For computationally expensive optimization problems that require several hours or days to solve, the cost of an analysis failure that prevents the optimization from progressing and completing successfully is high (in terms of both computing time and the amount of time the user has to wait). In such cases, it is preferable to sacrifice some efficiency in exchange for greater robustness.

It is well known that CN approaches may not converge without an initial guess that is sufficiently close to the solution. On the other hand, even with relaxation strategies, it is possible that NLBGS approaches may not converge or may converge too slowly for problems with high coupling strength (Haftka et al, 1992; Heil et al, 2008; Chauhan et al, 2018). If a CN approach is not converging, some initial NLBGS iterations can be used to obtain a better initial guess (Haftka et al, 1992). If an NLBGS approach is not converging, switching to a CN approach may solve the problem.

Based on the above, we conclude that it is difficult to successfully predict beforehand which approach will perform better for a particular problem. Both efficiency and robustness need to be considered for this. Also, during an optimization, the strength of coupling between components, and the proximity of the initial guess to the solution may vary as the optimizer explores the design space, further making it difficult to know beforehand which approach will be the best choice.

To address this problem, we propose a heuristic algorithm that uses convergence rate information to select and switch between CN and NLBGS approaches while solving an MDO problem. This algorithm can help the user save time by reducing the user effort and computational time spent testing approaches. Using the proposed algorithm also reduces the likelihood that an expensive optimization process does not finish due to a convergence failure in the MDA. These two advantages are often underrated. Given the continued decrease in computational cost and increased access to computing resources, and the trade-off between the computational cost and engineering labor, companies increasingly prefer to sacrifice computational efficiency if it means fewer labor hours and increased success in overnight computations. The ultimate goal of the proposed algorithm is to allow engineers and researchers to use tools like OpenMDAO and focus on their design problems and results without having to spend considerable amounts of time studying and experimenting with different options and approaches. This algorithm is the primary new contribution of this work and is described in Section 4. We also propose a new analytic benchmark problem, described in Section 2.2, that is versatile and scalable, since the dimensionality, the sparsity, the nonlinearity, and the coupling strength of the problem can be customized.

Hulme et al (2000) also developed a heuristic MDA algorithm with the goal of building upon the weaknesses of NLBGS and CN approaches to find a balance between efficiency and robustness. However, their algorithm does not select or switch between the above approaches, but instead uses neural network concepts, concurrent sub-optimizations, and data fusion models for an alternative MDA approach. This approach has not gained popularity over the years. In contrast, the algorithm presented in this paper is developed for switching and selecting between existing MDA approaches and is not an alternative MDA approach on its own. We consider our strategy to be much simpler and more practical to implement with existing methods and frameworks.

2 Benchmark problems and tools

In this section, we describe the details of the benchmark problems and the framework used for this work.

2.1 OpenMDAO

OpenMDAO is a Python-based open-source framework for multidisciplinary analysis and optimization developed at NASA (Gray et al, 2010, 2014). It formulates the MDO problem using the modular analysis and unified derivatives (MAUD) architecture (Hwang and Martins, 2018). In MAUD, disciplines are implemented as system *components* that are then combined into a hierarchical structure of *groups* to facilitate the solution of the MDA using a range of approaches and methods. OpenMDAO facilitates the implementation of different MDA approaches and provides the flexibility to switch between them during an analysis or optimization. Additionally, it provides automated derivative computations that can be used for gradient-based optimization. These features make OpenMDAO a convenient and attractive framework for MDA and MDO.

The ability to easily switch between MDA approaches provides a platform for an automated algorithm that selects and switches between MDA approaches during the design optimization of multidisciplinary systems. In the following sections we describe our benchmark problems, which are implemented using OpenMDAO.

2.2 Taylor series-based analytical scalable problem

To study the behavior of MDA approaches as the dimensionality of the problem increases and as other properties like coupling strength vary, we developed an analytical scalable problem (Chauhan et al, 2018). The scalable problem consists of a parametrized system of equations based on a multivariate Taylor series that allows arbitrary control of its dimensionality and other characteristics, such as nonlinearity, structure, and coupling strength. The intent is for this scalable problem to represent systems of equations that arise in practical computational design in terms of problem size, numerical properties (such as conditioning), and problem structure (i.e., the sparsity and structure of the Jacobian matrix). This scalable problem allows users to:

1. define a system of equations using only closed-form mathematical expressions,
2. set the problem size—i.e., the number of variables,
3. divide the equations arbitrarily into groups to represent disciplines,
4. generate both linear and nonlinear equations, and
5. control the coupling between disciplines and the Jacobian structure of the problem.

While benchmark problems exist (Padula et al, 1996; Balling and Wilkinson, 1997; Kodiyalam and Yuan, 1998; Yi et al, 2008; Tosserams et al, 2010; Tedford and Martins, 2010), there is a lack of problems that are scalable and provide the level of flexibility that we desire. Balling and Wilkinson (1997) presented a flexible analytical scalable problem formulation for testing MDO architectures. However, this formulation only generates components with nonlinear governing equations and does not include the option for coupled linear equations. Hulme and Bloebaum (1997) developed a problem formulation that is similar to the one we present. However, our formulation is developed with a focus on being scalable (by using formulas for coefficients instead of storing values) and providing more control over characteristics like coupling strength. Tedford and Martins (2010) also developed an analytical scalable problem to benchmark MDO architectures; however, their formulation does not include nonlinear components.

The advantage of using a Taylor series-based formulation is that it provides flexibility and control over all the various characteristics listed above. This provides the user with the ability to test a range of different types of problems. The formulation (Chauhan et al, 2018) consists of a system of n residual equations that depends on n variables $(v_1, \ldots, v_n)$,

$$\begin{aligned} R_1(v_1, \ldots, v_n) &= 0 \\ &\vdots \\ R_n(v_1, \ldots, v_n) &= 0 \end{aligned} \quad . \tag{1}$$

The LHS of the i^{th} equation in this system is obtained using the Taylor polynomial form:

$$R_i(v_1, \ldots, v_n) = \sum_{r=1}^{d_i} \frac{1}{r!} \sum_{\substack{(j_1,\ldots,j_r) \\ 1 \le j_k \le n \\ j_1,\ldots,j_r \in \mathcal{A}(i)}} \frac{\partial^r R_i}{\partial v_{j_1} \ldots \partial v_{j_r}} \prod_{k=1}^{r} (v_{j_k} - v^*_{j_k}). \tag{2}$$

The partial derivative terms in Eq. (2) are specified by the user. In this paper we use

$$\frac{\partial^r R_i}{\partial v_{j_1} \ldots \partial v_{j_r}} = f(r) \prod_{k=1}^{r} \alpha(i, j_k) g(i, j_k), \tag{3}$$

where

$$\alpha(i, j_k) = \begin{cases} 1, & \text{if } i \text{ and } j_k \text{ belong to the same component} \\ \alpha, & \text{otherwise} \end{cases} .$$

The nomenclature and the problem parameters are detailed in Table 1. The system of equations given by Eq. (1) can be split up arbitrarily to represent components. Increasing the coupling strength amplification factor, α, increases the magnitude of the local sensitivities of the variables of one component with respect to the variables of another.

Table 1: User-specified parameters for the scalable problem formulation

Parameter	Description	Problem characteristic
n	Number of variables	Problem size
d_i	Degree of polynomial	Nonlinearity
v_i^*	Solution value	Solution (set to zero WLOG)
$\mathcal{A}(i)$	Arguments of the i^{th} equation. $\mathcal{A} : \{1, \ldots, n\} \to \mathcal{P}(\{1, \ldots, n\})$ where $\mathcal{P}()$ is the power set.	Jacobian structure and coupling
$f(r)$	A user-defined function of r	Nonlinearity
$g(i, j_k)$	A user-defined function of $\lvert i - j_k \rvert$	Conditioning and coupling strength
$\alpha(i, j_k)$	Coupling strength amplification factor	Coupling strength

The details of the test problems generated using this formulation are provided in Appendix A, and more problem-specific details are provided as the results are discussed in the remainder of this paper.

2.3 OpenAeroStruct

OpenAeroStruct (Jasa et al, 2018) is an open-source low-fidelity tool for aerostructural optimization built with the OpenMDAO framework. It couples a vortex-lattice method (VLM) with a 6 degree-of-freedom finite element model (FEM). The structure for a lifting surface is modeled as a tubular spar, which may not be an accurate representation of typical wing structures, but is useful for studying coupled aerostructural optimization trends. The spanwise spacing specified for the VLM discretization is also used for discretizing the spar structure. OpenAeroStruct is implemented using OpenMDAO in a modular manner, where several components represent disciplines and sub-disciplines. The partial derivatives for the different components are computed using a combination of analytical expressions, automatic differentiated code, and the complex-step approximation (Martins et al, 2003). The total derivatives for optimization are computed using the coupled-adjoint method (Kenway et al, 2014; Martins and Hwang, 2013).

The test problems we use for this paper involve the coupled aerostructural analysis and optimization of a simple trapezoidal wing with different sweep angles. The baseline wing planform geometry and performance characteristics are based on the Bombardier Q400 regional aircraft (see Table 2).

Table 2: Estimates and specifications used for the wing analyses and fuel burn optimizations

Specification	Value
Span	28.4 m
Root chord length	3.34 m
Tip chord length	1.34 m
Thickness-to-chord ratio	0.14
Range	2,000 km
Aircraft zero-fuel weight without wing structure	25,444 kg
Cruise altitude	24,000 ft
Cruise Mach number	0.5
Thrust-specific fuel consumption estimate	$0.43\,\mathrm{lb/lb\,h^{-1}}$
Rest of aircraft zero-lift coefficient of drag	0.0142

For the optimization problems, the objective is to minimize fuel burn by varying the thickness distribution of the tubular spar and the twist distribution of the wing. These distributions are controlled using B-splines with 5 control points each. The fuel burn is estimated using the Breguet range equation with the assumption that the speed, thrust-specific fuel consumption, and lift-to-drag ratio stay constant during the flight. The optimization problems are also subject to constraints to ensure that the lift of the wing is equal to the weight of the aircraft and that the equivalent von Mises stresses in the beam elements do not exceed the yield stress of the specified material with factors of safety. The yield stress is set to 500 MPa (based on Al 7075), and a factor of safety of 2.5×1.5 is used to approximately size for maneuver loads. Also, the elastic axis of the tubular spar is aligned with the 40% chord line of the wing. The optimization problem is summarized using Formulation (4).

$$\begin{aligned}
\text{minimize} \quad & \text{fuel burn} \\
\text{with respect to} \quad & \text{wing twist} \\
& \text{spar thickness} \\
\text{subject to} \quad & \text{lift} = \text{weight} \\
& \sigma_{\text{von Mises}} \leq \tfrac{500\,\mathrm{MPa}}{2.5\times1.5} \\
& -10^\circ \leq \text{wing twist} \leq 10^\circ \\
& 0.002\,\mathrm{m} \leq \text{spar thickness} \leq 0.2\,\mathrm{m}
\end{aligned} \tag{4}$$

We use different sweep angles, initial thickness distributions, and elastic moduli to vary the coupling between the aerodynamic and structural disciplines. We run these problems on a computer with a 2.7 GHz Intel Core i7-7500U processor and 16 GB RAM and discuss the results in Sections 3 and 5. The fork of the OpenAeroStruct GitHub repository used for this paper is publicly available[1].

3 Background

3.1 Convergence order and rate

Different iterative methods display different convergence behaviors. To study these behaviors, we use convergence *order* and convergence *rate*. Suppose $\boldsymbol{x}^*$ is the solution that an iterative process is converging towards. The convergence order and asymptotic error constant are defined by taking the following limit as the number of iterations, k, approaches infinity:

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^p} \leq \gamma, \tag{5}$$

where γ is called the asymptotic error constant and p is the convergence order. If $p = 1$ and there exists some γ such that $0 < \gamma < 1$, then the convergence has a linear order. If $p = 2$ for some finite γ, then the convergence has a quadratic order. In general, if $p > 1$ for some finite γ, then the convergence is superlinear. The above definition is useful for a sufficiently large k (i.e., close to the solution).

However, to track convergence in general as the iteration process progresses, including during early stages, we can define the error ratio as

$$\gamma_k = \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} . \tag{6}$$

If the order of convergence is linear, then γ_k will approach a constant as k increases. If the convergence is superlinear, then γ_k will approach 0 as k increases. The error ratio can also be used to define the rate of convergence as

$$\text{rate} = -\log_{10}(\gamma_k) , \tag{7}$$

which is simply the number of orders of magnitude that the error decreased after an iteration.

It is well known that the order of convergence of Gauss–Seidel and Jacobi FPI is linear. However, with nonlinear problems and dynamic relaxation strategies, this linear convergence behavior is not observed throughout the iteration process, especially during early stages. Similarly, Newton's method is well known for its quadratic convergence, but this quadratic convergence behavior is not necessarily observed throughout the iteration process, especially at early stages. Therefore, the definition given by Eq. (7) is useful for tracking and comparing convergence rates as iterations progress. Since different iterative processes also require different amounts of time per iteration, we use the following temporal rate as a more practical metric for comparison purposes:

$$\text{rate}_\text{t} = -\frac{\log_{10}(\gamma_k)}{t_k} , \tag{8}$$

where t_k is the time required for the k^th iteration.

Additionally, since $\boldsymbol{x}^*$ is usually unknown, the governing equations residuals can be used to track convergence, leading to a residual-based definition for the error ratio,

$$\gamma_k = \frac{\|\boldsymbol{r}_{k+1} - \boldsymbol{r}^*\|}{\|\boldsymbol{r}_k - \boldsymbol{r}^*\|} , \tag{9}$$

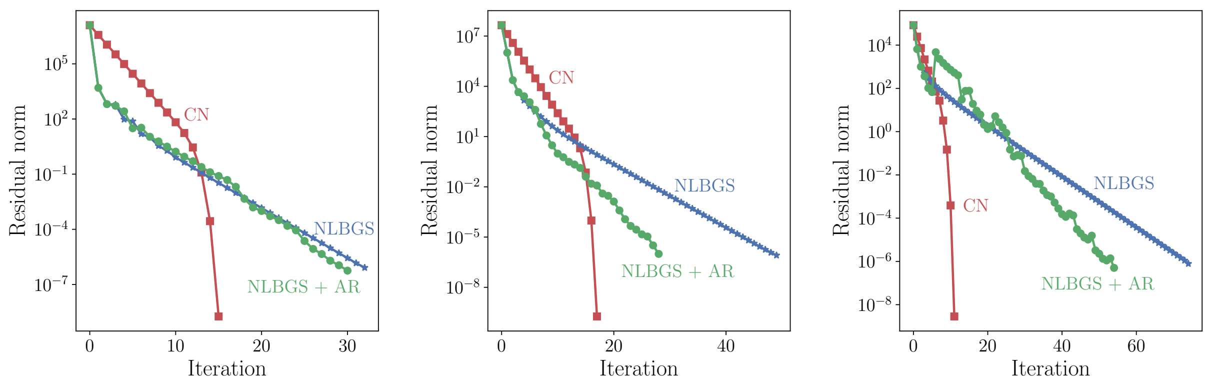
where $\boldsymbol{r}$ is the vector of residuals of the governing equations. Since the residuals are zero at the solution, this can be simplified to

$$\gamma_k = \frac{\|\boldsymbol{r}_{k+1}\|}{\|\boldsymbol{r}_k\|} . \tag{10}$$

We use this residual-based definition to compute convergence rates for the work presented in this paper.

