Benchmarking approaches for the multidisciplinary analysis of complex systems using a Taylor series-based scalable problem

Shamsheer S. Chauhan[1], **John T. Hwang**[2], **Joaquim R.R.A. Martins**[3]

[1] University of Michigan, Ann Arbor, MI, USA, sschau@umich.edu
[2] University of Michigan, Ann Arbor, MI, USA, hwangjt@umich.edu
[3] University of Michigan, Ann Arbor, MI, USA, jrram@umich.edu

Abstract

In the practical use of multidisciplinary design optimization, the prevalent approach for the multidisciplinary analyses (MDA) is nonlinear block Gauss–Seidel iteration, which consists in solving each discipline in a sequential manner, and repeating this sequence until convergence. This approach is easy to implement but often exhibits slow convergence rates or does not converge at all. An alternative is to use approaches based on Newton's method to solve the coupled system, also known as tightly coupled or monolithic approaches. Past work, especially in the field of fluid-structure interaction, shows that Newton-based tightly coupled approaches can be more efficient and robust than loosely coupled approaches for the analyses of coupled systems. With the computing power and methods currently available, it is expected that the application of MDA and MDO to systems of greater complexity in terms of coupling and number of disciplines will increase. This makes it important to compare loosely and tightly coupled approaches for complex systems. To address the lack of literature providing such comparisons, we use a novel and highly flexible Taylor series-based analytical scalable problem with OpenMDAO—an open-source framework for MDA and MDO—to compare coupled Newton and nonlinear block Gauss–Seidel approaches for complex systems. We find that assembly time of the linear systems involved, linear solver efficiency, and strength of coupling in the problem play a major role in determining which approach is more efficient for a given problem. We also observe that the coupled Newton approaches are more robust and scale better than the nonlinear block Gauss–Seidel approaches as the strength of coupling between components increases.

Keywords

Complex systems, coupled Newton, block Gauss–Seidel, scalable problem, benchmarking, coupling strength

1 Introduction

Before solving a multidisciplinary design optimization (MDO) problem, we need to decide which architecture and type of optimizer to use. These decisions affect both the speed and robustness of optimizations. For the commonly used multidisciplinary feasible (MDF) architecture [1], another decision that affects the speed and robustness of optimizations is the approach for the multidisciplinary analysis (MDA).

A popular 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 [2–10]. The block Jacobi approach is another option, where discipline analyses are solved in parallel. Since the variables are only updated at each iteration instead of being used as soon as they are computed, the block Jacobi approach takes more iterations than block Gauss–Seidel. These approaches are also often referred to as *fixed point iterations*, *loosely coupled* approaches [8, 11], or *partitioned* approaches [4, 5].

When it is possible to compute derivatives for the residual equations being solved, we can use Newton-based methods, where the systems of equations for the multiple disciplines are solved simultaneously. We refer to this as the coupled Newton (CN) approach in this paper. This type of approach is also often referred to as a *tightly coupled* approach [8, 11] or a *monolithic* approach [4, 5]. Due to its convergence properties, CN has the potential to provide significant reductions in computation time over loosely coupled approaches. Additionally, with a good initial guess, CN approaches can also be more robust than NLBGS type approaches [5, 12, 13].

Recently, tightly coupled approaches have become popular in the field of fluid-structure interaction (FSI) [5, 12, 14–16]. Fernandez and Moubachir [14] compare NLBGS and CN approaches for the transient flow in a thin elastic vessel which 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. [5] 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 loosely coupled 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. [16] compare NLBGS and CN approaches using an unsteady flow benchmark [17]. Their results show that the NLBGS approach (with Aitken's acceleration) requires over 4 times more CPU time than the CN approach. In the related field of aerostructural optimization, Maute et al. [3] developed an NLBGS method with relaxation for the solution of high fidelity aerostructural systems. Later, to improve robustness and efficiency, Barcelos et al. [13] proposed a Schur–Newton–Krylov method for the solution of aerostructural systems. Kenway et al. [18] 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 long-range wide-body aircraft. Their results show that both methods require similar aerostructural solution times, but the coupled Newton–Krylov method required less time when the right preconditioning options were selected.

There is a lack of benchmarking that compares CN and NLBGS approaches in fields outside FSI, especially for complex systems, where use of NLBGS is widespread. This work addresses the need for better benchmarking for such systems by developing a new scalable problem and by studying the factors that influence the relative performance of CN and NLBGS approaches. With the improved computing power and methods currently available, it is expected that the application of numerical analysis and optimization to multidisciplinary systems with greater complexity in terms of coupling and number of disciplines will increase. This makes it important to compare these approaches for those interested in the analysis and optimization of complex systems in the present and the future.

NLBGS approaches are easier to implement with existing solvers and do not require significant modifications or in-depth understanding of the solvers. However, NASA's OpenMDAO framework facilitates the implementation of tightly coupled analyses. OpenMDAO is an open-source Python based computing platform for multidisciplinary analysis and optimization. It formulates the MDA problem using the modular analysis and unified derivatives (MAUD) architecture [19]. In MAUD, disciplines are implemented as system *components* that are then combined into a hierarchical structure of *groups* to facilitate the solution of the multidisciplinary analysis using a range of approaches and methods. In OpenMDAO and MAUD, any system of explicit or implicit equations can be treated as a separate discipline by defining components. OpenMDAO simplifies the assembly and solution of the linear systems involved when using CN approaches. Using Krylov subspace methods, OpenMDAO can solve these systems with lower computation and memory costs than direct methods. OpenMDAO also provides the user with the flexibility to switch between CN and NLBGS approaches, or use hybrid approaches that combine both. Additionally, it provides automated derivative computations that can be used for gradient-based optimization.

The scalable problem developed for this work consists of a parametrized system of equations that allows arbitrary control of its dimensionality and other characteristics, such as nonlinearity and structure. The intent is for this scalable problem to represent actual 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). The scalable problem should:

1. Define a system of equations using only closed-form mathematical expressions.
2. Let users set the problem size—i.e., the number of variables.
3. Let users divide the equations arbitrarily into groups to represent disciplines.
4. Generate both linear and nonlinear equations.

5. Let users control the coupling between disciplines and the Jacobian structure of the problem.

While benchmark MDO problems exist [20–25], there is a lack of problems that are scalable and provide the desired level of flexibility. Balling and Wilkinson [21] 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. Tedford and Martins [25] also developed an analytical scalable problem to benchmark MDO architectures, but the formulation does not include nonlinear components.

In this paper we present an analytical problem formulation based on the multivariate Taylor series. 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 flexibility to test a range of different types of problems and also the ability to set up problems that represent applications more realistically.

2 Scalable problem formulation

The formulation 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.