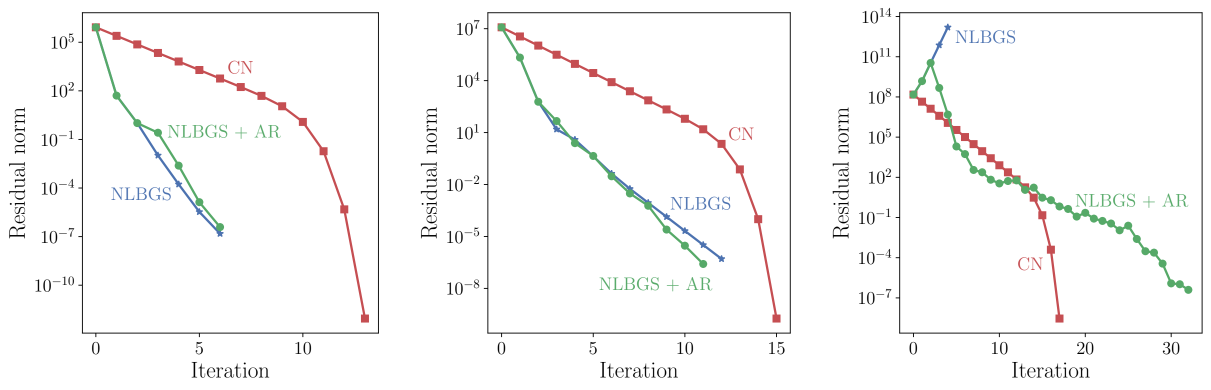
3.2 Comparing NLBGS and CN approaches

The plots in Fig. 1 contrast the convergence behaviors of three MDA approaches using three random nonlinear problems generated with the Taylor series-based scalable problem formulation described in Section 2.2. The problems contain systems of equations representing 20 components, with each component containing between 200 and 1,000 equations. Random number generators are used to specify the degree of the polynomial equations belonging to each component such that, on average, half the components are expected to be linear and the other half are expected to be nonlinear. Among the half of the problems that are expected to be nonlinear, half are expected to be cubic and the other half are expected to be quadratic. Additionally, the coupling strength amplification factor is set to 1 for these problems and the initial guess for every variable is randomly set to an integer between -150 and 150, excluding 0, which is a solution. See Appendix A for more details.



(a) Random problem 1 (b) Random problem 2 (c) Random problem 3

Figure 1: Convergence of three randomly-generated scalable problem examples with different MDA approaches. Convergence tolerance = 10^{-6}. (CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation)



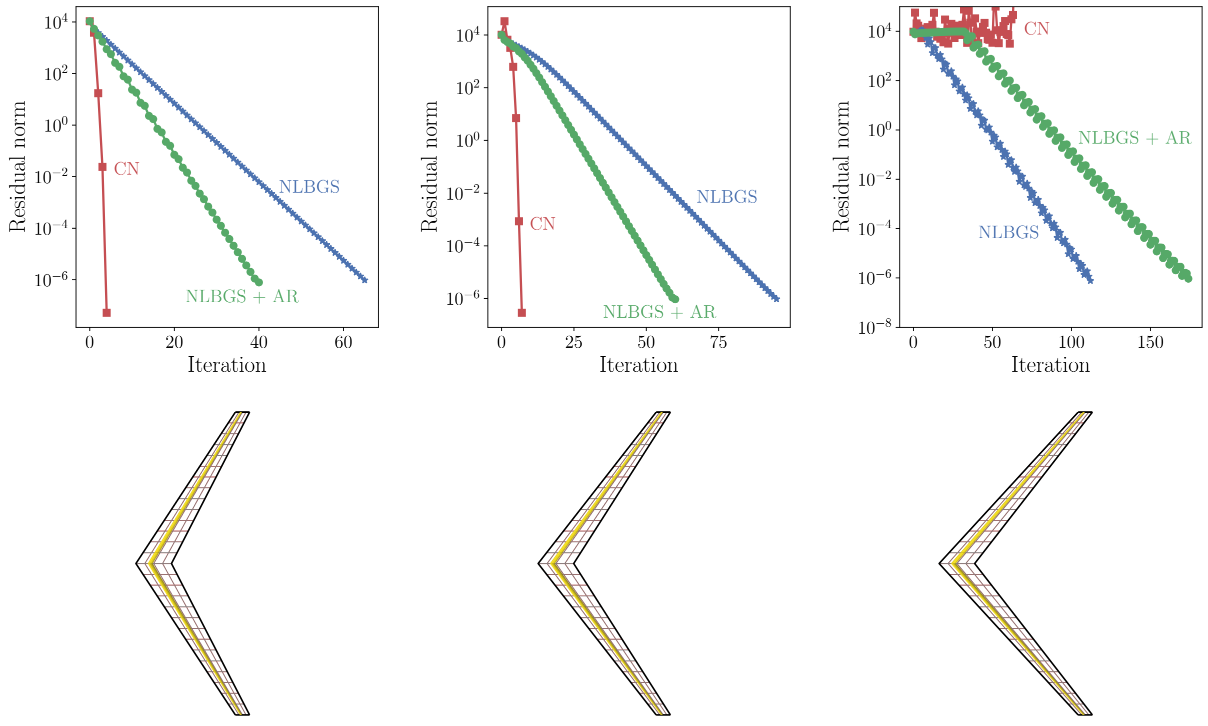
(a) $\alpha = 0.5$ (b) $\alpha = 1.0$ (c) $\alpha = 1.5$

Figure 2: Convergence of a randomly-generated scalable problem example with different MDA approaches and different coupling strength amplification factors, α. Convergence tolerance = 10^{-6}

The CN convergence curves in Fig. 1 initially exhibit linear convergence, followed by superlinear convergence at later stages. This illustrates that the quadratic convergence that Newton-based methods are well known for should not be expected at all stages of convergence. The NLBGS curves display their characteristic linear convergence order, but the slopes change during the initial stages before becoming approximately constant. With Aitken's relaxation (AR), we also observe roughly linear convergence, but there are fluctuations as the relaxation factors change with each iteration. We observed similar behaviors with a set of over 100 randomly-generated problems of varying sizes using the Taylor series-based scalable problem formulation. These examples demonstrate that the MDA approach efficiency depends on the stage of convergence. For nonlinear problems in general, it is difficult to predict at what point one approach has a higher convergence rate than the other, and at what point an approach settles into its characteristic convergence behavior.

Figure 2 shows the convergence plots for another set of randomly-generated nonlinear problems using the Taylor series-based scalable problem formulation. In this set of plots, the same problem settings are used for the problems corresponding to all three subplots, except for the coupling strength amplification factor, α, which increases from left to right. The number of iterations required by the CN approach does not change significantly as the coupling strength amplification factor is increased. On the other hand, the number of iterations required by the NLBGS approaches is more sensitive to the coupling strength factor.

With a coupling strength amplification factor of $\alpha = 1.5$, the NLBGS approach without relaxation did not converge.



(a) 30° sweep (b) 35° sweep (c) 40° sweep

Figure 3: Convergence of three aerostructural analysis problems with different MDA approaches; the corresponding meshes are shown below the plots. Convergence tolerance = 10^{-6}. (CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation)

Similarly, the plots in Fig. 3 show the convergence behaviors for three example problems using OpenAeroStruct. In these plots, the same mesh is used with three sweep angles (30°, 35°, and 40°, where the baseline wing geometry is sheared by these angles). The initial mesh is planar and untwisted, and the initial thickness for the entire tubular spar is 1.5 cm. For the CN approach, we use the GMRES solver in SciPy (Jones et al, 2001) to compute the Newton steps. For preconditioning, we select the linear block Gauss–Seidel option in OpenMDAO. For the NLBGS approaches, the system is solved by cycling through the solver of every component in the coupled group that contains the aerodynamic and structural components. The relaxation factor with AR is limited to a range of 0.1 to 1.5 for these problems, and the convergence tolerance for the L2-norm of the residuals is set to 10^{-6}.

For the 30° sweep case (Fig. 3a), the CN approach converges and exhibits superlinear convergence in a few iterations, and the NLBGS curve has a consistent slope from the early stages. However, for the 35° case (Fig. 3b), the change in slope of the NLBGS curve is more significant during the initial iterations, and CN initially diverges and then converges. For the 40° case (Fig. 3c), CN does not converge and the NLBGS curve shows fluctuations. As the sweep angle is increased in these examples, increasing the coupling strength between the aerodynamic and structural disciplines, the overall slopes of the NLBGS curves decrease in magnitude and the number of iterations required increases.

The curves in Figs. 1 and 3 can be deceiving when trying to infer the relative performance of the CN and NLBGS approaches. Since there is no timing information available in these plots, it is easy to incorrectly conclude that the CN approach can be expected to be much faster in general when it converges. Therefore, for a more significant comparison, we plot the convergence data shown in Fig. 3b, but against wall time instead of the number of iterations in Fig. 4. For this particular problem and implementation, each iteration

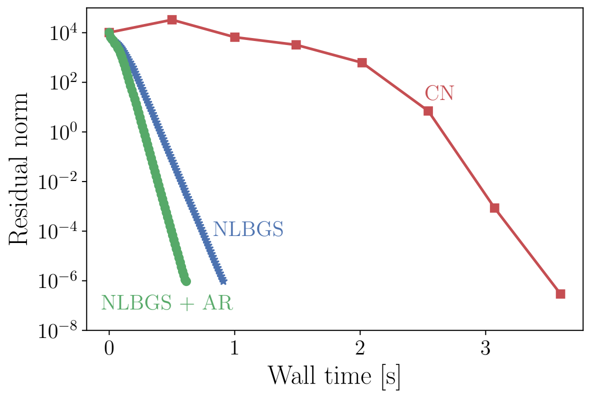


Figure 4: When plotting the convergence data from Fig. 3b against wall time instead of iteration number, we reach a different conclusion. The CN approach requires fewer iterations, but is slower due to the higher cost of each iteration.

with the CN approach happens to require much more time than each NLBGS iteration. So in this case, even though much fewer CN iterations are required, NLBGS converges more quickly. However, this will not be the case in general. As discussed in the introduction, depending on the problem and the implementation, in some cases the CN approach can converge more quickly.

3.3 Coupling strength and general metric

As discussed above, and observed by researchers in the past (Haftka et al, 1992; Heil, 2004; Barcelos et al, 2006; Heil et al, 2008; Chauhan et al, 2018), the strength of coupling between the components of coupled systems impacts the performance of NLBGS approaches. As the coupling strength increases, the number of iterations required for convergence also increases. Here, by *coupling* we mean the two-way information flow between components of a system. In this sense, for example, a system with two components in which only one component depends on information from the other would be considered uncoupled.

In the past, local sensitivities have been used to quantify the strength of the couplings between two components (Bloebaum, 1995). The local sensitivities are defined as the local derivatives representing the change in the output of a component with respect to the change in its input from another. This may be computed analytically or approximated using methods like finite-difference approximations. With this definition, a dependence of one component on another is called a coupling. Note that this is different from the definition of coupling used in this paper, which refers to the two-way dependence between components.

To gain some insight into the relationship between the strength of coupling in a multidisciplinary system and the performance of NLBGS approaches, we can look at the linear block Gauss–Seidel convergence criterion. Consider a multidisciplinary system with n components. The update to the vector of unknown variables, at the end of the k^{th} block Gauss–Seidel iteration for the linearized multidisciplinary system, can be written as

$$\begin{bmatrix}\Delta\boldsymbol{v}^{(1)}\\ \Delta\boldsymbol{v}^{(2)}\\ \vdots\\ \Delta\boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix}\boldsymbol{I} & 0 & \cdots & 0\\ -\frac{\partial\boldsymbol{v}^{(2)}}{\partial\boldsymbol{v}^{(1)}} & \boldsymbol{I} & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ -\frac{\partial\boldsymbol{v}^{(n)}}{\partial\boldsymbol{v}^{(1)}} & \cdots & -\frac{\partial\boldsymbol{v}^{(n)}}{\partial\boldsymbol{v}^{(n-1)}} & \boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}0 & \frac{\partial\boldsymbol{v}^{(1)}}{\partial\boldsymbol{v}^{(2)}} & \cdots & \frac{\partial\boldsymbol{v}^{(1)}}{\partial\boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & \frac{\partial\boldsymbol{v}^{(n-1)}}{\partial\boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}}_{\boldsymbol{G}}\begin{bmatrix}\Delta\boldsymbol{v}^{(1)}\\ \Delta\boldsymbol{v}^{(2)}\\ \vdots\\ \Delta\boldsymbol{v}^{(n)}\end{bmatrix}_{k}, \tag{11}$$

where $\partial\boldsymbol{v}^{(i)}/\partial\boldsymbol{v}^{(j)}$ are the local sensitivities of the variables of the i^{th} component with respect to the variables of the j^{th} component. The development of this equation is included in Appendix B. At each iteration, the previous update is multiplied by $\boldsymbol{G}$. This means that the rate of convergence of the linear block Gauss–Seidel iterative process depends on the spectral radius of the iteration matrix, $\boldsymbol{G}$ (Saad, 2003). The smaller the spectral radius, the faster the convergence. If the spectral radius of $\boldsymbol{G}$ in Eq. (11) is greater than 1, the iterations will not converge without a relaxation strategy. Note that for nonlinear systems, the matrix $\boldsymbol{G}$ will not stay constant as the NLBGS iterations progress. The amount $\boldsymbol{G}$ changes will depend on the nonlinearity and the proximity to the solution.

Furthermore, using the chain rule that relates the variables through the residuals,

$$\frac{\partial \boldsymbol{v}^{(i)}}{\partial \boldsymbol{v}^{(j)}} = -\left(\frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(i)}}\right)^{-1} \frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(j)}} , \tag{12}$$

Eq. (11) can also be written as

$$\begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \cdots & 0 \\ \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}} \end{bmatrix}^{-1} \begin{bmatrix} 0 & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -\frac{\partial \boldsymbol{R}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{\boldsymbol{G}} \begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k} , \tag{13}$$

where $\boldsymbol{R}^{(i)}$ is the vector of residual functions for the i^{th} component. Intermediate steps can be found in Appendix B. Once again, for convergence, the spectral radius of the iteration matrix, $\boldsymbol{G}$, in Eq. (13) should be less than 1. The smaller the spectral radius, the faster the convergence. This spectral radius can be computed using the Jacobian of the governing equations residuals corresponding to the system components, which may be available when using a CN approach or gradient-based optimization with analytic derivatives.

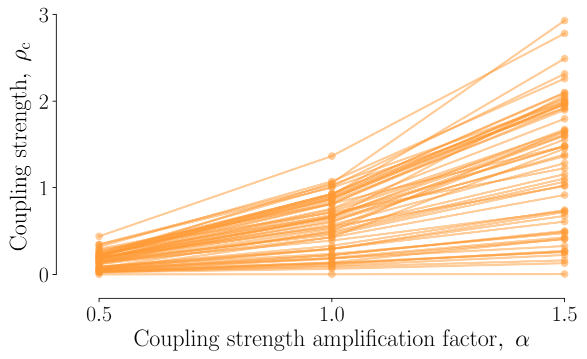
Although the block Gauss–Seidel convergence criteria and iteration matrix (Eq. (13)) are fairly well known, their relationship to local sensitivities (Eq. (11)) and the strength of coupling between components in a multidisciplinary system is not. Experimentally, described in the next paragraph, we observe that as the magnitudes of the local sensitivities increase, and consequently the strength of the coupling between the components increases, the spectral radius of the iteration matrix, $\boldsymbol{G}$, also increases. Since the strength of coupling in a system is usually described qualitatively, we also propose using this spectral radius as a coupling strength metric for the system. This is a more general form of the convergence criterion for two components provided by Haftka et al (1992). Using a more compact form of the iteration matrix, we define the *coupling strength metric* as

$$\rho_c = \text{Spectral radius}\left(\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{lower}}^{-1} \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{upper}}\right) , \tag{14}$$

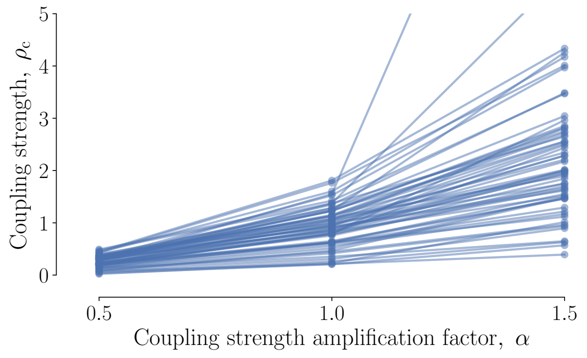
where $[\partial \boldsymbol{R}/\partial \boldsymbol{v}]_{\text{lower}}$ is the block lower triangle of the Jacobian, including the diagonal blocks, and $[\partial \boldsymbol{R}/\partial \boldsymbol{v}]_{\text{upper}}$ is the block upper triangle of the Jacobian excluding the diagonal blocks.

To experimentally test how increasing the strength of coupling between the components of a complex system impacts the spectral radius of the iteration matrix, we generate random instances of the Taylor series-based scalable problem formulation. Each problem consists of 20 systems of equations representing 20 components. We use a set of 60 linear problems with components composed of 200 to 4,000 equations, and a set of 60 nonlinear problems with components composed of 100 to 2,000 equations (we use random number generators to assign the number of equations to each component). We use the settings for these 120 problems to generate three subsets of problems by specifying three different values for the coupling strength amplification factor, α: 0.5, 1.0, and 1.5. The coupling strength amplification factor, α, increases the magnitudes of the partial derivatives in the off-diagonal blocks of Eq. (13). From Eq. (12) we see that this increases the magnitudes of the local sensitivities. The initial guess for every variable is randomly set to an integer between -15 and 15, excluding 0.

Figure 5 shows the coupling strength metric for the problems described above. Plots showing solution times with CN and NLBGS approaches are also included in Appendix C for reference. As the coupling strength amplification factor increases, the coupling strength metric also increases. Note that these are computed using the initial Jacobian of the partial derivatives of the residual equations. This coupling strength metric is by no means a perfect representation of the coupling in a multidisciplinary system. For



(a) Linear problem set



(b) Nonlinear problem set

Figure 5: The coupling strength metric for each problem increases with the coupling strength amplification factor.

the nonlinear problems, these values are expected to change as the MDA iterations progress. The variation of these values will depend on the nonlinearity and proximity to the solution. Also, the order in which components are executed is known to impact the convergence of block Gauss–Seidel iterations (McCulley and Bloebaum, 1996). However, we believe that this metric is useful for quantifying a system's coupling strength, especially when studying the design and optimization of complex systems.

4 An automated selection algorithm

As discussed so far using illustrative examples, it is difficult to conclude beforehand which MDA approach will be better for a particular problem, with respect to both speed and robustness. To address this issue, we propose a heuristic algorithm that switches between NLBGS and CN approaches by using convergence rate information and iteration timings. The pseudo-code for the proposed algorithm is given in Algorithm 1. Algorithms 2 and 3 are the pseudo-codes for the modified NLBGS and CN solvers called by Algorithm 1. These algorithms can be adapted to use other MDA approaches as well. For example, nonlinear block-Jacobi can be used instead of NLBGS by simply solving the components in parallel instead of in sequence (line 8 in Algorithm 2).

In general, the convergence rate of a particular approach evolves as iterations proceed and the particular implementation of an approach also impacts the wall time for each iteration. At different points, depending on the proximity to the solution, different approaches may display superior convergence rates. Therefore, we must use convergence rate information that also takes iteration time into account to determine which approach to select.

The proposed automated selection (AS) algorithm can help the user save time by reducing the effort and computational time spent comparing approaches. Using this algorithm also reduces the likelihood that an

Algorithm 1 Automated selection MDA algorithm

```
1: function AS($\boldsymbol{v}, atol$)
2:     $u_{\text{cache}} \leftarrow \boldsymbol{v}$    ▷ Store initial guess
3:     $flag_{\text{checkCN1}} \leftarrow$ False    ▷ Initialize flag for retesting CN
4:     $flag_{\text{checkCN2}} \leftarrow$ False    ▷ Initialize flag for retesting CN when NLBGS diverges
5:     $u_{\text{CNcache}} \leftarrow$ None    ▷ Initialize variable
6:     $t_1 \leftarrow$ time()    ▷ Start timing
7:     Perform 3 iterations of NLBGS
8:     $t_2 \leftarrow$ time()    ▷ End timing
9:     if $\|\boldsymbol{r}_{\text{latest}}\| < atol$ then
10:        return $\boldsymbol{v}$    ▷ Exit if converged
11:    end if
12:    $t_{\text{NLBGS}} \leftarrow (t_2 - t_1)/3$    ▷ Compute the average time per iteration for NLBGS
13:    $rate_{\text{NLBGS}} \leftarrow \frac{-\log_{10}\left(\frac{1}{2}\frac{\|\boldsymbol{r}_{\text{latest}}\|}{\|\boldsymbol{r}_{\text{latest}-1}\|}+\frac{1}{2}\frac{\|\boldsymbol{r}_{\text{latest}-1}\|}{\|\boldsymbol{r}_{\text{latest}-2}\|}\right)}{t_{\text{NLBGS}}}$    ▷ Estimate NLBGS convergence rate
14:    if $rate_{\text{NLBGS}} < 0$ then
15:        $\boldsymbol{v} \leftarrow u_{\text{cache}}$    ▷ Return to initial guess if NLBGS is diverging
16:    end if
17:    $u_{\text{cache}} \leftarrow \boldsymbol{v}$    ▷ Store a copy of current variables
18:    $t_1 \leftarrow$ time()    ▷ Start timing
19:    Perform 1 iteration of CN
20:    $t_2 \leftarrow$ time()    ▷ End timing
21:    $t_{\text{CN}} \leftarrow t_2 - t_1$
22:    $rate_{\text{CN}} \leftarrow \frac{-\log_{10}\left(\frac{\|\boldsymbol{r}_{\text{latest}}\|}{\|\boldsymbol{r}_{\text{latest}-1}\|}\right)}{t_{\text{CN}}}$    ▷ Compute CN convergence rate
23:    if $rate_{\text{CN}} < 0$ then
24:        $\boldsymbol{v} \leftarrow u_{\text{cache}}$    ▷ Return to previous point if CN diverges
25:    else if $rate_{\text{CN}} < rate_{\text{NLBGS}}$ then
26:        $u_{\text{CNcache}} \leftarrow \boldsymbol{v}$    ▷ Store a copy of variables if CN converges but is not faster
27:    end if
28:    while $\|\boldsymbol{r}_{\text{latest}}\| > atol$ do
29:        if $rate_{\text{NLBGS}} > rate_{\text{CN}}$ or $rate_{\text{CN}} < 0$ then    ▷ Switch to NLBGS if faster or CN is diverging
30:            if $flag_{\text{checkCN1}}$ and $rate_{\text{CN}} < 0$ then
31:                $\boldsymbol{v} \leftarrow u_{\text{cache}}$    ▷ Return to previous point if CN diverged during retest
32:            end if
33:            $t_1 \leftarrow$ time()    ▷ Start timing
34:            Switch to NLBGS solver
35:            $t_2 \leftarrow$ time()    ▷ End timing
36:            $u_{\text{cache}} \leftarrow \boldsymbol{v}$    ▷ Store a copy of current variables
37:            $t_{\text{NLBGS}} \leftarrow (t_2 - t_1)/iter_{\text{NLBGS}}$    ▷ Compute the average time per iteration for NLBGS
38:            $rate_{\text{NLBGS}} \leftarrow \frac{-\log_{10}\left(\frac{1}{2}\frac{\|\boldsymbol{r}_{\text{latest}}\|}{\|\boldsymbol{r}_{\text{latest}-1}\|}+\frac{1}{2}\frac{\|\boldsymbol{r}_{\text{latest}-1}\|}{\|\boldsymbol{r}_{\text{latest}-2}\|}\right)}{t_{\text{NLBGS}}}$    ▷ Estimate NLBGS convergence rate
39:        else
40:            $flag_{\text{checkCN1}} \leftarrow$ False    ▷ Reset flag
41:        end if
42:        if $\|\boldsymbol{r}_{\text{latest}}\| < atol$ then    ▷ Exit if converged
43:            return $\boldsymbol{v}$
44:        end if
45:        if $flag_{\text{checkCN1}}$ is False then
46:            Switch to CN solver
47:            $rate_{\text{CN}} \leftarrow -1$    ▷ Set rate to $< 0$ after exiting CN
48:            $flag_{\text{checkCN2}} \leftarrow$ False    ▷ Reset flag
49:        else
50:            $t_1 \leftarrow$ time()    ▷ Start timing
51:            Perform 1 iteration of CN
52:            $t_2 \leftarrow$ time()    ▷ End timing
53:            $t_{\text{CN}} \leftarrow t_2 - t_1$
54:            $rate_{\text{CN}} \leftarrow \frac{-\log_{10}\left(\frac{\|\boldsymbol{r}_{\text{latest}}\|}{\|\boldsymbol{r}_{\text{latest}-1}\|}\right)}{t_{\text{CN}}}$    ▷ Compute CN convergence rate
55:        end if
56:    end while
57:    return $\boldsymbol{v}$    ▷ The solution is $\boldsymbol{v}$
58: end function
```

expensive optimization process does not finish due to a convergence failure in the MDA. The ultimate goal is to allow engineers and researchers to use tools like OpenMDAO and focus on their design problems and results without having to spend considerable amounts of time studying and experimenting with different options and approaches.

Algorithm 1 begins by calling the NLBGS solver and performing three NLBGS iterations to estimate the convergence rate for the NLBGS approach. We use three as the number of initial iterations to provide a relaxation strategy (like AR) enough iterations to start and aid convergence. A dynamic relaxation strategy may also lead to some fluctuations in the convergence, and therefore, an average rate is estimated using the last two iterations. After this, one iteration of the CN approach is performed to estimate the convergence rate for the CN approach. Only one CN iteration is used because of the higher cost of a CN iteration and also because, if the CN approach does not provide superior convergence at this point, it is unlikely to be the better option at this stage. If the CN approach exhibits a superior convergence rate, then this rate is also likely to increase due to the Newton method's superlinear convergence properties discussed earlier. However, a user may choose to use multiple iterations to estimate the convergence rate of the CN approach (at the cost of speed) to potentially favor robustness.

Next, the algorithm selects the approach with the higher convergence rate, as defined by Eq. (8). If the CN approach converges with a lower rate than the NLBGS approach after the initial iteration is performed, a copy of the unknown variables is stored and is used later as a starting point if the NLBGS approach starts diverging. If both approaches are diverging, the algorithm selects the NLBGS approach. This choice is made with the expectation that, with a relaxation strategy, NLBGS is more likely to converge using heavy under-relaxation than a CN approach (which requires globalization when diverging). In Algorithms 1 to 3, for simplicity and clarity, $\boldsymbol{r}_{\text{latest}}$ is the latest vector of residuals, and $\boldsymbol{r}_{\text{latest}-1}$ and $\boldsymbol{r}_{\text{latest}-2}$ are the vectors of residuals from the previous two iterations. Also, $\boldsymbol{v}$ is the vector of unknown variables that we are solving for.

After the initial rate estimations, the algorithm enters a while loop with a convergence criterion. In the version presented in this paper, the norm of the residuals must be less than a specified absolute convergence tolerance, *atol*. If the NLBGS approach is selected, we follow Algorithm 2. In this algorithm, if the norm of the residuals decreases by an order of magnitude, we exit this algorithm to retest the CN approach and check if a point has been reached where the CN approach has a higher convergence rate. If the CN approach does not have a higher convergence rate, the main algorithm returns to the NLBGS solver. We wait for the norm of the residuals to decrease by an order of magnitude before retesting the CN approach because, based on our experience and experimentation with the problems described in this paper, if the CN approach does not provide a superior convergence rate to begin with, and the initial guess is not very close to the solution, then it is likely to require convergence by multiple orders of magnitude before providing a significantly improved and superior convergence rate.

Before exiting the NLBGS solver to retest the CN approach, we use the convergence rate to estimate the amount of time remaining until convergence for the NLBGS solver. We compare this value to the last measured time for one CN iteration to decide whether it is worth retesting the CN approach. If the predicted time to convergence for the NLBGS approach is less than the last measured time for a CN iteration, then the NLBGS process continues.

In the NLBGS algorithm, we also have some additional conditional statements for robustness (lines 27 to 39). If the NLBGS solver is not converging, we reset the variables to the point that gave the lowest residuals norm and test the CN solver again. If this does not help, and the CN solver was converging when it was initially tested, we reset the unknown variables to the point where the CN approach converged and test it again. This is only attempted once. For the above two checks, we use the simple criterion of checking whether the norm of the residuals has decreased after 10 iterations to determine if it is an appropriate time to test the CN solver again. We use 10 iterations to allow for fluctuations due to nonlinearity and the relaxation strategy, and to avoid checking the more expensive CN solver too frequently.