Table 1: Nomenclature and description of the 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

2.1 Symbolic example

The following is a simple example to illustrate how the above equations are used. For a system with parameters $n = 2$, $d_i = 2$, $v_i^* = i$, $f(r) = 2^{-r}$, $\alpha = 1$, and $g(i, j_k) = 3^{-|i-j_k|}$, the equations corresponding to a fully populated Jacobian ($\mathcal{A}(i) = \{1, 2\}$) are

$$R_1 = \frac{1}{2}(v_1 - 1) + \frac{1}{6}(v_2 - 2) + \frac{1}{8}(v_1 - 1)^2 + 2 \cdot \frac{1}{24}(v_1 - 1)(v_2 - 2) + \frac{1}{72}(v_2 - 2)^2 \tag{4}$$

and

$$R_2 = \frac{1}{6}(v_1 - 1) + \frac{1}{2}(v_2 - 2) + \frac{1}{72}(v_1 - 1)^2 + 2 \cdot \frac{1}{24}(v_1 - 1)(v_2 - 2) + \frac{1}{8}(v_2 - 2)^2. \tag{5}$$

If we want a system with a lower triangular Jacobian, we can exclude the v_2 terms in Eq. (4) and reduce the system to

$$R_1 = \frac{1}{2}(v_1 - 1) + \frac{1}{8}(v_1 - 1)^2 \tag{6}$$

and

$$R_2 = \frac{1}{6}(v_1 - 1) + \frac{1}{2}(v_2 - 2) + \frac{1}{72}(v_1 - 1)^2 + 2 \cdot \frac{1}{24}(v_1 - 1)(v_2 - 2) + \frac{1}{8}(v_2 - 2)^2. \tag{7}$$

This is equivalent to restricting the j_k values in the second summation of the RHS of Eq. (2) to 1 for $i = 1$ (i.e., $\mathcal{A}(1) = \{1\}$).

2.2 Extension to an MDO problem

Although the focus of this paper is on MDA approaches, we also developed a scalable MDO formulation that builds upon the above analysis problem and can be stated as

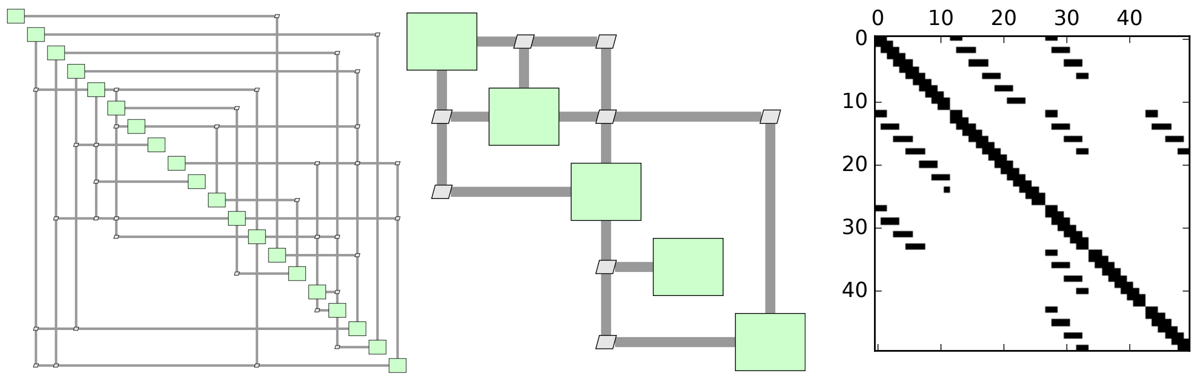
$$\begin{array}{rll} \text{minimize} & R_0(v_1, \dots, v_n) & \\ \text{with respect to} & v_i & i = 1, \dots, n_{\text{dv}} \\ \text{with governing equations} & R_i(v_1, \dots, v_n) = 0 & i = n_{\text{dv}} + 1, \dots, n \\ \text{subject to} & R_i(v_1, \dots, v_n) \geq 0 & i = n + 1, \dots, n + n_{\text{ic}} \end{array} \tag{8}$$

where n is the total number of variables, n_{dv} is the number of design variables, and n_{ic} is the number of inequality constraints. Note that with this MDO formulation, the solution to the governing equations now depends on the design variables. Therefore, the user-specified solution values, v_i^*, for the analysis formulation, become user-specified constants for the MDO formulation and no longer play the role of solution values for the governing equations.

3 Benchmark problem details

For this study, we are interested in the types of complex systems that may arise in engineering design problems. Typically, complex engineering design problems like aircraft [26], satellites [9, 27, 28], and wind-turbines [10, 29] involve $\mathcal{O}(10)$ disciplines. For this reason, we select 20 as a reasonably representative number of components for the benchmark problems. The design structure matrices (DSM) [30, 31] 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. 1a. Fig. 1 also shows a smaller illustrative example with 5 components and a total of 50 residual equations. Fig. 1c shows the Jacobian structure detailing the internal structure of the diagonal and off-diagonal blocks shown in Fig. 1b. The patterns shown in Fig. 1c for the diagonal and off-diagonal blocks are used for the benchmark problems. The component blocks on the diagonal have fully filled in tridiagonal structures and the off-diagonal blocks have tridiagonal structures with every other row empty.

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 run these problems on a desktop computer with a 4 GHz Intel Core i7-4790K processor and 32 GB RAM. Fortran [32] is used to compute the residuals and partial derivatives for the systems of equations. The rest of the computations are performed using the Python-based OpenMDAO framework. Two sets of problems are studied. The first set consists of problems with linear and nonlinear components. The second set consists of problems with only linear components.



(a) A sample randomly generated component-level Jacobian structure for the benchmark problems

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

(c) Internal structure of the Jacobian for the smaller illustrative example in Fig. 1b

Figure 1: Jacobian structures

3.1 Problem set 1: Linear and nonlinear components

Nine subsets of problems are generated by specifying three different values for the coupling amplification factor, α, and three different ranges for numbers of equations per component. Each subset contains 20 problems. The three values used for α are 0.5, 1.0, and 1.5. The three ranges used for the number of equations per component are 100 to 500, 200 to 1000, and 500 to 2000. These ranges are used to randomly assign a number of equations for each component. Each component is also randomly assigned a degree for the polynomial residual equations that belong to it. Random number generators are used for this such that, on average, half of the components are expected to be linear and half are expected to be nonlinear. Of the half that are nonlinear, half are expected to be quadratic and the other half are expected to be cubic. The three subsets of problems that have the same number of equations per component also share the same 20 randomly generated DSMs and other problem settings (except for α, which is varied to study the effect of coupling strength). The initial guess for every variable is randomly set to an integer between -15 and 15, excluding 0, which is the solution. The problems are set to converge when the L2-norm of all the residuals is less than 10^{-6}.

3.2 Problem set 2: Linear components only

The settings for these problems are the same as the settings for problem set 1, except the ranges for the number of equations per component and the degree of the polynomial equations. The three ranges used for the number of equations per component are 200 to 1000, 500 to 2000, and 1000 to 4000.