When the CN approach is selected, either after the initial convergence rate estimations or after switching from the NLBGS approach, we use Algorithm 3. If the CN approach diverges such that the increase in the norm of the residuals is greater than $10 \cdot atol$, then the solver exits and we return to Algorithm 1. The factor of 10 is used to avoid exiting prematurely when there is a small amount of divergence near the solution due to numerical errors and finite precision. If the CN solver diverges when it is not very close to the solution, we choose to exit and try the NLBGS solver for globalization. Depending on how tight the user-specified

Algorithm 2 NLBGS solver algorithm

```
1: function NLBGS($\boldsymbol{v}$, $\boldsymbol{r}$, $t_{\mathrm{CN}}$, $u_{\mathrm{CNcache}}$, $flag_{\mathrm{checkCN1}}$, $flag_{\mathrm{checkCN2}}$, $atol$)
2:     $iter_{\mathrm{NLBGS}} \leftarrow 0$                                   ▷ Initialize iteration counter
3:     $basenorm \leftarrow \|\boldsymbol{r}_{\mathrm{latest}}\|$                        ▷ Store a copy of the residuals norm
4:     $normcopy \leftarrow \|\boldsymbol{r}_{\mathrm{latest}}\|$                        ▷ Store another copy
5:     $u_{\mathrm{cache}} \leftarrow \boldsymbol{v}$                                   ▷ Store a copy of the variables
6:     while $\|\boldsymbol{r}_{\mathrm{latest}}\| > atol$ and $flag_{\mathrm{checkCN1}}$ is False do
7:         $t_1 \leftarrow$ time()                                      ▷ Start timing
8:         Sequentially solve each discipline once
9:         Apply relaxation strategy and compute residuals
10:        $t_2 \leftarrow$ time()                                      ▷ End timing
11:        $iter_{\mathrm{NLBGS}} \leftarrow iter_{\mathrm{NLBGS}} + 1$                       ▷ Increment counter
12:        if $\|\boldsymbol{r}_{\mathrm{latest}}\| < normcopy$ then
13:            $normcopy \leftarrow \|\boldsymbol{r}_{\mathrm{latest}}\|$                    ▷ Store lower residuals norm
14:            $u_{\mathrm{cache}} \leftarrow \boldsymbol{v}$                               ▷ Update copy
15:        end if
16:        if $\|\boldsymbol{r}_{\mathrm{latest}}\|/basenorm < 0.1$ and $iter_{\mathrm{NLBGS}} > 3$ then
17:            $\gamma \leftarrow \left(\frac{1}{2}\frac{\|\boldsymbol{r}_{\mathrm{latest}}\|}{\|\boldsymbol{r}_{\mathrm{latest}-1}\|} + \frac{1}{2}\frac{\|\boldsymbol{r}_{\mathrm{latest}-1}\|}{\|\boldsymbol{r}_{\mathrm{latest}-2}\|}\right)$       ▷ Estimate error ratio
18:            if $\gamma < 1$ then
19:                $n_{\mathrm{remaining}} \leftarrow \log_{10}\left(\frac{atol}{\|\boldsymbol{r}_{\mathrm{latest}}\|}\right) / \log_{10}(\gamma)$
20:                $t_{\mathrm{remaining}} \leftarrow n_{\mathrm{remaining}} \cdot (t_2 - t_1)$      ▷ Estimate time remaining to convergence
21:                if $t_{\mathrm{remaining}} > t_{\mathrm{CN}}$ then
22:                    $flag_{\mathrm{checkCN1}} \leftarrow$ True             ▷ Set flag for retesting CN to True
23:                end if
24:            else
25:                $flag_{\mathrm{checkCN1}} \leftarrow$ True                 ▷ Set flag for retesting CN to True
26:            end if
27:        else if $iter_{\mathrm{NLBGS}} > 10$ then
28:            if $\|\boldsymbol{r}_{\mathrm{latest}}\| > \|\boldsymbol{r}_{\mathrm{latest}-10}\|$ then
29:                if $flag_{\mathrm{checkCN2}}$ is False then
30:                    $\boldsymbol{v} \leftarrow u_{\mathrm{cache}}$
31:                    $flag_{\mathrm{checkCN2}} \leftarrow$ True                       ▷ Reset flag
32:                    $flag_{\mathrm{checkCN1}} \leftarrow$ True             ▷ Set flag for retesting CN to True
33:                else if $u_{\mathrm{CNcache}}$ is not None then
34:                    $\boldsymbol{v} \leftarrow u_{\mathrm{CNcache}}$                  ▷ Return to an earlier point
35:                    $u_{\mathrm{CNcache}} \leftarrow$ None                       ▷ Reset variable
36:                    $flag_{\mathrm{checkCN1}} \leftarrow$ True             ▷ Set flag for retesting CN to True
37:                end if
38:            end if
39:        end if
40:    end while
41:    return $\boldsymbol{v}$, $\boldsymbol{r}$, $u_{\mathrm{CNcache}}$, $flag_{\mathrm{checkCN1}}$, $flag_{\mathrm{checkCN2}}$, $iter_{\mathrm{NLBGS}}$
42: end function
```

convergence tolerance is, a user may choose to adjust the above factor.

The practical implementation of the AS algorithm may require additional instructions not included in the algorithms mentioned above. For example, it may be necessary to keep track of the total number of iterations for each MDA approach in case a maximum number of iterations is specified by the user to prevent the algorithms from looping indefinitely. We have omitted these instructions in the interest of clarity. The implementation (with OpenMDAO version 1.7.3) used herein is available in a GitHub repository[2].

Algorithm 3 CN solver algorithm

```
1: function CN($\boldsymbol{v}$, $\boldsymbol{r}$, $atol$)
2:     while $\|\boldsymbol{r}_{\text{latest}}\| > atol$ do
3:         $t_1 \leftarrow$ time()                                   ▷ Start timing
4:         $u_{\text{cache}} \leftarrow \boldsymbol{v}$                         ▷ Store a copy of the variables
5:         Take a Newton step and compute residuals
6:         $t_2 \leftarrow$ time()                                   ▷ End timing
7:         $t_{\text{CN}} \leftarrow t_2 - t_1$
8:         if $\|\boldsymbol{r}_{\text{latest}}\| - \|\boldsymbol{r}_{\text{latest}-1}\| > 10\ atol$ then
9:             $\boldsymbol{v} \leftarrow u_{\text{cache}}$              ▷ Return to previous point if diverging
10:            return $\boldsymbol{v}$, $\boldsymbol{r}$, $t_{\text{CN}}$                 ▷ Exit if diverging
11:        end if
12:    end while
13:    return $\boldsymbol{v}$, $\boldsymbol{r}$, $t_{\text{CN}}$
14: end function
```

5 Results

5.1 Benchmarking using the scalable problem

First, we present performance results of the AS algorithm tested with problems generated using the Taylor series-based scalable problem formulation described in Section 2.2. We used two sets of nonlinear problems for a total of 600 randomly-generated analysis problems. The first set consists of 300 problems, with each component in a problem containing between 200 and 1,000 equations. The second set consists of another 300 problems, with each component in a problem containing between 500 and 2,000 equations (we use random number generators to assign the number of equations to each component). Each problem consists of 20 components to represent a complex system. This gives us 600 test problems that have between 4,000 and 40,000 variables each.

Each problem set contains two subsets of 150 problems each. One subset has the initial guesses for the variables set to random integers between 15 and -15, except 0 which is a solution. The other subset has the initial guesses for the variables set to random integers between 150 and -150, except 0. This is done to vary the proximity of the initial guess to the solution. For each subset, we generate 50 problems and use the settings of these problems to generate 150 problems by specifying three different values for the coupling strength amplification factor, α: 0.5, 1.0, and 1.5.

We use these problems to compare the performance of the AS, NLBGS, and CN approaches. Before discussing the results, we provide some implementation details specific to these test cases here. Other implementation details are described in Appendix A. The CN approach has an iteration limit of 30, and the NLBGS approach has an iteration limit of 200. Based on initial testing, we found that when the CN solver converges successfully, it usually does so in fewer than 15 iterations; therefore, we set the iteration limit for the CN solver to 30 to avoid excessive iterations when it failed to converge to the specified tolerance. Similarly, we found 200 to be a sufficiently large number of iterations for the NLBGS solver. The absolute convergence tolerance for the fGMRES solver (used for the CN approach) is set to 10^{-12} and the iteration limit is set to 100. The convergence tolerance for the L2-norm of the residuals is set to 10^{-6}. The AS approach switches between the above two approaches with the same settings. Additionally, we must note that these nonlinear analytical problems have multiple solutions. Therefore, different MDA solvers, due to their differing properties, approach different solutions. This provides a challenging set of problems to test an algorithm that switches between approaches.

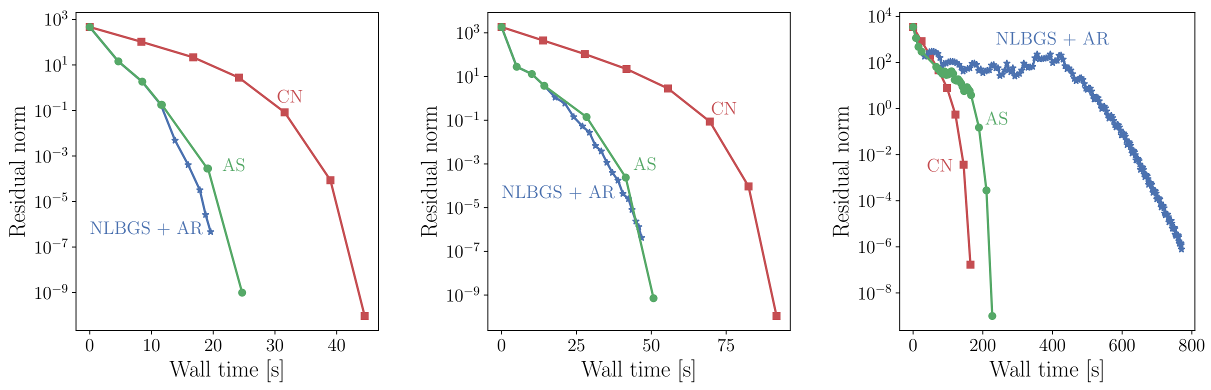
Table 3 shows the mean and median analysis wall times for the test problems. In order for the statistics to be meaningful, they are only included if more than 75% of the problems of the corresponding problem subset and approach converged successfully. Additionally, for fair comparison, if statistics are included, they are for the problems that successfully converged with all the approaches that have statistics included for. We observe that the NLBGS approach is faster than the CN approach for a majority of the problems with the low and moderate coupling strength amplification factors ($\alpha = 0.5$ and 1.0). However, for the problems with high coupling strength, the CN approach is more robust than the NLBGS approach, which frequently diverges despite the relaxation strategy.

Table 3: Wall time statistics for the 600 analysis test problems with the AS, CN, and NLBGS approaches, and varying coupling amplification factor, α. The lowest values are highlighted with bold font, and values are omitted if less than 75% of problems converged.

	$\alpha = 0.5$			$\alpha = 1.0$			$\alpha = 1.5$		
	AS	CN	NLBGS + AR	AS	CN	NLBGS + AR	AS	CN	NLBGS + AR
mean [s]	**34.44**	46.73	34.84	**75.84**	113.82	90.43	**291.73**	313.54	–
median [s]	25.96	34.72	**25.59**	**54.88**	71.78	61.37	191.30	**185.01**	–

Table 4: Percentage of problems that converged with varying coupling amplification factor for the 600 analysis test problems

$\alpha = 0.5$			$\alpha = 1.0$			$\alpha = 1.5$		
AS	CN	NLBGS + AR	AS	CN	NLBGS + AR	AS	CN	NLBGS + AR
100%	100%	100%	99%	99%	84%	87%	89%	35%



(a) $\alpha = 0.5$ (b) $\alpha = 1.0$ (c) $\alpha = 1.5$

Figure 6: Convergence behavior for three of the Taylor series-based scalable problem test cases with varying coupling strength amplification factor, α, testing the automated selection algorithm. Convergence tolerance $= 10^{-6}$. (AS = Automated selection, CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation)

For the problems with the low and moderate coupling strength amplification factors, the AS algorithm either provides the lowest mean and median times, or times very close to the lowest. For the problems with high coupling strength, the AS algorithm provides the same level of robustness as the CN approach. The AS, CN, and NLBGS approaches successfully converged 95%, 96%, and 73% of all the test problems, respectively. Table 4 shows the percentages of problems that converged successfully for the different coupling strength amplification factors. Note that in the cases where the AS or CN approaches failed to converge, it is usually due to a failure to converge tightly enough to the specified tolerance, rather than divergence. We attribute this to numerical error. Figure 13 in Appendix D plots the timings for reference.

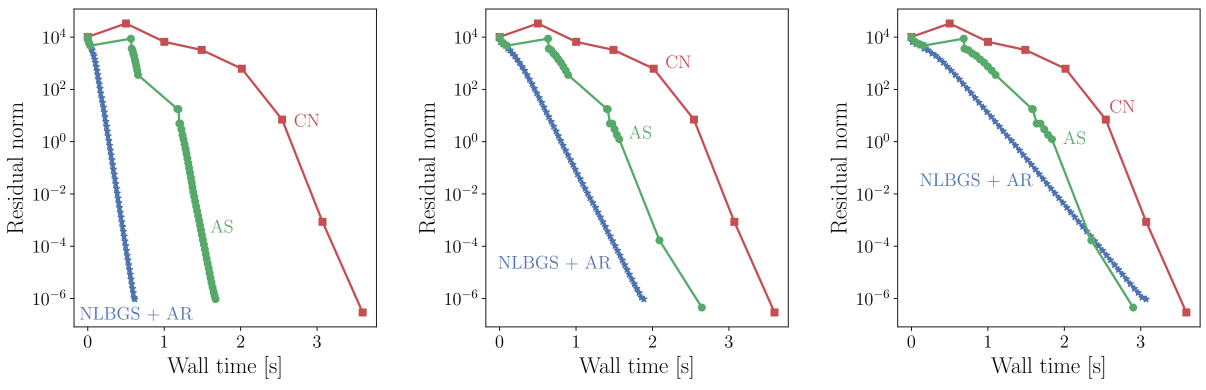
To illustrate the behavior of the AS algorithm, Fig. 6 shows the convergence histories for three of the scalable problem test cases. These three problems have the same settings except for the coupling strength amplification factor, α, which increases left to right. In Figs. 6a and 6b, the algorithm tests the NLBGS and CN approaches, and then switches to the CN approach because the estimated rate for the CN approach is slightly higher. The larger steps in the AS curves show the points at which the CN approach is selected. In Fig. 6c, the algorithm initially selects the NLBGS approach after testing the NLBGS and CN approaches. Then, after converging one order of magnitude, it retests the CN approach and switches to the CN approach due to its higher convergence rate at this point.

To summarize the results with the 600 scalable problem test cases, we find that the AS algorithm is the best overall choice compared to the CN and NLBGS approaches. For the problems with relatively low coupling strength, NLBGS is very efficient, and AS provides similar performance. For the problems with the moderate coupling strength factor, NLBGS suffers from having to perform more iterations because of the increase in coupling, so AS is noticeably faster. For the problems with the high coupling strength factor, NLBGS does not converge for most cases, CN is the best choice, and AS provides similar performance.

5.2 Benchmarking using OpenAeroStruct

Next, we present performance results of the AS algorithm tested with OpenAeroStruct. Figure 7a shows the AS algorithm tested with the OpenAeroStruct analysis example problem used for Fig. 4 (wing with 35° sweep and 1.5 cm spar thickness). To simulate implementations in which the CN approach is more competitive compared to the NLBGS approach (similar to the approaches and problems used in the studies cited in the introduction (Fernández and Moubachir, 2005; Heil et al, 2008; Sheldon et al, 2014; Kenway et al, 2014)), we run the same problem with the NLBGS iterations slowed down by adding pauses to the solver code. Figures 7b and 7c show the results with each NLBGS iteration slowed down by 0.02 and 0.04 seconds, respectively (approximately 5 and 10 times the time for one NLBGS iteration on the computer used).