4 Solution approaches

Four solution approaches are benchmarked for this paper using OpenMDAO release 1.7.3. Two are variations of the CN approach, and two are variations of the NLBGS approach.

4.1 Approach 1 (A1): Monolithic CN

With this approach, a solution to all the residual equations is computed simultaneously using Newton's method. For computing the Newton step, the PETSc [33] Krylov subspace with preconditioning (KSP) flexible generalized minimal residual (fGMRES) [34] solver is used. For preconditioning, the linear block Gauss–Seidel (LNBGS) option is selected in OpenMDAO, which sequentially solves smaller linear systems using each component's Jacobian and SciPy's [35] `linalg.solve` direct solver.

4.2 Approach 2 (A2): CN with strongly connected grouping

For the second approach, the strongly connected components [36, 37] graph partitioning algorithm is used to identify the largest groups of components in which every component directly or indirectly depends on the output of every other component. The NetworkX [38] Python package is used by OpenMDAO for identifying strongly connected component groups. If more than one of these groups are present, they can be reordered relative to each other for feed-forward information flow and each group can be solved sequentially on its own. This improves computation time by reducing the size of the coupled systems of equations that need to be solved together by the Newton solver. Fig. 2 shows a simple example component level Jacobian structure with two strongly connected component groups outlined with dashed lines. For this example, with approach A2, the group containing components 3 to 5 is solved first using a CN solver and then the group containing components 1 and 2 is solved with a CN solver using the converged input from the first group. The group with components 3 to 5 can be individually solved first because it does not directly or indirectly depend on the outputs of components 1 and 2.

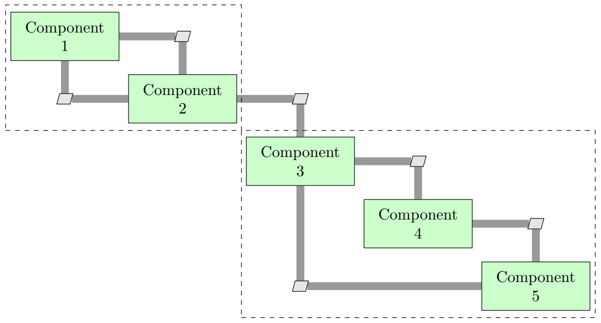


Figure 2: A sample Jacobian structure with two strongly connected component groups

Mathematically, this means that the CN system given by Eq. (9) is repeatedly solved first until convergence. Then the CN system given by Eq. (10), which uses the converged output from component 3 to compute its right hand side, will be repeatedly solved until convergence.

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(3)}}{\partial \boldsymbol{v}^{(3)}} & \frac{\partial \boldsymbol{R}^{(3)}}{\partial \boldsymbol{v}^{(4)}} & 0 \\ 0 & \frac{\partial \boldsymbol{R}^{(4)}}{\partial \boldsymbol{v}^{(4)}} & \frac{\partial \boldsymbol{R}^{(4)}}{\partial \boldsymbol{v}^{(5)}} \\ \frac{\partial \boldsymbol{R}^{(5)}}{\partial \boldsymbol{v}^{(3)}} & 0 & \frac{\partial \boldsymbol{R}^{(5)}}{\partial \boldsymbol{v}^{(5)}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{v}^{(3)} \\ \Delta \boldsymbol{v}^{(4)} \\ \Delta \boldsymbol{v}^{(5)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}^{(3)} \\ \boldsymbol{r}^{(4)} \\ \boldsymbol{r}^{(5)} \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(1)}}{\partial \boldsymbol{v}^{(2)}} \\ \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(1)}} & \frac{\partial \boldsymbol{R}^{(2)}}{\partial \boldsymbol{v}^{(2)}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{v}^{(1)} \\ \Delta \boldsymbol{v}^{(2)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}^{(1)} \\ \boldsymbol{r}^{(2)} \end{bmatrix} \tag{10}$$

Here, $\boldsymbol{R}^{(a)}$ is a vector containing the residual equations for the a^{th} component, $\boldsymbol{v}^{(a)}$ is a vector containing the variables for the a^{th} component, and $\boldsymbol{r}^{(a)}$ is a vector containing the values of the residuals for the a^{th} component. If all components happen to form a single strongly connected component group, this approach is equivalent to A1. Similarly to A1, the PETSc KSP fGMRES solver is used with OpenMDAO's LNBGS option for preconditioning to compute the Newton step for each group. The convergence tolerance for the L2-norm of the residuals of each group is set to

$$\text{atol} = 10^{-6} \sqrt{\frac{\text{number of equations in the group}}{\text{total number of equations in the problem}}} \,. \tag{11}$$

4.3 Approach 3 (A3): NLBGS with strongly connected grouping and Aitken's relaxation

In this approach, similarly to A2, the strongly connected components algorithm is used to identify groups that can be executed in a feed-forward manner and solved individually. The system of each group is solved by cycling through every component belonging to the group. SciPy's `linalg.solve` direct solver is used for solving linear components and the linear Newton systems for nonlinear components. If multiple strongly connected component groups are identified, the solution time reduces due to the lower number of components that need to be iterated through together [39]. OpenMDAO's built-in reordering algorithm is used to determine the execution order of the components within each group and Aitken's acceleration [40] based relaxation is used to help accelerate convergence and prevent divergence. The tolerances for each group are calculated using Eq. (11). If all components happen to form a single strongly connected component group, this approach is equivalent to approach 4.

4.4 Approach 4 (A4): NLBGS with Aitken's relaxation

This approach is based on the standard NLBGS approach where each component is solved sequentially. SciPy's `linalg.solve` direct solver is used for solving linear components and the linear Newton systems for nonlinear components. OpenMDAO's built-in reordering algorithm is used to determine the execution order of the components and Aitken's acceleration based relaxation is used to help accelerate convergence and prevent divergence.

4.5 Reordering

OpenMDAO's inbuilt reordering algorithm aims to help reduce the number of iterations required for an NLBGS approach. Subgraphs of the directed graph representation of the DSM are identified using the strongly connected components algorithm. Then, using a simple heuristic approach [41, 42], the most imbalanced node in terms of input and output edges is identified and the fewest edges are removed so that the node has unidirectional information flow. This process is repeated recursively until a directed acyclic graph is obtained, and finally a topological sort is used to obtain the execution order.

4.6 Aitken's acceleration

As discussed in the introduction, Aitken's acceleration [40] based relaxation methods have been shown to be simple and effective for NLBGS approaches. The procedure used for the benchmark problems in this paper, based on the approach used by Kenway et al. [18], is expressed as

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \theta^{(n)}\Delta\mathbf{u}^{(n)}, \tag{12}$$

where the $\mathbf{u}^{(n+1)}$ is the vector of variables obtained after applying the relaxation procedure, $\mathbf{u}^{(n)}$ is the vector from the previous iteration, θ is the relaxation factor, and the update $\Delta\mathbf{u}^{(n)}$ is given by

$$\Delta\mathbf{u}^{(n)} = \tilde{\mathbf{u}}^{(n+1)} - \mathbf{u}^{(n)}, \tag{13}$$

where $\tilde{\mathbf{u}}^{(n+1)}$ is the vector obtained at the end of the latest NLBGS iteration. The relaxation factor is updated at every iteration using

$$\begin{aligned} \tilde{\theta}^{(n)} &= \theta^{(n-1)}\left(1 - \frac{(\Delta\mathbf{u}^{(n)} - \Delta\mathbf{u}^{(n-1)})^{\mathrm{T}}\Delta\mathbf{u}^{(n)}}{\|\Delta\mathbf{u}^{(n)} - \Delta\mathbf{u}^{(n-1)}\|_2^2}\right), \\ \theta^{(n)} &= \max(0.25, \min(2.0, \tilde{\theta}^{(n)})). \end{aligned} \tag{14}$$

Note that this can only be applied after the second iteration. For the problems in this paper, the relaxation is applied to all the variables of a group at the end of each NLBGS iteration instead of after solving each component. Based on initial trials, we found that this approach is more effective for these problems. Additionally, based on initial trials, we found the range of 0.25 to 2.0 to be an effective range for the relaxation factor for both accelerating convergence and preventing divergence when possible.

5 Results and discussion

Figs. 3 to 5 show box plots for problem set 1. In these plots, the data points are blue if all problems corresponding to that particular approach and subset of problems converged, 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.

Fig. 3 plots the number of iterations for approaches A1 and A4. One iteration of A1 corresponds to one Newton step and one iteration of A4 corresponds to one cycle through all components. Fig. 3, shows that the number of iterations required by A4 is more sensitive to the strength of coupling in the problems than the other approaches. For the weakest coupling amplification factor ($\alpha = 0.5$), both approaches happen to require similar numbers of iterations. However, as the coupling amplification factor is increased, the number of iterations required by A4 increases significantly more than the number of iterations required by A1. For example, the median number of iterations for the problems with 500 to 2000 equations per component and A1 increases from 10 for $\alpha = 0.5$ to 11 for $\alpha = 1.0$. On the other hand, the median number of iterations for the same problems with approach A4 increases from 8.5 for $\alpha = 0.5$ to 20 for $\alpha = 1.0$. Similar trends are observed in other subplots of Fig. 3. Based on Fig. 3 alone, we might conclude that for problems with stronger coupling, a CN approach will be significantly more efficient than an NLBGS approach. However, Fig. 4 leads to a different conclusion.

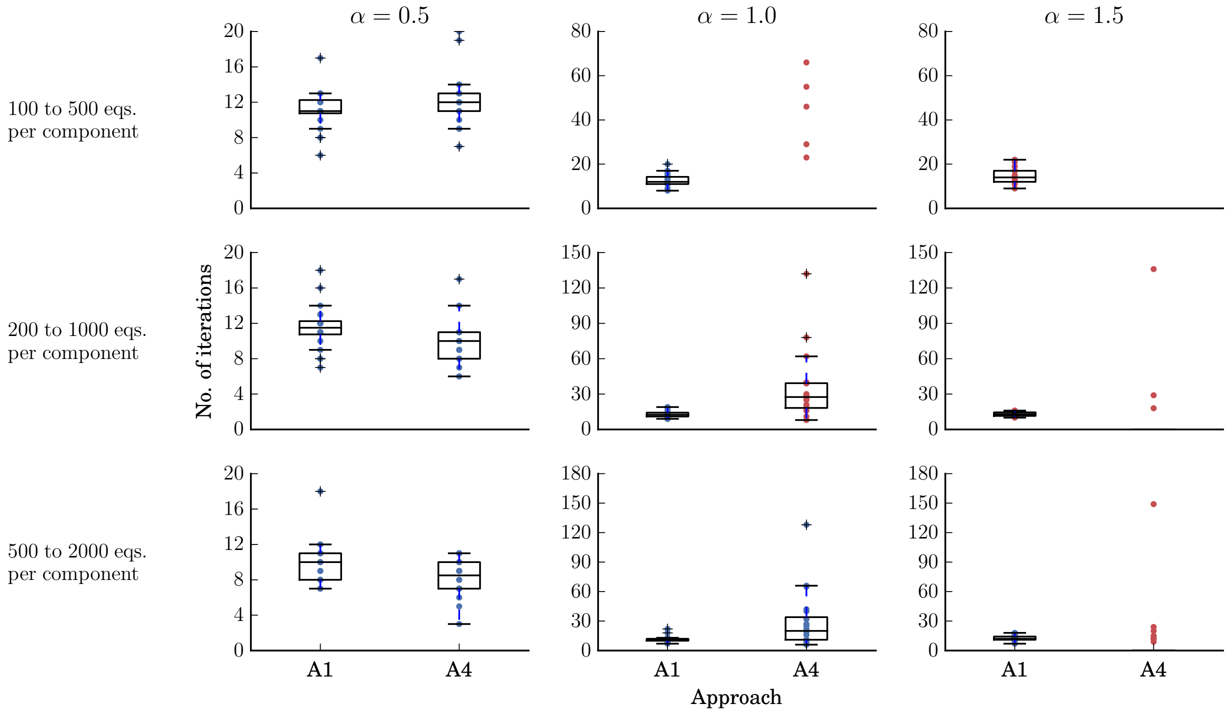


Figure 3: Number of iterations for approaches A1 and A4 with problem set 1. Data points are blue if all problems converged for the combination of approach and problem settings, otherwise they are red. Box and whiskers are not shown if less than 75% of the problems converged for a particular approach and combination of problem settings.

For the CN based approaches (A1 and A2), we use the LNBGS preconditioning option in OpenMDAO with the PETSc fGMRES solver for the Newton systems. This preconditions the larger Newton linear system for each fGMRES iteration by sequentially solving smaller linear systems using each component's Jacobian. For the NLBGS approaches (A3 and A4), each component is solved in sequence. A single linear solution is carried out if the component is linear, or multiple linear solutions are carried out for the component's linearized system by a Newton solver if the component is nonlinear. A direct solver is used for these three types of linear solutions that involve a component's Jacobian or linearized system. Fig. 4 plots the total number of direct linear solutions carried out during the different approaches. A1 and A2 perform significantly larger numbers of direct linear solutions. As the coupling strength is increased, the number of linear solutions carried out by all approaches increases, but they increase by different amounts. For example, the median number of direct linear solutions carried out for the problems with 500 to 2000 equations per component and A1 increases from 1319 for $\alpha = 0.5$ to 2520 for $\alpha = 1.0$. On the other hand, the median for the same problems with A4 increases from 294 for $\alpha = 0.5$ to 677 for $\alpha = 1.0$. Similarly, the median number of direct linear solutions carried out for these same problems increases from 911 to 1568 with A2, and from 240 to 405 with A3. Based on Fig. 4 alone, we might conclude that for these problems, an NLBGS approach will be significantly faster than a CN approach. However, Fig. 5 shows that this is not the case.