In Fig. 7a, we can see that the algorithm tests the CN approach once after the initial NLBGS iterations and again after the residuals norm drops an order of magnitude. After this, the algorithm does not test the CN approach again. In Figs. 7b and 7c, we observe that the algorithm switches to the CN approach after the third time that it tests it. This is because the slope of the NLGBS curve is no longer steeper than the CN curve at this point.



(a) Normal speed (b) 0.02 sec slowdown for NLBGS (c) 0.04 sec slowdown for NLBGS

Figure 7: Convergence behavior for the OpenAeroStruct analysis problem testing the automated selection algorithm. For the problems of Figs. (b) and (c), each iteration of the NLBGS solver is slowed down by adding pauses. (AS = Automated selection, CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation)

In earlier sections, we discussed the behavior of the two major MDA approaches and how several factors impact their performance. Once we move further and consider optimization problems, there are more factors that impact overall performance, making it necessary to test the proposed algorithm with optimization problems instead of only analysis problems. For example, if a certain MDA convergence tolerance is specified, the CN approach is more likely to overshoot this tolerance and converge to a lower value due to its quadratic convergence (see Fig. 2). For gradient-based optimization, this difference in accuracies can impact the gradients and the steps taken by the optimizer, potentially leading to different optimization convergence histories (different design points evaluated and different numbers of objective function calls) with the different MDA approaches.

To further test the AS algorithm, we use a set of optimization problems with OpenAeroStruct. The optimization cases consist in fuel burn minimization subject to stress and lift-equals-weight constraints, as described in Section 2.3. We use the gradient-based optimizer SNOPT (Gill et al, 2005) to solve the optimization problems, with optimality and feasibility tolerances set to 10^{-6}. The linear solver for computing gradients for the optimizer is SciPy's (Jones et al, 2001) GMRES, with the absolute convergence tolerance set to 10^{-12} and a limit of 200 iterations. The baseline mesh is based on the Q400 wing using 15 spanwise and 3 chordwise nodes for a half wing. The wing is initially planar and untwisted. The twist and thickness distributions are optimized for different fixed sweep angles (30°, 35°, 40°, and 45° sheared sweep) and different material properties. One set of optimization problems are solved with the Young's modulus and shear modulus for the structure set to 70 GPa and 30 GPa (which are approximate values for aluminum), and another set with 75% of these values. Additionally, for the spar, we use two different initial thicknesses of 2.0 cm and 1.5 cm. The lower initial thickness provides a more challenging start for the MDA approaches.

We compare the optimization results using the multidisciplinary feasible (MDF) architecture and four MDA approaches. The CN approach uses SciPy's GMRES solver to compute the Newton steps. The absolute convergence tolerance for the GMRES solver is 10^{-12} and the iteration limit is 200. For preconditioning, we use the linear block Gauss–Seidel option in OpenMDAO. Based on initial testing, we found that when the CN solver converges, it does so in fewer than ten iterations, so we set the iteration limit for the CN solver to 15 to avoid excessive iterations when it failed to converge. The NLBGS approach cycles through the individual components and uses Aitken's relaxation. The NLBGS iteration limit is 1,000, and the relaxation factor is between 0.01 and 1.5 for these problems. The convergence tolerance for the L2-norm of the residuals is set to 10^{-6}. The AS approach switches between the above two approaches with the same settings. For additional comparison, we also include another hybrid approach that simply switches to the CN approach after ten NLBGS iterations.

Figure 8a shows the total MDA wall times for optimization problems with the different MDA approaches, including the AS approach. Data points are missing for the cases where the optimization failed to converge. Since the number of optimization iterations and objective function calls are not the same with all the MDA approaches for each optimization problem, Fig. 8b is included to show the corresponding average wall time per MDA for the problems described above. In general, the average times per MDA with the AS algorithm and the 10-NLBGS-iteration hybrid approach are in between the average times for the other two approaches.

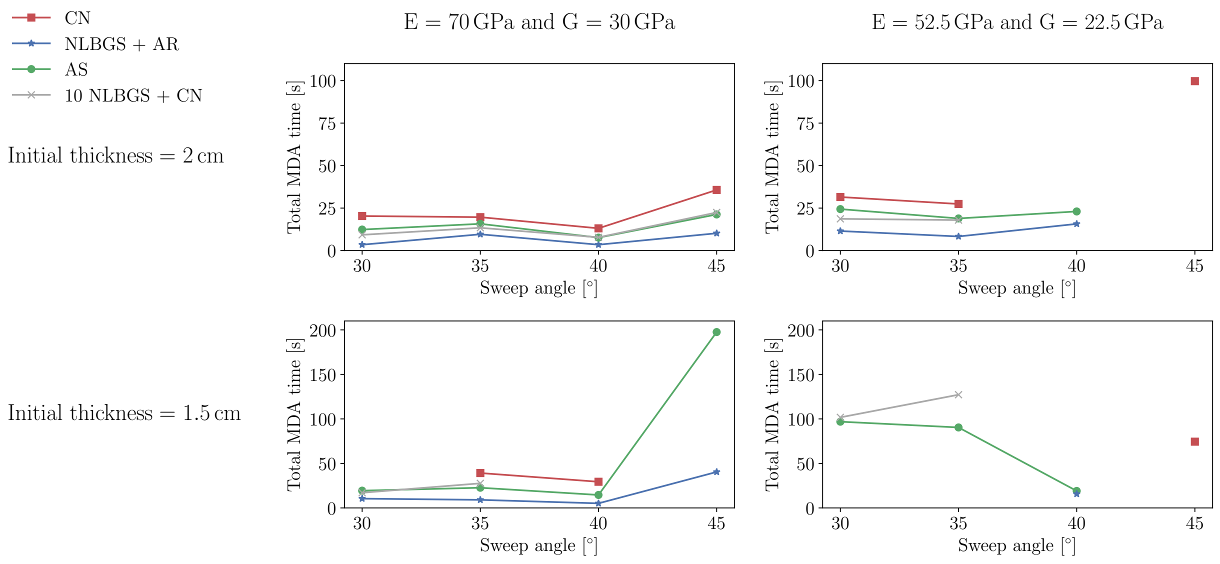
Figure 8 also shows that the AS algorithm is the most robust, with 14 out of the 16 optimization problems converging. Using NLBGS with relaxation, 12 out of the 16 optimization problems converged, and with CN, 10 out of the 16 optimization problems converged. Using the hybrid approach that switches to the CN approach after 10 NLBGS iterations, 10 out of the 16 optimization problems converged.

Since the time per MDA is significantly greater with the CN implementation used for these problems, the same problems are rerun by slowing each NLBGS iteration with a 0.02 second pause (approximately 5 times the time for one NLBGS iteration on the computer used), to simulate an implementation in which the CN approach is more competitive. This impacts the rates that the AS algorithm computes and uses to select an approach. Figure 9a shows the corresponding total MDA wall times for the optimization problems. Figure 9b shows the corresponding average wall times per MDA. Once again, in general, the average times per MDA with the AS algorithm are in between the average times for the CN and NLBGS approaches. Also, the AS algorithm is again the most robust for the optimizations, with 14 out of the 16 optimization problems converging. The total objective function call times, number of objective function calls, and optimized fuel burn values for the above cases are included in Appendix E for reference.

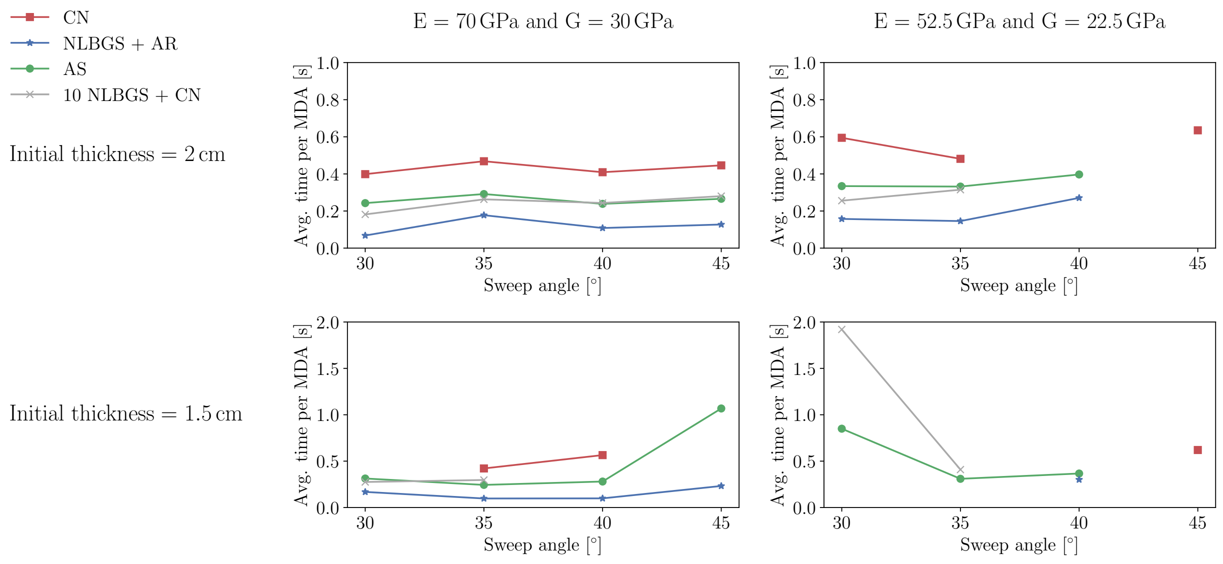
While the proposed AS algorithm does not fix all the shortcomings of the NLBGS and CN approaches, it provides a way to increase robustness and to take advantage of the strengths of each approach.

6 Conclusion

In this work, we compare nonlinear block Gauss–Seidel and coupled-Newton approaches for the MDA of coupled multidisciplinary systems, and propose a novel hybrid algorithm. This hybrid algorithm estimates convergence rates and uses this information to select and switch between the two approaches to provide a balance of efficiency and robustness. We implement this automated selection algorithm in the OpenMDAO framework and this implementation is publicly available. We also propose a new analytic benchmark problem formulation that is versatile and scalable, since the dimensionality, nonlinearity, sparsity, and coupling strength of the problem can be customized.



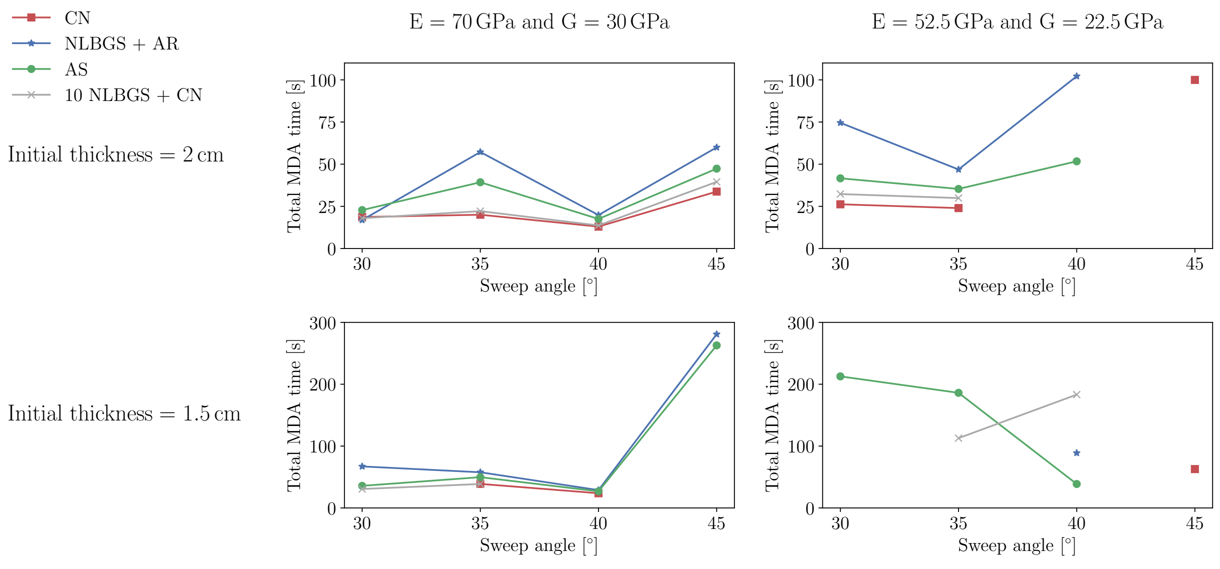
(a) Total wall time for objective function calls during the optimization



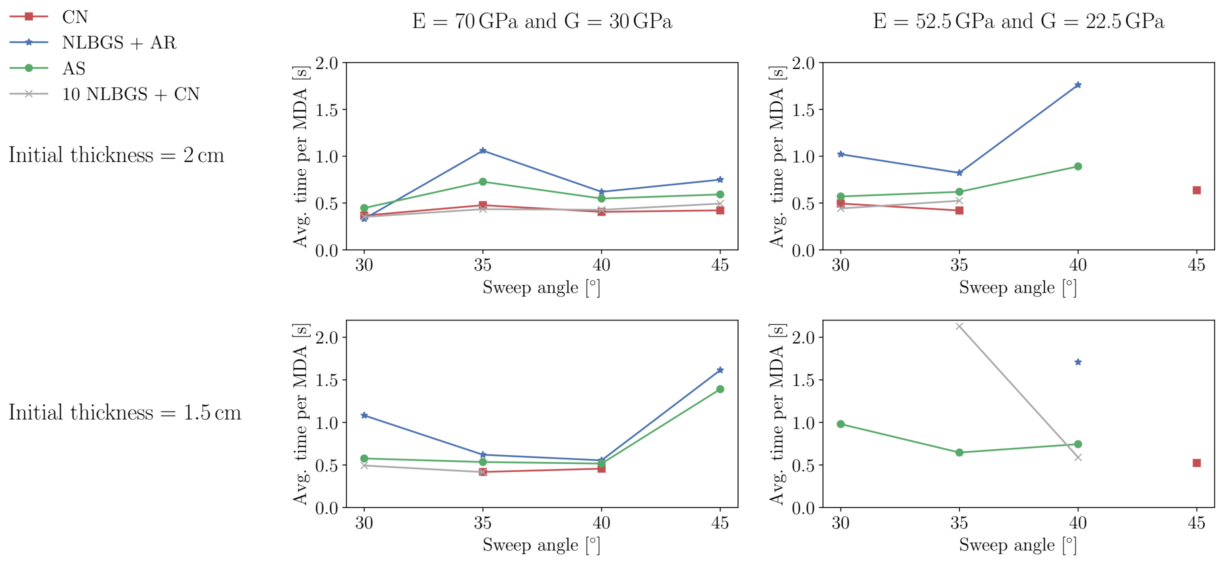
(b) Average wall time per objective function call

Figure 8: MDA times for fuel burn optimization with OpenAeroStruct. AS is the most robust with 14 out of 16 optimization problems converging. (CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation, AS = Automated selection, 10 NBLGS + CN = Switch to CN after 10 NLBGS with AR iterations)

We benchmark the automated selection algorithm using MDA problems constructed with the new Taylor series-based scalable problem formulation, and using optimization problems run with OpenAeroStruct, an open-source low-fidelity tool for aerostructural analysis and optimization. With a set of 600 randomly-generated analysis problems representing complex multidisciplinary systems, we show that this algorithm provides a balance of both speed and robustness. With a set of 16 OpenAeroStruct test problems, the proposed algorithm provides greater robustness while maintaining MDA times in between the times of the other two major approaches for most cases. This algorithm can help users save time by reducing the time



(a) Total wall time for objective function calls during the optimization



(b) Average wall time per objective function call

Figure 9: MDA times for fuel burn optimization with OpenAeroStruct. Each NLBGS iteration is slowed down by 0.02 seconds using a pause function. AS is the most robust with 14 out of 16 optimization problems converging. (CN = Coupled Newton, NLBGS = Nonlinear block Gauss–Seidel, AR = Aitken's relaxation, AS = Automated selection, 10 NBLGS + CN = Switch to CN after 10 NLBGS with AR iterations)

and effort spent testing approaches, and by reducing the likelihood that an expensive optimization process does not finish due to a convergence failure in the MDA.

The ultimate goal of the proposed algorithm is to allow engineers and researchers to use tools like OpenMDAO and focus on their design problems and results without having to spend considerable amounts of time studying and experimenting with different options and approaches. We expect this algorithm to be useful and applicable to a large range of MDO problems, especially those that commonly arise in the design of complex multidisciplinary systems with strong coupling.

Recommendations for future work include testing this algorithm with more MDO applications, further benchmarking it for problems with multiple analysis solutions (Allison et al, 2005), and studying the potential of using it with other MDA approaches like the one developed by Hulme et al (2000). Future work will also include integrating this algorithm with the main OpenMDAO repository so that all users can have access to it by default.

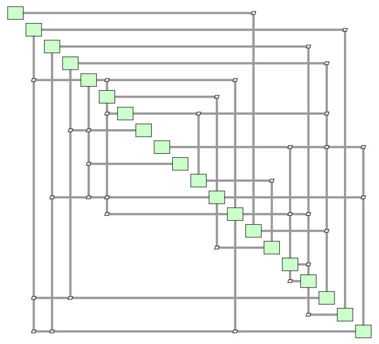
Acknowledgements

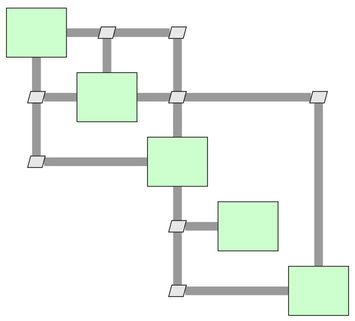
This work was supported by the National Science Foundation (award number 1435188). The authors would also like to thank Justin Gray for his support related to OpenMDAO, and John Jasa for his support related to OpenAeroStruct.

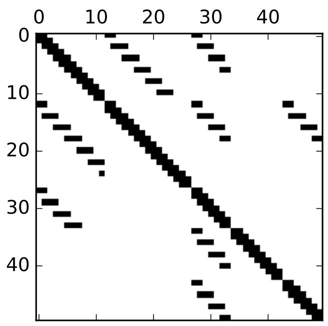
Appendix A

Additional details for the problems generated using the Taylor series-based scalable problem formulation are provided here. We use the scalable problem formulation described in Section 2.2 to generate test problems that consist of 20 components each. We consider 20 as a reasonably representative number of components for complex systems such as aircraft (Hwang and Martins, 2016), satellites (Mosher, 1999; Hwang et al, 2014; Hu et al, 2016), and wind-turbines (Gray et al, 2014; Ning and Petch, 2016), which involve $\mathcal{O}(10)$ disciplines.

The design structure matrices (DSM) (Steward, 1981; Lambe and Martins, 2012) for the problems are randomly generated such that each component depends on at least one other component. A sample randomly-generated component-level Jacobian structure (transpose of N^2 diagram) is shown in Fig. 10a. Figure 10b shows a smaller illustrative example with 5 components and a total of 50 residual equations. Figure 10c shows the Jacobian structure detailing the internal structure of the diagonal and off-diagonal blocks shown in Fig. 10b. The patterns shown in Fig. 10c for the diagonal and off-diagonal blocks are used for the benchmark problems. The component blocks on the diagonal have fully filled tridiagonal structures, and the off-diagonal blocks have tridiagonal structures where every other row is empty.


(a) A sample randomly-generated component-level Jacobian structure with 20 components.


(b) Component-level Jacobian structure for a smaller illustrative example.



(c) Internal structure of the Jacobian for the smaller illustrative example in Fig. 10b.

Figure 10: Jacobian structures (Chauhan et al, 2018)

We generate both linear and nonlinear systems of equations for the test problems. For the nonlinear problems, we use random number generators to specify the degree of the polynomial equations belonging to each component such that, on average, half of the components are expected to be linear, and the other half are expected to be nonlinear. Of the half that are expected to be nonlinear, half are expected to be quadratic, and the other half are expected to be cubic.

The nonlinearity and coupling functions used for the problems are $f(r) = r^{-1}$ and $g(i, j_k) = 1.0001^{-|i-j_k|}$. We provide details on the coupling strength amplification factors, α, and the numbers of equations per component in Sections 3.2, 3.3, and 5.1, as the problems are used. We run these problems on a desktop computer with a 4 GHz Intel Core i7-4790K processor and 32 GB RAM. We implemented the residual and partial derivative computations for the systems of equations in Fortran and wrapped it with Python (Peterson, 2009). The rest of the computations are performed using the Python-based OpenMDAO framework.

When solving these analytical problems with the CN approach, we use the Krylov subspace with preconditioning (KSP) flexible generalized minimal residual (fGMRES) (Saad, 1993) linear solver in PETSc (Balay et al, 1997). For preconditioning, we use the linear block Gauss–Seidel option in OpenMDAO, which sequentially solves smaller linear systems using each component's Jacobian and SciPy's (Jones et al, 2001) `linalg.solve` direct solver. For the NLBGS approaches, we solve the system by cycling through every component. SciPy's `linalg.solve` direct solver is used for solving linear components and the linear Newton systems for nonlinear components. We use OpenMDAO's built-in reordering algorithm to determine the execution order of the components (Gundersen and Hertzberg, 1983; Baharev et al, 2015; Chauhan et al, 2018). For the NLBGS approach with relaxation, we use Aitken's acceleration (Irons and Tuck, 1969) based relaxation (AR) (Chauhan et al, 2018) to help accelerate convergence and prevent divergence. The relaxation factor is limited between 0.25 and 2.0, while the convergence tolerance for the L2-norm of the residuals is set to 10^{-6}.

Appendix B

The development of Eqs. (11) and (13) discussed in Section 3.3 are presented here. This is included to develop the relationship between the convergence criterion of the block Gauss–Seidel method, local sensitivities, and the partial derivatives of residual equations.

We begin with local sensitivities (as mentioned earlier, these are commonly used to quantify the strength of the couplings between two components) and show how they relate to the convergence criterion of block Gauss–Seidel iterations. Consider a multidisciplinary system with n components. The updated vector of variables, at the end of the k^{th} block Gauss–Seidel iteration for the linearized multidisciplinary system, can be written as

$$\begin{aligned}\begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k} &+ \begin{bmatrix}0 & 0 & \cdots & 0\\ \frac{\partial \boldsymbol{v}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ \frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & \frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & 0\end{bmatrix}\left(\begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} - \begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k}\right)\\ &+ \begin{bmatrix}0 & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & \frac{\partial \boldsymbol{v}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}\left(\begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k} - \begin{bmatrix}\boldsymbol{v}^{(1)}\\ \boldsymbol{v}^{(2)}\\ \vdots\\ \boldsymbol{v}^{(n)}\end{bmatrix}_{k-1}\right)\end{aligned}, \tag{15}$$

where $\partial \boldsymbol{v}^{(i)}/\partial \boldsymbol{v}^{(j)}$ are the local sensitivities of the variables of the i^{th} component with respect to the variables of the j^{th} component. This equation can be better understood by thinking about the variables of the components that are solved first and last in the block Gauss–Seidel cycle. The updated variables for the first component can be written as (slope-intercept form)

$$\boldsymbol{v}^{(1)}_{k+1} = \boldsymbol{v}^{(1)}_{k} + \sum_{i=2}^{n} \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(i)}}\left(\boldsymbol{v}^{(i)}_{k} - \boldsymbol{v}^{(i)}_{k-1}\right). \tag{16}$$

Since the variables of other components have not been updated yet, this component uses the update to the variables from the previous iteration, $\left(\boldsymbol{v}^{(i)}_{k} - \boldsymbol{v}^{(i)}_{k-1}\right)$. The updated variables for the last component can be written as

$$\boldsymbol{v}^{(n)}_{k+1} = \boldsymbol{v}^{(n)}_{k} + \sum_{i=1}^{n-1} \frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(i)}}\left(\boldsymbol{v}^{(i)}_{k+1} - \boldsymbol{v}^{(i)}_{k}\right). \tag{17}$$

Since the variables of all the other components have been updated, this component uses the update to the variables from the current iteration, $\left(\boldsymbol{v}^{(i)}_{k+1} - \boldsymbol{v}^{(i)}_{k}\right)$, instead of $\left(\boldsymbol{v}^{(i)}_{k} - \boldsymbol{v}^{(i)}_{k-1}\right)$. The other components between the first and last components will use some combination of the updates to the variables from the current and previous iteration. This is why we see a lower and an upper triangular matrix in Eq (15).

Eq. (15) can be further written in terms of updates to the variables as

$$\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \begin{bmatrix} 0 & 0 & \cdots & 0\\ \frac{\partial \boldsymbol{v}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ \frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & \frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & 0\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} + \begin{bmatrix} 0 & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & \frac{\partial \boldsymbol{v}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k}\ , \tag{18}$$

and simplified to

$$\begin{bmatrix} \boldsymbol{I} & 0 & \cdots & 0\\ -\frac{\partial \boldsymbol{v}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \boldsymbol{I} & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ -\frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & -\frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \begin{bmatrix} 0 & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & \frac{\partial \boldsymbol{v}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k} \tag{19}$$

or

$$\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix} \boldsymbol{I} & 0 & \cdots & 0\\ -\frac{\partial \boldsymbol{v}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \boldsymbol{I} & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ -\frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & -\frac{\partial \boldsymbol{v}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & \boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix} 0 & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{v}^{(1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & \frac{\partial \boldsymbol{v}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}}_{\boldsymbol{G}}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k}\ . \tag{20}$$

At each iteration, the previous update is multiplied by $\boldsymbol{G}$. This means that the rate of convergence of the block Gauss–Seidel iterative process will depend on the spectral radius of the iteration matrix, $\boldsymbol{G}$ (Saad, 2003). The smaller the spectral radius, the faster the convergence. If the spectral radius of $\boldsymbol{G}$ in Eq. (20) is greater than 1, the iterations will not converge without a relaxation strategy. Note that for nonlinear systems, the matrix $\boldsymbol{G}$ will not stay constant as the iterations progress. The amount $\boldsymbol{G}$ changes will depend on the nonlinearity and the proximity to the solution.

Furthermore, using the chain rule,

$$\frac{\partial \boldsymbol{r}^{(i)}}{\partial \boldsymbol{v}^{(j)}} = \frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(j)}} + \frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(i)}}\frac{\partial \boldsymbol{v}^{(i)}}{\partial \boldsymbol{v}^{(j)}} = 0\ , \tag{21}$$

which can be rearranged as

$$\frac{\partial \boldsymbol{v}^{(i)}}{\partial \boldsymbol{v}^{(j)}} = -\left(\frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(i)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(i)}}{\partial \boldsymbol{v}^{(j)}}\ , \tag{22}$$

Eq. (19) can also be written as

$$\begin{bmatrix} \boldsymbol{I} & 0 & \cdots & 0\\ \left(\frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \boldsymbol{I} & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ \left(\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & \left(\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k+1} = \begin{bmatrix} 0 & -\left(\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & -\left(\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \ddots & \vdots\\ \vdots & \vdots & \ddots & -\left(\frac{\partial \boldsymbol{R}^{(n-1)}}{\partial \boldsymbol{v}^{(n-1)}}\right)^{-1}\frac{\partial \boldsymbol{R}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}}\\ 0 & 0 & \cdots & 0\end{bmatrix}\begin{bmatrix}\Delta \boldsymbol{v}^{(1)}\\ \Delta \boldsymbol{v}^{(2)}\\ \vdots \\ \Delta \boldsymbol{v}^{(n)}\end{bmatrix}_{k}\ . \tag{23}$$

Where, $\boldsymbol{R}^{(i)}$ are the residual functions for the i^{th} component and $\boldsymbol{r}^{(i)}$ are the residual values for the i^{th} component. Multiplying both sides by

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \cdots & 0\\ 0 & \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}} & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}}\end{bmatrix}$$

gives

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \cdots & 0 \\ \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \cdots & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n-1)}} & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -\frac{\partial \boldsymbol{R}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k} . \qquad (24)$$

Finally, this can be rearranged to give

$$\begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}} & 0 & \cdots & 0 \\ \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & \frac{\partial \boldsymbol{R}^{(n)}}{\partial \boldsymbol{v}^{(n)}} \end{bmatrix}^{-1} \begin{bmatrix} 0 & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(2)}} & \cdots & -\frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -\frac{\partial \boldsymbol{R}^{(n-1)}}{\partial \boldsymbol{v}^{(n)}} \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{\boldsymbol{G}} \begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \\ \vdots \\ \Delta \boldsymbol{v}^{(n)} \end{bmatrix}_{k} . \qquad (25)$$

Once again, for convergence, the spectral radius of the iteration matrix, $\boldsymbol{G}$, in Eq. (25) should be less than 1. This spectral radius can be computed using the Jacobian of the residual equations of the governing equations of the system components, which may be available when using a coupled-Newton approach or gradient-based optimization with analytic derivatives.

Appendix C

Analysis solution timings for the analytical problems discussed in Section 3.3 are included here to relate the effect of coupling strength. In Figs. 11 and 12, if all problems converged for a particular approach and subset of problems, the data points are blue, otherwise they are red. If less than 75% of the problems converged for a particular approach and subset of problems, the data points are shown, but the box and whiskers are not.

Figure 11 compares the analysis solution times for the nonlinear problem set with the CN approach and two different linear solvers for computing the Newton steps. One set of results corresponds to the CN approach with SciPy's (Jones et al, 2001) `linalg.lu_solve` direct linear solver (DLS), and the second set corresponds to the CN approach with the fGMRES iterative linear solver (ILS) described in Appendix A. As we move from left to right in Fig. 11, we observe that the analysis timings are less sensitive to coupling strength with the DLS than with the ILS. This is attributed to the effect that the coupling strength has on the conditioning of the Newton linear system and on the effectiveness of the preconditioning strategy. Also, as expected, we see that the analysis timings scale poorly with problem size for the CN approach with the DLS.