Fig. 5 plots the wall time for problem set 1 and Table 2 lists the corresponding mean and median times.

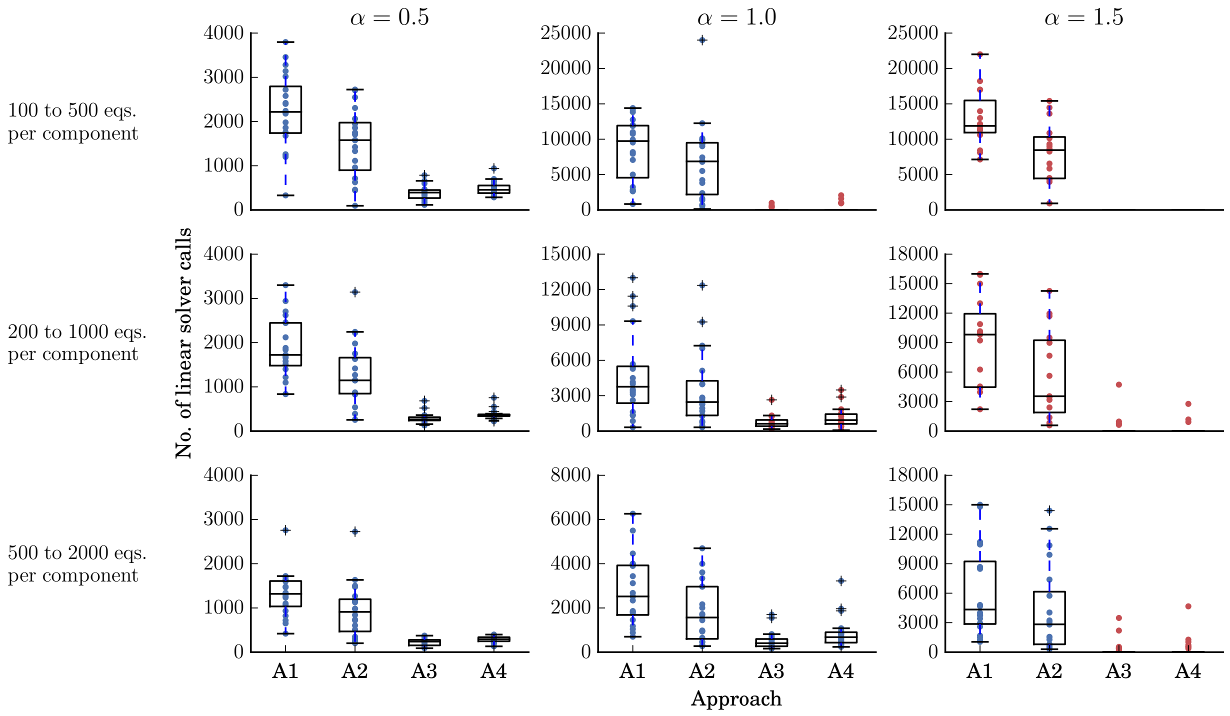


Figure 4: Number of direct linear solutions carried out with the four approaches and problem set 1. Data points are blue if all problems converged for the combination of approach and problem settings, otherwise they are red. Box and whiskers are not shown if less than 75% of the problems converged for a particular approach and combination of problem settings.

A2 consistently performs better than A1, and A3 consistently performs better than A4. This demonstrates the benefits of grouping based on the strongly connected components algorithm. However, it is difficult to conclude which one of A2 and A3 is faster in general. Two major costs contribute to the observed performance of these approaches. One is the cost of assembling the linear systems involved, and the other is the cost of solving these linear systems. NLBGS approaches suffer from the disadvantage of requiring increasingly larger numbers of iterations as the coupling strength increases. Along with increasing the number of linear solutions carried out, this also increases the number of times the linearized systems are assembled for the components. This may be inexpensive for linear components, which only require recomputing the right hand side. However, this may become expensive for nonlinear components that require recomputing Jacobians for each iteration. Since the required number of iterations for NLBGS approaches is observed to be more sensitive to coupling strength, their cost, as coupling strength increases, is also more sensitive to the cost of recomputing the terms of the linear systems involved. On the other hand, while the number of Newton iterations for a CN approach does not depend strongly on the coupling strength, CN based approaches have the disadvantage of assembling and solving larger linear systems that tend to have poorer conditioning as the coupling strength increases. This requires more iterations for iterative solvers, which are generally the only practical option for large systems due to the poor scaling of direct solvers with problem size. Therefore, we conclude that the cost of all the approaches studied in this paper is related to the coupling strength of the problems.

Table 3 lists the mean and median wall times for problem set 2 which contains only problems with linear components. These results are included to study the effect of nonlinearity on the different approaches. Once again, it is not clear which of A2 and A3 will be faster in general. For this problem set, the NLBGS approaches benefit from the lower cost of setting up the linear systems for the linear components. However, the cost for the CN approaches also decrease because, neglecting numerical errors, only one Newton iteration is required to converge the coupled linear system. Overall, we conclude that the relative performance of the