Figure 12 compares the analysis solution times for the linear and nonlinear problem set with the CN approach (with the ILS as described in Appendix A) and the NLBGS approach with AR. As the coupling strength increases in this figure, moving from left to right, fewer problems converge with the NLBGS approach than with the CN approach.

Appendix D

Analysis solution timings for the analytical test problems discussed in Section 5.1 are plotted here. In Fig. 13, if all problems converged for a particular approach and subset of problems, the data points are blue, otherwise they are red. If less than 75% of the problems converged for a particular approach and subset of problems, the data points are shown, but the box and whiskers are not.

Appendix E

The total objective function call times, number of objective function calls, and optimized fuel burn values for the optimization cases discussed in Section 5.2 are included here. Tables 5 to 8 correspond to the cases without any artificial delay for the NLBGS iterations. Tables 9 to 12 correspond to the cases with a 0.02 sec delay for each NLBGS iteration.

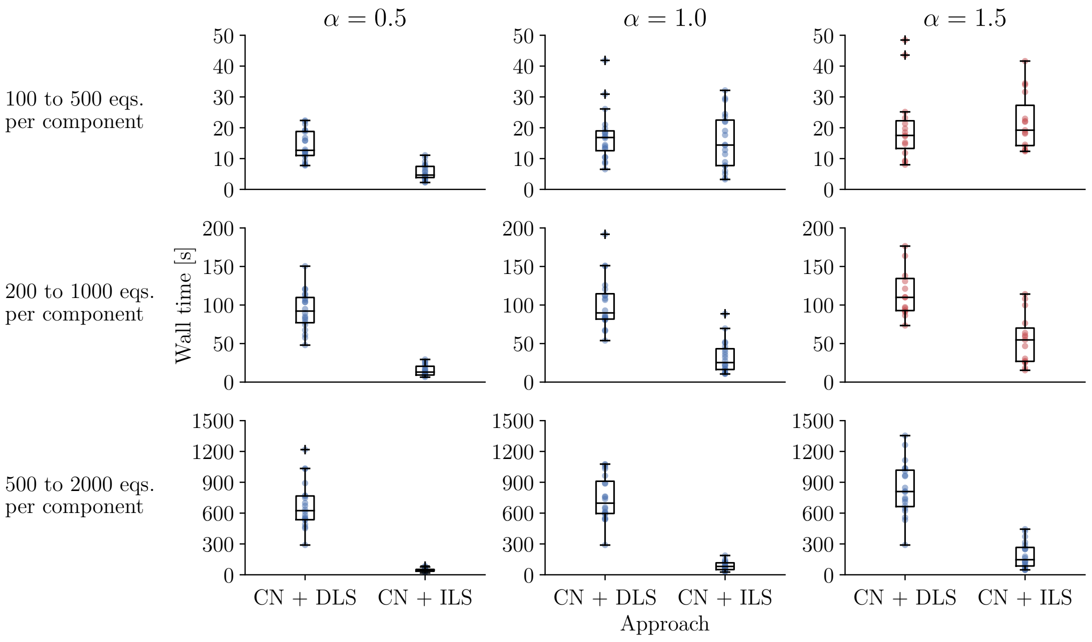


Figure 11: Solution wall time for the nonlinear problem set described in Section 3.3 with CN approaches. The coupling strength amplification factor, α, increases left to right. The analysis solution times are less sensitive to coupling strength with the direct linear solver (DLS) than with the iterative linear solver (ILS). Data points are blue if all problems converged for the combination of approach and problem settings, otherwise they are red. Box and whiskers are not shown if less than 75% of the problems converged.

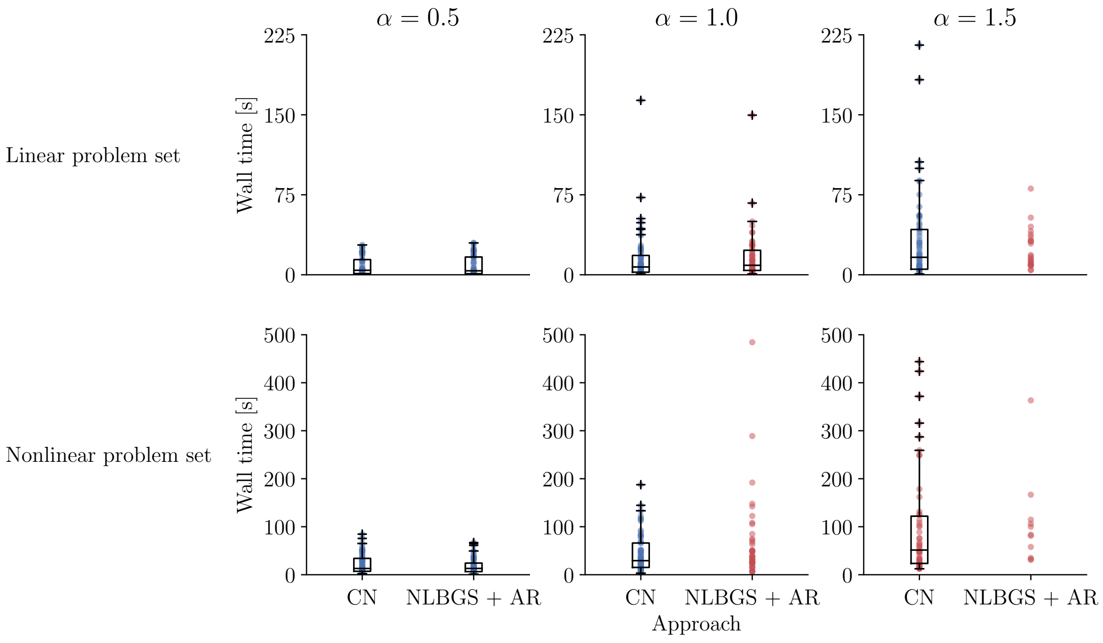


Figure 12: Solution wall time for the linear and nonlinear problem sets described in Section 3.3 with the CN and NLBGS approaches. The CN approach is more robust with respect to coupling strength than the NLBGS approach.

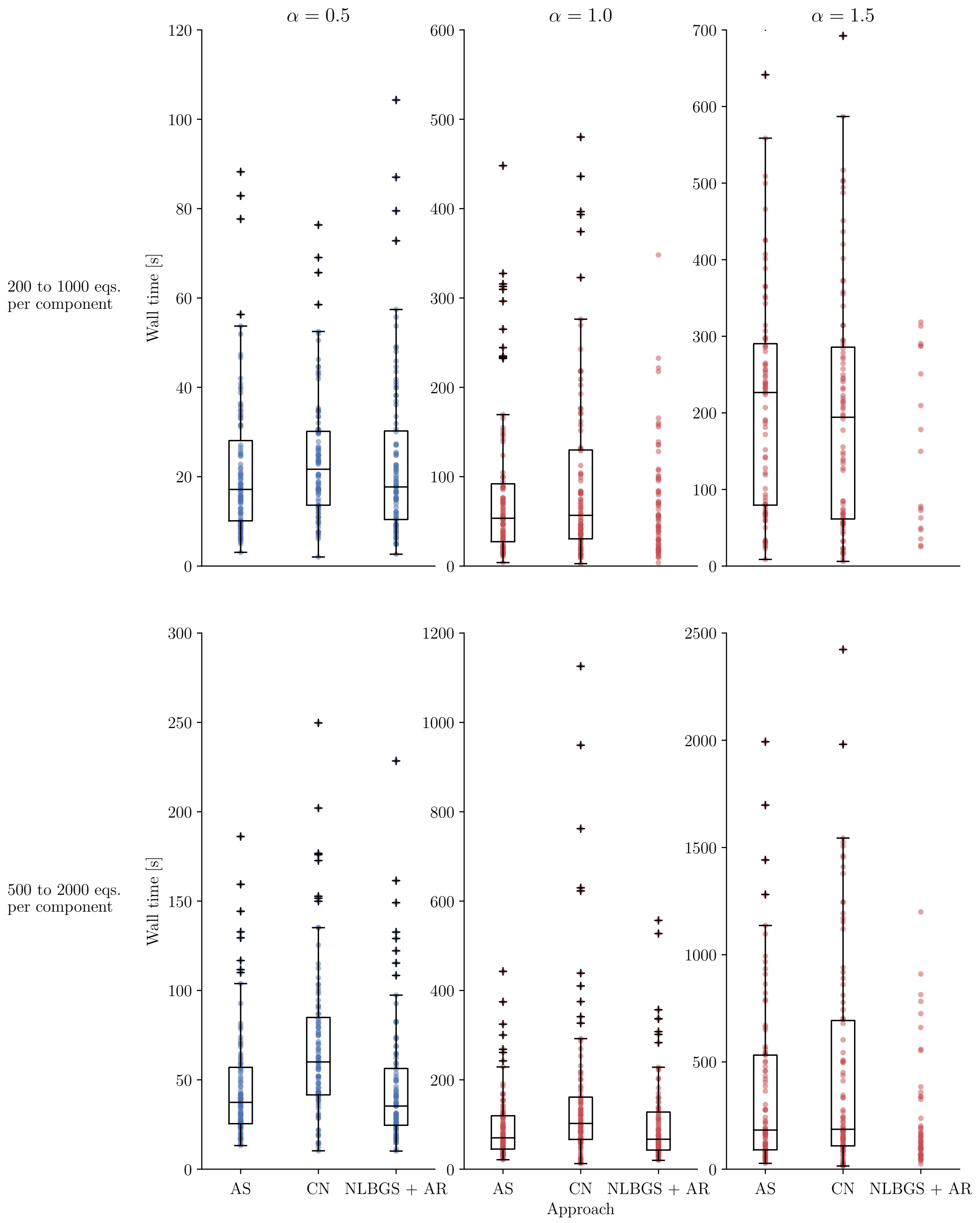


Figure 13: Solution wall time for the analytical problems described in Section 5.1 with the automated selection (AS), CN, and NLBGS approaches. Data points are blue if all problems converged for the combination of approach and problem settings, otherwise they are red. Box and whiskers are not shown if less than 75% of the problems converged. For the subset corresponding to 200 to 1,000 eqs. per component, $\alpha = 1.0$, and AS, an extreme outlier at 2739.4 s has been omitted from the visible area in order to make the other data points more visible. Similarly, for the subsets corresponding to 500 to 2,000 eqs. per component, $\alpha = 1.0$, and AS and CN, two extreme data points at 2136.9 s and 2063.3 s, respectively, have been omitted from the visible area. (CN = Coupled Newton, NLBGS + AR = Nonlinear block Gauss–Seidel + Aitken's relaxation)

Table 5: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 2 cm, E = 70 GPa, and G = 30 GPa

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	3.4632	9.5859	3.4667	10.1850
	No. of objective function calls	51	54	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6203	2453.4013	2564.0994
AS	Total objective function call time [s]	12.3744	15.7428	7.6321	21.2630
	No. of objective function calls	51	54	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6203	2453.4013	2564.0994
CN	Total objective function call time [s]	20.3333	19.6667	13.0925	35.6903
	No. of objective function calls	51	42	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6448	2453.4013	2564.0994
10 NLBGS + CN	Total objective function call time [s]	9.2666	13.4116	7.7866	22.4154
	No. of objective function calls	51	51	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6199	2453.4013	2564.0994

Table 6: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 2 cm, E = 52.5 GPa, and G = 22.5 GPa (DNC = did not converge successfully)

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	11.4933	8.3092	15.7035	-
	No. of objective function calls	73	57	58	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	2455.8987	DNC
AS	Total objective function call time [s]	24.4031	18.9122	23.0266	-
	No. of objective function calls	73	57	58	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	2455.8987	DNC
CN	Total objective function call time [s]	31.5173	27.4315	-	99.6932
	No. of objective function calls	53	57	-	157
	Optimized fuel burn [kg]	2344.3242	2388.5153	DNC	2570.7689
10 NLBGS + CN	Total objective function call time [s]	18.6671	17.9469	-	-
	No. of objective function calls	73	57	-	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	DNC	DNC

Table 7: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 1.5 cm, E = 70 GPa, and G = 30 GPa (DNC = did not converge successfully)

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	10.4334	9.0689	5.1368	40.4214
	No. of objective function calls	62	93	52	174
	Optimized fuel burn [kg]	2341.0272	2385.6186	2453.3352	2564.0986
AS	Total objective function call time [s]	19.3982	22.6836	14.5699	197.5448
	No. of objective function calls	62	93	52	185
	Optimized fuel burn [kg]	2341.0272	2385.6186	2453.3352	2564.0993
CN	Total objective function call time [s]	-	39.1774	29.3865	-
	No. of objective function calls	-	93	52	-
	Optimized fuel burn [kg]	DNC	2385.6186	2453.3352	DNC
10 NLBGS + CN	Total objective function call time [s]	17.0656	27.6118	-	-
	No. of objective function calls	62	93	-	-
	Optimized fuel burn [kg]	2341.0272	2385.6186	DNC	DNC

Table 8: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 1.5 cm, E = 52.5 GPa, and G = 22.5 GPa (DNC = did not converge successfully)

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	-	-	15.6457	-
	No. of objective function calls	-	-	52	-
	Optimized fuel burn [kg]	DNC	DNC	2455.8781	DNC
AS	Total objective function call time [s]	96.8730	90.4025	19.0696	-
	No. of objective function calls	114	292	52	-
	Optimized fuel burn [kg]	2344.3182	2388.5149	2455.8781	DNC
CN	Total objective function call time [s]	-	-	-	74.4103
	No. of objective function calls	-	-	-	120
	Optimized fuel burn [kg]	DNC	DNC	DNC	2570.7677
10 NLBGS + CN	Total objective function call time [s]	101.7250	127.1690	-	-
	No. of objective function calls	53	311	-	-
	Optimized fuel burn [kg]	2344.3207	2388.5151	DNC	DNC

Table 9: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 2 cm, E = 70 GPa, and G = 30 GPa. For these cases the NLBGS iterations are slowed down by 0.02 sec.

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	16.8777	57.2158	19.8381	59.9634
	No. of objective function calls	51	54	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6203	2453.4013	2564.0994
AS	Total objective function call time [s]	22.7925	39.2888	17.5418	47.3558
	No. of objective function calls	51	54	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6203	2453.4013	2564.0994
CN	Total objective function call time [s]	18.6830	20.0364	12.9833	33.7610
	No. of objective function calls	51	42	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6448	2453.4013	2564.0994
10 NLBGS + CN	Total objective function call time [s]	17.9622	22.1656	13.6845	39.5152
	No. of objective function calls	51	51	32	80
	Optimized fuel burn [kg]	2341.0113	2385.6199	2453.4013	2564.0994

Table 10: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 2 cm, E = 52.5 GPa, and G = 22.5 GPa. For these cases the NLBGS iterations are slowed down by 0.02 sec (DNC = did not converge successfully).