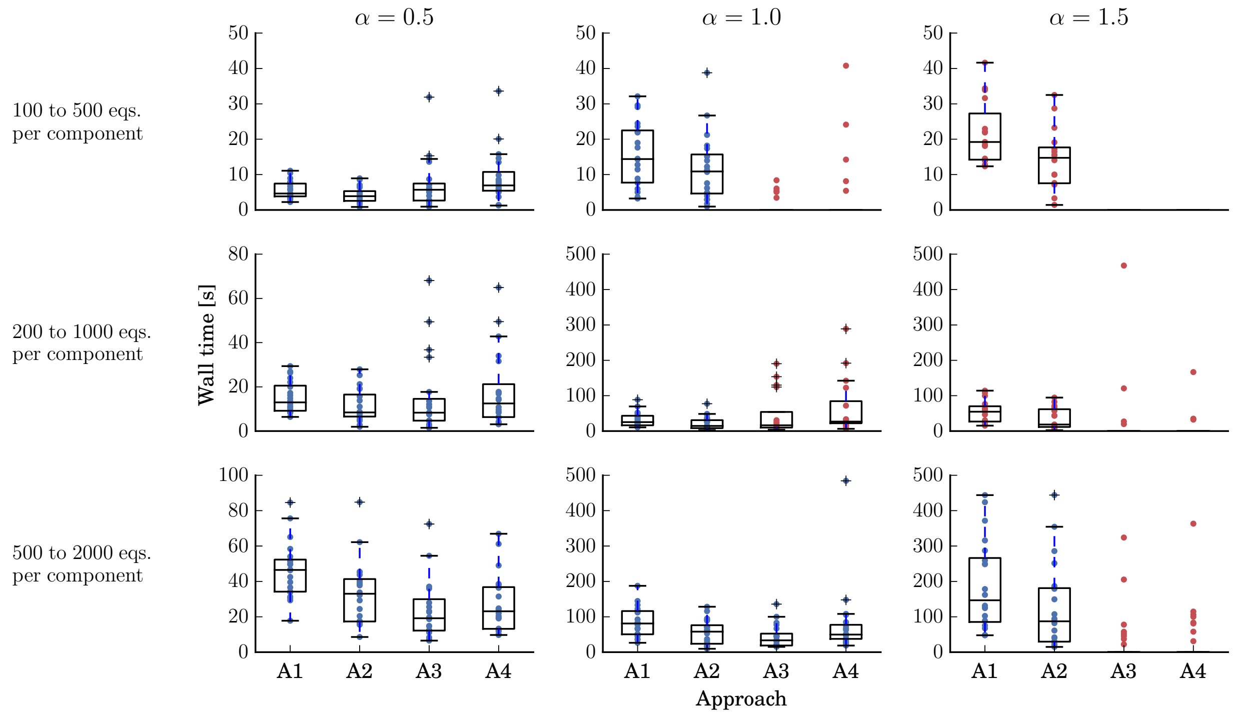


Figure 5: Solution wall time for the four approaches with problem set 1. Data points are blue if all problems converged for the combination of approach and problem settings, otherwise they are red. Box and whiskers are not shown if less than 75% of the problems converged for a particular approach and combination of problem settings.

different approaches depends on multiple factors including the strength of coupling in the problem, the cost of assembling the linearized systems involved, and the performance of the selected linear solvers. Depending on different combinations of these factors, which are dependent on the problem itself, the solver implementation, and the solver settings, one approach may be faster than the other.

Even though it is difficult to conclude which of A2 and A3 will be faster in general, it is clear that A2 converges for a greater number of the benchmark problems than A3. Table 4 lists the percentages of problems that converged for both problem sets. As the coupling amplification factor is increased for the problems, even with the under-relaxation provided by Aitken's acceleration, the NLBGS approaches tend to become unstable and diverge. Note that some problems did not converge with the CN approaches as well for $\alpha = 1.5$. However, unlike most problems that did not converge with the NLBGS approaches, these problems did not diverge. The Newton solvers could not converge to the specified tolerance due to numerical errors arising from poor conditioning.

We can also analyze how the performance of different approaches scales with coupling strength. Fig. 6a plots the wall times for the 12 problems from problem set 1 that converged for all three values of α used. By observing the changing slopes of the curves, we see that the wall times scale better for A2 as the coupling strength increases.

Since α is a parameter specific to the problem formulation presented in this paper, we introduce a more general coupling strength metric, ρ_c, which we use in Fig. 6b. This is defined as

$$\rho_c = \text{Spectral radius}\left(\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{lower}}^{-1}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{upper}}\right), \tag{15}$$

where $\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{lower}}$ is the block lower triangle of the Jacobian (including the diagonal blocks) and $\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{v}}\right]_{\text{upper}}$ is the block upper triangle of the Jacobian (excluding the diagonal blocks). For Fig. 6b, this parameter is

Table 2: Wall time statistics (in seconds) for problem set 2 (with linear and nonlinear components). Values are omitted if less than 75% of problems converged, and the lowest values are highlighted with bold font.

		$\alpha = 0.5$				$\alpha = 1.0$				$\alpha = 1.5$			
		A1	A2	A3	A4	A1	A2	A3	A4	A1	A2	A3	A4
100 to 500 eqs.	mean	5.62	**4.26**	7.34	9.29	15.73	**11.64**	-	-	22.03	**14.32**	-	-
per component	median	4.65	**3.90**	5.71	6.94	14.39	**10.89**	-	-	19.22	**14.74**	-	-
200 to 1000 eqs.	mean	15.24	**11.51**	15.37	18.41	32.19	**21.97**	47.96	66.09	55.07	**36.72**	-	-
per component	median	12.98	8.43	**8.32**	12.49	25.33	**14.54**	16.36	26.82	54.71	**18.48**	-	-
500 to 2000 eqs.	mean	46.47	33.41	**23.69**	27.30	86.26	55.70	**44.30**	79.89	190.00	**124.98**	-	-
per component	median	46.49	33.02	**19.18**	23.11	81.06	58.00	**33.45**	49.72	146.71	**87.24**	-	-

Table 3: Wall time statistics (in seconds) for problem set 2 (with linear components only). Values are omitted if less than 75% of problems converged, and the lowest values are highlighted in bold font.

		$\alpha = 0.5$				$\alpha = 1.0$				$\alpha = 1.5$			
		A1	A2	A3	A4	A1	A2	A3	A4	A1	A2	A3	A4
200 to 1000 eqs.	mean	0.91	0.61	**0.50**	0.80	2.11	**1.30**	1.83	3.84	20.48	**6.51**	-	-
per component	median	0.92	0.59	**0.50**	0.84	1.68	1.07	**1.04**	2.84	4.64	**2.56**	-	-
500 to 2000 eqs.	mean	4.24	2.54	**2.38**	3.89	7.26	**4.69**	5.37	10.35	21.53	**10.10**	-	-
per component	median	4.20	2.36	**2.30**	3.73	7.10	**3.50**	3.62	8.04	11.90	**5.47**	-	-
1000 to 4000 eqs.	mean	19.39	13.47	**13.37**	20.21	37.09	**23.39**	24.87	36.02	52.01	**29.44**	-	-
per component	median	20.77	14.21	**12.74**	20.58	25.24	**14.33**	18.85	27.49	40.76	**20.94**	-	-

Table 4: Percentage of problems that converged with varying coupling amplification factor

	$\alpha = 0.5$				$\alpha = 1.0$				$\alpha = 1.5$			
Problem Set	A1	A2	A3	A4	A1	A2	A3	A4	A1	A2	A3	A4
1	100%	100%	100%	100%	100%	100%	68%	68%	85%	83%	20%	18%
2	100%	100%	100%	100%	100%	100%	95%	95%	100%	100%	38%	38%

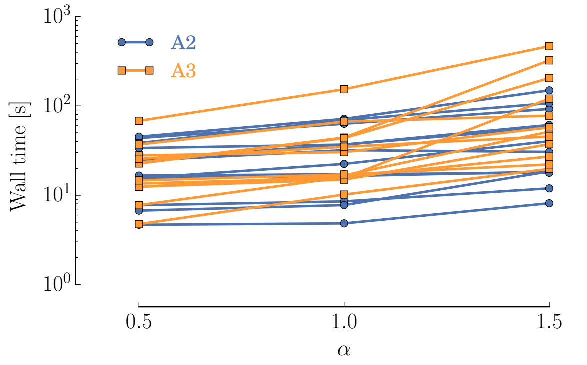
calculated for each strongly connected component group in a problem and the largest value is selected to represent the problem. For problem set 1, the mean and median ρ_c are 1.17 and 0.87, respectively. The mean and median ρ_c for the problems in this set that did not converge with A3 are 2.28 and 1.89, respectively. Fig. 6b shows that visibly fewer problems converged with A3 for ρ_c greater than 1.0 (12 for A3 vs. 69 for A2). Similarly, for problem set 2, the mean and median ρ_c are 0.69 and 0.43, respectively. The mean and median ρ_c for the problems in this set that did not converge with A3 are 1.74 and 1.73, respectively. Note that, for a nonlinear component, the Jacobian changes with every iteration. The values of ρ_c used in Fig. 6b correspond to the initial Jacobian.

6 Conclusion

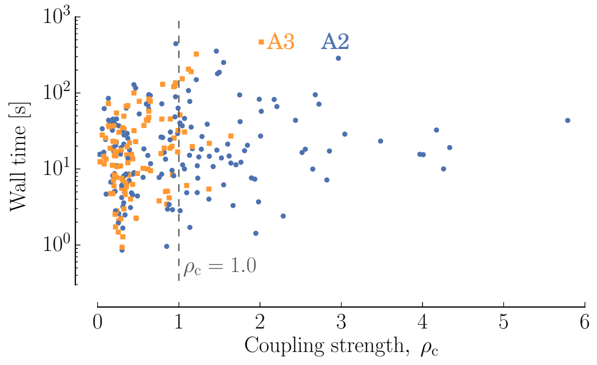
In this paper, we presented a novel Taylor series-based scalable problem, implemented it in the OpenMDAO framework, and used it to compare approaches for the multidisciplinary analysis of complex systems. The scalable problem is a Taylor series-based system of equations, with several tunable parameters including the total number of variables, degree of nonlinearity, coupling strength, and sparsity structure.

This problem is implemented in NASA's OpenMDAO framework, an open-source Python-based software package for multidisciplinary optimization. OpenMDAO also provides a library of integrated nonlinear and linear solvers that simplifies experimenting with and switching between multiple solvers.

We used the OpenMDAO implementation of the scalable problem to compare coupled Newton and nonlinear block Gauss–Seidel approaches for the analysis of complex systems. The four approaches compared are: a monolithic coupled Newton approach (A1), a coupled Newton approach with grouping based on the



(a) Wall times for the 12 problems from problem set 1 that converge for all three coupling amplification factors



(b) Scatter plot with wall times for problem set 1 with respect to coupling strength, ρ_c

Figure 6: Wall times for problem set 1 showing the behaviors of A2 and A3 with increasing coupling strength

strongly connected components algorithm (A2), a nonlinear block Gauss–Seidel approach with grouping based on the strongly connected components algorithm and relaxation based on Aitken's acceleration (A3), and a nonlinear block Gauss–Seidel approach with relaxation based on Aitken's acceleration (A4).

The two approaches with grouping based on the strongly connected components algorithm consistently outperform the other two approaches when multiple strongly connected component groups can be identified. However, it is not possible to reach a general conclusion on which single approach will be faster. The relative performances depend on several factors: linear system assembly time, linear solver efficiency, strength of coupling in the problem, number of variables per component, number of components, degree of nonlinearity, and the overall nonlinear solver tolerance. We found that the first three factors play a major role in determining which approach is more efficient for a given problem. We present plots showing how these three factors affect the trade-off and also introduce a general metric for the strength of coupling between components. Based on the 360 randomly generated problems, we also observed that the coupled Newton approaches are more robust and less likely to diverge than the nonlinear block Gauss–Seidel approaches as the strength of coupling between components increases. Additionally, the coupled Newton approach is observed to scale better as coupling strength increases.

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.

References

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

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

[3] K. Maute, M. Nikbay, and C. Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA Journal*, 39(11):2051–2061, November 2001. ISSN 0001-1452. doi:10.2514/2.1227. URL `http://dx.doi.org/10.2514/2.1227`.

[4] Ulrich Küttler and Wolfgang A. Wall. Fixed-point fluid-structure interaction solvers with dynamic

relaxation. *Computational Mechanics*, 43(1):61–72, 2008. ISSN 1432-0924. doi:10.1007/s00466-008-0255-5. URL http://dx.doi.org/10.1007/s00466-008-0255-5.

[5] Matthias Heil, Andrew L. Hazel, and Jonathan Boyle. Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches. *Computational Mechanics*, 43(1):91–101, 2008. ISSN 1432-0924. doi:10.1007/s00466-008-0270-6. URL http://dx.doi.org/10.1007/s00466-008-0270-6.

[6] Melike Nikbay, Levent Öncü, and Ahmet Aysan. Multidisciplinary code coupling for analysis and optimization of aeroelastic systems. *Journal of Aircraft*, 46(6):1938–1944, November 2009. ISSN 0021-8669. doi:10.2514/1.41491. URL http://dx.doi.org/10.2514/1.41491.

[7] M. M. Joosten, W. G. Dettmer, and D. Perić. 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, 2009. ISSN 1097-0207. doi:10.1002/nme.2503. URL http://dx.doi.org/10.1002/nme.2503.

[8] D.E. Keyes, L.C. McInnes, C. Woodward, W.D. Gropp, E. Myra, and M. Pernice. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 10/2012 2012. URL http://hpc.sagepub.com/content/27/1/4.full.pdf+html.

[9] John T. Hwang, Dae Young Lee, James W. Cutler, and Joaquim R. R. A. Martins. Large-scale multidisciplinary optimization of a small satellite's design and operation. *Journal of Spacecraft and Rockets*, 51(5):1648–1663, April 2014. ISSN 0022-4650. doi:10.2514/1.A32751. URL http://dx.doi.org/10.2514/1.A32751.

[10] Justin Gray, Tristan Hearn, Kenneth Moore, John T. Hwang, Joaquim R. R. A. Martins, and Andrew Ning. 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, June 2014. doi:10.2514/6.2014-2042. URL http://dx.doi.org/10.2514/6.2014-2042.

[11] E. Arian. Convergence estimates for multidisciplinary analysis and optimization. Technical Report NASA/CR-97-201752, NAS 1.26:201752, ICASE-97-57, Institute for Computer Applications in Science and Engineering; Hampton, VA United States, 1997.

[12] Matthias Heil. 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, 2004. ISSN 0045-7825. doi:10.1016/j.cma.2003.09.006. URL http://dx.doi.org/10.1016/j.cma.2003.09.006.