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	74.5445	46.8662	102.1477	-
	No. of objective function calls	73	57	58	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	2455.8987	DNC
AS	Total objective function call time [s]	41.6273	35.3379	51.7059	-
	No. of objective function calls	73	57	58	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	2455.8987	DNC
CN	Total objective function call time [s]	26.2211	23.9812	-	99.9636
	No. of objective function calls	53	57	-	157
	Optimized fuel burn [kg]	2344.3242	2388.5153	DNC	2570.7689
10 NLBGS + CN	Total objective function call time [s]	32.2955	29.9007	-	-
	No. of objective function calls	73	57	-	-
	Optimized fuel burn [kg]	2344.3228	2388.5153	DNC	DNC

Table 11: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 1.5 cm, E = 70 GPa, and G = 30 GPa. For these cases the NLBGS iterations are slowed down by 0.02 sec (DNC = did not converge successfully).

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	67.0960	57.6636	28.8311	280.8396
	No. of objective function calls	62	93	52	174
	Optimized fuel burn [kg]	2341.0272	2385.6186	2453.3352	2564.0986
AS	Total objective function call time [s]	35.7094	49.7338	26.8468	262.8498
	No. of objective function calls	62	93	52	189
	Optimized fuel burn [kg]	2341.0272	2385.6186	2453.3352	2564.0964
CN	Total objective function call time [s]	-	38.8701	23.7411	-
	No. of objective function calls	-	93	52	-
	Optimized fuel burn [kg]	DNC	2385.6186	2453.3352	DNC
10 NLBGS + CN	Total objective function call time [s]	30.5977	38.7081	-	-
	No. of objective function calls	62	93	-	-
	Optimized fuel burn [kg]	2341.0272	2385.6186	DNC	DNC

Table 12: Total objective function call time, number of objective function calls, and optimized fuel burn values for the optimization cases with an initial spar thickness of 1.5 cm, E = 52.5 GPa, and G = 22.5 GPa. For these cases the NLBGS iterations are slowed down by 0.02 sec (DNC = did not converge successfully).

		Sweep angle			
MDA approach		30°	35°	40°	45°
NLBGS + AR	Total objective function call time [s]	-	-	88.8133	-
	No. of objective function calls	-	-	52	-
	Optimized fuel burn [kg]	DNC	DNC	2455.8781	DNC
AS	Total objective function call time [s]	212.7032	186.1161	38.7096	-
	No. of objective function calls	217	288	52	-
	Optimized fuel burn [kg]	2344.3192	2388.5149	2455.8781	DNC
CN	Total objective function call time [s]	-	-	-	62.8136
	No. of objective function calls	-	-	-	120
	Optimized fuel burn [kg]	DNC	DNC	DNC	2570.7677
10 NLBGS + CN	Total objective function call time [s]	112.8286	183.2096	-	-
	No. of objective function calls	53	311	-	-
	Optimized fuel burn [kg]	2344.3207	2388.5151	DNC	DNC

References

Allison J, Kokkolaras M, Papalambros P (2005) On the impact of coupling strength on complex system optimization for single-level formulations. In: ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp 265–275, DOI 10.1115/detc2005-84790, URL http://dx.doi.org/10.1115/DETC2005-84790

Arian E (1997) Convergence estimates for multidisciplinary analysis and optimization. Tech. Rep. NASA/CR-97-201752, NAS 1.26:201752, ICASE-97-57, Institute for Computer Applications in Science and Engineering; Hampton, VA United States

Baharev A, Schichl H, Neumaier A, Achterberg T (2015) An exact method for the minimum feedback arc set problem. University of Vienna

Balay S, Gropp WD, McInnes LC, Smith BF (1997) Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, Birkhäuser Press, pp 163–202. DOI 10.1007/978-1-4612-1986-6_8, URL http://doi.org/10.1007/978-1-4612-1986-6_8

Balling R, Wilkinson C (1997) Execution of multidisciplinary design optimization approaches on common test problems. AIAA Journal 35(1):178–186, DOI 10.2514/2.7431, URL http://dx.doi.org/10.2514/2.7431

Barcelos M, Bavestrello H, Maute K (2006) A Schur-Newton-Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis. Computer Methods in Applied Mechanics and Engineering 195(1718):2050–2069, DOI 10.1016/j.cma.2004.09.013, URL http://dx.doi.org/10.1016/j.cma.2004.09.013

Bazilevs Y, Calo VM, Zhang Y, Hughes TJR (2006) Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. Computational Mechanics 38(4):310–322, DOI 10.1007/s00466-006-0084-3, URL http://dx.doi.org/10.1007/s00466-006-0084-3

Bloebaum CL (1995) Coupling strength-based system reduction for complex engineering design. Structural optimization 10(2):113–121, DOI 10.1007/BF01743538, URL http://dx.doi.org/10.1007/BF01743538

Cervera M, Codina R, Galindo M (1996) On the computational efficiency and implementation of block-iterative algorithms for nonlinear coupled problems. Engineering Computations 13(6):4–30, DOI 10.1108/02644409610128382, URL http://doi.org/10.1108/02644409610128382

Chauhan SS, Hwang JT, Martins JRRA (2017) Benchmarking approaches for the multidisciplinary analysis (MDA) of complex systems using a Taylor series-based scalable problem. DOI 10.13140/RG.2.2.12973.49128, URL http://doi.org/10.13140/RG.2.2.12973.49128, a presentation for the 12th World Congress of Structural and Multidisciplinary Optimization, Braunschweig, Germany, June, 2017.

Chauhan SS, Hwang JT, Martins JRRA (2018) Benchmarking approaches for the multidisciplinary analysis of complex systems using a Taylor series-based scalable problem. In: Advances in Structural and Multidisciplinary Optimization: Proceedings of the 12th World Congress of Structural and Multidisciplinary Optimization (WCSMO12), Springer International Publishing, Cham, pp 98–116, DOI 10.1007/978-3-319-67988-4_7, URL http://doi.org/10.1007/978-3-319-67988-4_7

Fernández MÁ, Moubachir M (2005) A Newton method using exact Jacobians for solving fluid-structure coupling. Computers & Structures 83(23):127–142, DOI 10.1016/j.compstruc.2004.04.021, URL http://dx.doi.org/10.1016/j.compstruc.2004.04.021

Gill PE, Murray W, Saunders MA (2005) An SQP algorithm for large-scale constrained optimization. Society for Industrial and Applied Mathematics 47(1), DOI 10.1137/S0036144504446096, URL http://doi.org/10.1137/S0036144504446096

Gray J, Moore KT, Naylor BA (2010) OpenMDAO: An open source framework for multidisciplinary analysis and optimization. In: Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, TX, DOI 10.2514/6.2010-9101, URL http://doi.org/10.2514/6.2010-9101

Gray J, Hearn T, Moore K, Hwang JT, Martins JRRA, Ning A (2014) Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO. In: 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA AVIATION, AIAA, DOI 10.2514/6.2014-2042, URL http://dx.doi.org/10.2514/6.2014-2042

Gundersen T, Hertzberg T (1983) Partitioning and tearing of networks applied to process flowsheeting. Modeling, Identification and Control: A Norwegian Research Bulletin 4(3):139–165, DOI 10.4173/mic.1983.3.2, URL http://doi.org/10.4173%2Fmic.1983.3.2

Haftka RT, Sobieszczanski-Sobieski J, Padula SL (1992) On options for interdisciplinary analysis and design optimization. Structural optimization 4(2):65–74, DOI 10.1007/BF01759919, URL http://doi.org/10.1007/BF01759919

Heil M (2004) An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems. Computer Methods in Applied Mechanics and Engineering 193(12):1–23, DOI 10.1016/j.cma.2003.09.006, URL http://dx.doi.org/10.1016/j.cma.2003.09.006

Heil M, Hazel AL, Boyle J (2008) Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches. Computational Mechanics 43(1):91–101, DOI 10.1007/s00466-008-0270-6, URL http://dx.doi.org/10.1007/s00466-008-0270-6

Hu X, Chen X, Lattarulo V, Parks GT (2016) Multidisciplinary optimization under high-dimensional uncertainty for small satellite system design. AIAA Journal 54(5):1732–1741, DOI 10.2514/1.J054627, URL http://dx.doi.org/10.2514/1.J054627

Hulme K, Bloebaum C, Nozaki Y (2000) A performance-based investigation of parallel and serial approaches to multidisciplinary analysis convergence. In: 8th Symposium on Multidisciplinary Analysis and Optimization, AIAA, DOI 10.2514/6.2000-4812, URL http://doi.org/10.2514/6.2000-4812

Hulme KF, Bloebaum CL (1997) Development of a multidisciplinary design optimization test simulator. Structural optimization 14(2):129–137, DOI 10.1007/BF01812515, URL http://dx.doi.org/10.1007/BF01812515

Hwang JT, Martins JRRA (2016) Allocation-mission-design optimization of next-generation aircraft using a parallel computational framework. In: 57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech, AIAA, DOI 10.2514/6.2016-1662, URL http://dx.doi.org/10.2514/6.2016-1662

Hwang JT, Martins JRRA (2018) A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives. ACM Transactions on Mathematical Software (In press)

Hwang JT, Lee DY, Cutler JW, Martins JRRA (2014) Large-scale multidisciplinary optimization of a small satellite's design and operation. Journal of Spacecraft and Rockets 51(5):1648–1663, DOI 10.2514/1.A32751, URL http://dx.doi.org/10.2514/1.A32751

Irons BM, Tuck RC (1969) A version of the Aitken accelerator for computer iteration. International Journal for Numerical Methods in Engineering 1(3):275–277, DOI 10.1002/nme.1620010306, URL http://dx.doi.org/10.1002/nme.1620010306

Jasa JP, Hwang JT, Martins JRRA (2018) Open-source coupled aerostructural optimization using Python. Structural and Multidisciplinary Optimization DOI 10.1007/s00158-018-1912-8, URL http://doi.org/10.1007/s00158-018-1912-8

Jones E, Oliphant T, Peterson P, et al (2001) SciPy: Open source scientific tools for Python. URL http://www.scipy.org/

Joosten MM, Dettmer WG, Perić D (2009) Analysis of the block Gauss-Seidel solution procedure for a strongly coupled model problem with reference to fluid-structure interaction. International Journal for Numerical Methods in Engineering 78(7):757–778, DOI 10.1002/nme.2503, URL http://dx.doi.org/10.1002/nme.2503

Kenway GKW, Kennedy GJ, Martins JRRA (2014) Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. AIAA Journal 52(5):935–951, DOI 10.2514/1.J052255, URL http://dx.doi.org/10.2514/1.J052255

Keyes D, McInnes L, Woodward C, Gropp W, Myra E, Pernice M (2012) Multiphysics simulations: Challenges and opportunities. The International Journal of High Performance Computing Applications DOI 10.1177/1094342012468181, URL http://dx.doi.org/10.1177/1094342012468181

Kodiyalam S, Yuan C (1998) Evaluation of methods for multidisciplinary design optimization, phase I. Tech. rep., National Aeronautics and Space Administration

Küttler U, Wall WA (2008) Fixed-point fluid-structure interaction solvers with dynamic relaxation. Computational Mechanics 43(1):61–72, DOI 10.1007/s00466-008-0255-5, URL http://dx.doi.org/10.1007/s00466-008-0255-5

Lambe AB, Martins JRRA (2012) Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. Structural and Multidisciplinary Optimization 46(2):273–284, DOI 10.1007/s00158-012-0763-y, URL http://dx.doi.org/10.1007/s00158-012-0763-y

Martins JRRA, Hwang JT (2013) Review and unification of methods for computing derivatives of multidisciplinary computational models. AIAA Journal 51(11):2582–2599, DOI 10.2514/1.J052184, URL http://doi.org/10.2514/1.J052184

Martins JRRA, Lambe AB (2013) Multidisciplinary design optimization: A survey of architectures. AIAA Journal 51(9):2049–2075, DOI 10.2514/1.J051895, URL http://dx.doi.org/10.2514/1.J051895

Martins JRRA, Sturdza P, Alonso JJ (2003) The complex-step derivative approximation. ACM Transactions on Mathematical Software 29(3):245–262, DOI 10.1145/838250.838251, URL http://doi.acm.org/10.1145/838250.838251

Maute K, Nikbay M, Farhat C (2001) Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. AIAA Journal 39(11):2051–2061, DOI 10.2514/2.1227, URL http://dx.doi.org/10.2514/2.1227

McCulley C, Bloebaum CL (1996) A genetic tool for optimal design sequencing in complex engineering systems. Structural optimization 12(2):186–201, DOI 10.1007/BF01196956, URL http://doi.org/10.1007/BF01196956

Mosher T (1999) Conceptual spacecraft design using a genetic algorithm trade selection process. Journal of Aircraft 36(1):200–208, DOI 10.2514/2.2426, URL http://dx.doi.org/10.2514/2.2426

Ning A, Petch D (2016) Integrated design of downwind land-based wind turbines using analytic gradients. Wind Energy 19(12):2137–2152, DOI 10.1002/we.1972, URL http://dx.doi.org/10.1002/we.1972

Padula S, Alexandrov N, Green L (1996) MDO test suite at NASA Langley Research Center. In: 6th Symposium on Multidisciplinary Analysis and Optimization, Multidisciplinary Analysis Optimization Conferences, AIAA, DOI 10.2514/6.1996-4028, URL http://dx.doi.org/10.2514/6.1996-4028

Peterson P (2009) F2PY: A tool for connecting Fortran and Python programs. International Journal of Computational Science and Engineering 4(4):296–305, DOI 10.1504/ijcse.2009.029165, URL http://doi.org/10.1504%2Fijcse.2009.029165

Saad Y (1993) A flexible inner-outer preconditioned GMRES algorithm. SIAM Journal on Scientific Computing 14(2):461–469, DOI 10.1137/0914028, URL http://doi.org/10.1137%2F0914028

Saad Y (2003) Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, DOI 10.1137/1.9780898718003, URL http://doi.org/10.1137/1.9780898718003

Sheldon JP, Miller ST, Pitt JS (2014) Methodology for comparing coupling algorithms for fluid-structure interaction problems. World Journal of Mechanics 4(2), DOI 10.4236/wjm.2014.42007, URL http://dx.doi.org/10.4236/wjm.2014.42007

Steward DV (1981) The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Management EM-28(3):71–74, DOI 10.1109/TEM.1981.6448589, URL http://doi.org/10.1109/TEM.1981.6448589

Tedford NP, Martins JRRA (2010) Benchmarking multidisciplinary design optimization algorithms. Optimization and Engineering 11(1):159–183, DOI 10.1007/s11081-009-9082-6, URL http://dx.doi.org/10.1007/s11081-009-9082-6

Tosserams S, Etman LFP, Rooda JE (2010) A micro-accelerometer MDO benchmark problem. Structural and Multidisciplinary Optimization 41(2):255–275, DOI 10.1007/s00158-009-0422-0, URL http://doi.org/10.1007/s00158-009-0422-0

Trefethen LN, Bau III D (1997) Numerical Linear Algebra. SIAM: Society for Industrial and Applied Mathematics

Turek S, Hron J (2006) Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow, Springer Berlin Heidelberg, Berlin, Heidelberg. DOI 10.1007/3-540-34596-5_15, URL http://dx.doi.org/10.1007/3-540-34596-5_15

Yi SI, Shin JK, Park GJ (2008) Comparison of MDO methods with mathematical examples. Structural and Multidisciplinary Optimization 35(5):391–402, DOI 10.1007/s00158-007-0150-2, URL http://doi.org/10.1007/s00158-007-0150-2