[13] Manuel Barcelos, Henri Bavestrello, and Kurt Maute. 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, 2006. doi:10.1016/j.cma.2004.09.013. URL http://dx.doi.org/10.1016/j.cma.2004.09.013.

[14] Miguel Ángel Fernández and Marwan Moubachir. A Newton method using exact Jacobians for solving fluid-structure coupling. *Computers & Structures*, 83(23):127 – 142, 2005. ISSN 0045-7949. doi:10.1016/j.compstruc.2004.04.021. URL http://dx.doi.org/10.1016/j.compstruc.2004.04.021.

[15] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. R. Hughes. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38(4):310–322, 2006. ISSN 1432-0924. doi:10.1007/s00466-006-0084-3. URL http://dx.doi.org/10.1007/s00466-006-0084-3.

[16] Jason P. Sheldon, Scott T. Miller, and Jonathan S. Pitt. Methodology for comparing coupling algorithms for fluid-structure interaction problems. *World Journal of Mechanics*, 4(2), 2014. doi:10.4236/wjm.2014.42007. URL http://dx.doi.org/10.4236/wjm.2014.42007.

[17] Stefan Turek and Jaroslav Hron. *Proposal for Numerical Benchmarking of Fluid-Structure Interaction between an Elastic Object and Laminar Incompressible Flow.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-34596-1. doi:10.1007/3-540-34596-5_15. URL http://dx.doi.org/10.1007/3-540-34596-5_15.

[18] Gaetan K. W. Kenway, Graeme J. Kennedy, and Joaquim R. R. A. Martins. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. *AIAA Journal*, 52(5):935–951, March 2014. ISSN 0001-1452. doi:10.2514/1.J052255. URL http://dx.doi.org/10.2514/1.J052255.

[19] John T. Hwang and Joaquim R. R. A. Martins. A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives. *ACM Transactions on Mathematical Software*, 2017. (Accepted subject to revisions).

[20] S. Padula, N. Alexandrov, and L. Green. MDO test suite at NASA langley research center. In *6th Symposium on Multidisciplinary Analysis and Optimization*, Multidisciplinary Analysis Optimization Conferences. American Institute of Aeronautics and Astronautics, September 1996. doi:10.2514/6.1996-4028. URL http://dx.doi.org/10.2514/6.1996-4028.

[21] R. Balling and C. Wilkinson. Execution of multidisciplinary design optimization approaches on common test problems. *AIAA Journal*, 35(1):178–186, January 1997. ISSN 0001-1452. doi:10.2514/2.7431. URL http://dx.doi.org/10.2514/2.7431.

[22] S. Kodiyalam and C. Yuan. Evaluation of methods for multidisciplinary design optimization, phase I. Technical report, National Aeronautics and Space Administration, 1998.

[23] S. I. Yi, J. K. Shin, and G. J. Park. Comparison of mdo methods with mathematical examples. *Structural and Multidisciplinary Optimization*, 35(5):391–402, May 2008. ISSN 1615-1488. doi:10.1007/s00158-007-0150-2. URL https://doi.org/10.1007/s00158-007-0150-2.

[24] S. Tosserams, L. F. P. Etman, and J. E. Rooda. A micro-accelerometer MDO benchmark problem. *Structural and Multidisciplinary Optimization*, 41(2):255–275, 2010. ISSN 1615147X. doi:10.1007/s00158-009-0422-0. URL http://dx.doi.org/10.1007/s00158-009-0422-0.

[25] Nathan P. Tedford and Joaquim R. R. A. Martins. Benchmarking multidisciplinary design optimization algorithms. *Optimization and Engineering*, 11(1):159–183, 2010. ISSN 13894420. doi:10.1007/s11081-009-9082-6. URL http://dx.doi.org/10.1007/s11081-009-9082-6.

[26] John T. Hwang and Joaquim R. R. A. Martins. 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, January 2016. doi:10.2514/6.2016-1662. URL http://dx.doi.org/10.2514/6.2016-1662.

[27] T. Mosher. Conceptual spacecraft design using a genetic algorithm trade selection process. *Journal of Aircraft*, 36(1):200–208, January 1999. ISSN 0021-8669. doi:10.2514/2.2426. URL http://dx.doi.org/10.2514/2.2426.

[28] Xingzhi Hu, Xiaoqian Chen, Valerio Lattarulo, and Geoffrey T. Parks. Multidisciplinary optimization under high-dimensional uncertainty for small satellite system design. *AIAA Journal*, 54(5):1732–1741, February 2016. ISSN 0001-1452. doi:10.2514/1.J054627. URL http://dx.doi.org/10.2514/1.J054627.

[29] Andrew Ning and Derek Petch. Integrated design of downwind land-based wind turbines using analytic gradients. *Wind Energy*, 19(12):2137–2152, 2016. ISSN 1099-1824. doi:10.1002/we.1972. URL http://dx.doi.org/10.1002/we.1972. we.1972.

[30] Donald V. Steward. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28(3):71–74, 1981. doi:10.1109/TEM.1981.6448589. URL https://doi.org/10.1109/TEM.1981.6448589.

[31] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46(2):273–284, 2012. ISSN 1615-1488. doi:10.1007/s00158-012-0763-y. URL `http://dx.doi.org/10.1007/s00158-012-0763-y`.

[32] P. Peterson. F2PY: A tool for connecting Fortran and Python programs. *International Journal of Computational Science and Engineering*, 4(4):296–305, 2009.

[33] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*, pages 163–202. Birkhäuser Press, 1997.

[34] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, mar 1993. doi:10.1137/0914028. URL `https://doi.org/10.1137%2F0914028`.

[35] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`. [Online].

[36] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2): 146–160, jun 1972. doi:10.1137/0201010. URL `https://doi.org/10.1137%2F0201010`.

[37] Esko Nuutila and Eljas Soisalon-Soininen. On finding the strongly connected components in a directed graph. *Information Processing Letters*, 49(1):9 – 14, 1994. ISSN 0020-0190. doi:10.1016/0020-0190(94)90047-7. URL `http://dx.doi.org/10.1016/0020-0190(94)90047-7`.

[38] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[39] A. S. Shaja and K. Sudhakar. Optimized sequencing of analysis components in multidisciplinary systems. *Research in Engineering Design*, 21(3):173–187, 2010. ISSN 1435-6066. doi:10.1007/s00163-009-0082-5. URL `http://dx.doi.org/10.1007/s00163-009-0082-5`.

[40] Bruce M. Irons and Robert C. Tuck. A version of the aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1(3):275–277, 1969. ISSN 1097-0207. doi:10.1002/nme.1620010306. URL `http://dx.doi.org/10.1002/nme.1620010306`.

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

[42] A. Baharev, H. Schichl, A. Neumaier, and T. Achterberg. An exact method for the minimum feedback arc set problem. *University of Vienna*, 2015.