An adaptive, inexact gradient-based algorithm for multidisciplinary design optimization

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

Large-scale multidisciplinary design optimization problems often involve thousands of design variables and tens of thousands of state variables. If formulated using a simultaneous analysis and design architecture, these optimization problems would have tens of thousands of optimization variables and tens of thousands of equality constraints. Such problems can be particularly difficult to solve even with a gradient-based approach. This paper presents an adaptive, inexact quasi-Newton algorithm for solving large-scale multidisciplinary design optimization problems using a simultaneous analysis and design architecture. This algorithm is novel in two ways. First, this algorithm uses new inexactness criteria that are easy to compute and can be used with any Krylov solver with or without a preconditioner. Second, this algorithm adaptively chooses the stopping criteria for the Krylov solver that makes the best use of its performance. This algorithm is applied to an equality-constrained cantilever bar problem with up to 3,000 design variables. We observe that this algorithm is robust and that it yields a reduction greater than 50% in terms of the total Krylov iterations, compared with the exact quasi-Newton method.

I. Introduction

Multidisciplinary design optimization (MDO) uses numerical optimization methods to solve complex engineering design problems that involve coupled numerical models. A typical MDO problem is the conceptual design of an aircraft [1–3]. For an aircraft design problem, disciplines such as aerodynamics, structures, and controls are tightly coupled. Optimizing the design of an aircraft as a whole requires an integrated optimization framework that considers how the different disciplines interact with each other. MDO has also been applied to other complex engineering design problems for electric aircraft [4], eVTOL aircraft [5], wind turbines [6, 7], launch vehicles [8], satellites [9] and automobiles [10, 11].

For a complex engineering design problem like the aircraft design, the optimal design is only insightful when the accuracy of the engineering model is sufficient. For disciplines like structures and aerodynamics, sufficient accuracy can only be achieved by using high-fidelity FEM and CFD solvers, which require a significant number of design variables and state variables in the MDO problem. For a large-scale MDO problem with thousands of design variables, the numerical optimization is often carried out using a gradient-based approach. Gradient-based algorithms scale better than the gradient-free algorithms; however, they require an efficient and accurate approach to calculate the derivatives. NASA's OpenMDAO software framework [12] enables a gradient-based approach using analytic derivatives to solve for large-scale MDO problems. In OpenMDAO, the analytic derivatives are calculated by using the unified derivative equation [13], which unifies the chain rule, adjoint method, and other derivative computation methods.

Simultaneous analysis and design (SAND) is a widely used architecture to formulate an MDO problem. SAND solves the residual equations and the optimization problem at the same time by treating the state variables as optimization variables and the residual equations as equality constraints. The SAND architecture is effective for small-size problems, but its performance degrades when the problem has large number of design variables and much larger number of state variables. In SAND, a large-scale MDO problem with, e.g., 1,000 design variables and 10,000 state variables, is formulated as an optimization problem with 11,000 optimization variables and 10,000 equality constraints. Using a Newton-type optimization algorithm could involve solving a $21{,}000 \times 21{,}000$ Karush–Kuhn–Tucker (KKT) system, which takes significant computing time to solve to a tight tolerance. One way to accelerate the computation is to solve to a much looser tolerance in intermediate optimization iterations, i.e., apply an inexact Newton approach.

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

In current inexact Newton methods, an inexact tolerance is computed, and the Krylov solver for the KKT system terminates at the first iteration in which the tolerance is satisfied. We see two problems for the current inexact Newton methods. First, the majority of the inexact methods use a forcing parameter criterion to compute the inexact tolerance; however, they do not guarantee a descent direction at each iteration. For those methods that guarantee a descent direction (e.g., the inexact Lagrange-Newton-Krylov-Schur (LNKS) method [14]), the inexactness criteria used are only applicable to a specific preconditioner. Second, the Krylov solver stops as soon as the inexactness tolerance is met, regardless of the convergence rate of the Krylov solver. However, in many cases, running the Krylov solver for a small number of additional iterations would result in a significantly more accurate solution.

We propose an adaptive, inexact quasi-Newton algorithm for large-scale equality-constrained MDO using the SAND architecture. There are two key features of this algorithm. First, this algorithm uses the new criterion we derived to compute the inexact tolerance. This tolerance assures a descent direction on the augmented Lagrangian merit function at each iteration. Additionally, the new criteria are easy-to-compute and applicable to any Krylov solver with or without a preconditioner. Second, in this algorithm, we adaptively select the stopping criteria for the Krylov solver based on its convergence rate to capitalize whenever convergence rate is high. This general approach is also applicable to interior point methods for large-scale inequality-constrained MDO.

This paper proceeds as follows. In Sec. II, we provide some background on MDO architecture, existing quasi-Newton methods, and current inexact quasi-Newton methods. In Sec. III, we present our proposed algorithm. In Sec. IV, we demonstrate the effectiveness of our algorithm on a cantilever bar optimization problem.

II. Background

A. MDO architecture

In the MDO field, the term *MDO architecture* is used to refer to the method through which the multidisciplinary coupling is addressed and how the optimization problem is solved [15]. The two most widely used MDO architectures are multidisciplinary feasible (MDF) [16] and simultaneous analysis and design (SAND) [17]. MDF and SAND are equivalent to the reduced-space method and the full-space method from PDE-constrained optimization.

In the reduced-space method, only design variables x are treated as optimization variables, and the state variables y are computed by $\mathcal{Y}(x)$, which is an explicit computation of the discipline states. The reduced-space optimization problem (considering only equality constraints, without loss of generality) is given by

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

where $x \in \mathbb{R}^n$ are the optimization variables, $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $C : \mathbb{R}^n \to \mathbb{R}^m$ is the vector-valued constraint function, and $\mathcal{R} : \mathbb{R}^k \to \mathbb{R}^k$ is the vector-valued residual function.

In the full-space method, both x and y are treated as optimization variables. Instead of evaluating $\mathcal{Y}(x)$, the discipline residual equations ($\mathcal{R}(x, y) = 0$) are treated as equality constraints of the optimization problem. The full-space optimization problem is

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

Comparing these two methods, the full-space method results in a higher-dimensional optimization problem, as it has both x and y as optimization variables. Thus, it takes more iterations for the optimization problem to converge than in the reduced-space method. In contrast, the reduced-space method requires more computation in each iteration as y must be solved via the interdisciplinary equality constraints.

Another architecture of interest is the strong unification of reduced space and full space (SURF) method proposed by Joshy and Hwang [18, 19]. The SURF method is based on a full-space architecture, and provides a hybrid version of reduced-space and full-space methods. Using a full-space or SURF architecture to solve a large-scale MDO problem results in a high-dimensional equality-constrained optimization problem, that necessitates a gradient-based method and provides the motivation for the inexact approaches investigated here.

B. Existing quasi-Newton methods

We now consider a general equality-constrained optimization problem (without distinguishing between full-space and reduced-space formulations):

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x) \\ \text{s.t.} \quad & C(x) = 0. \end{aligned} \tag{3}$$

The Lagrangian function is defined as

$$\mathcal{L}(x, \lambda) := \mathcal{F}(x) + \lambda^T C(x), \tag{4}$$

where $\lambda \in \mathbb{R}^m$ is vector of Lagrange multipliers. For conciseness, we introduce the following nomenclature:

$$\begin{aligned} g(x) &:= \partial_x \mathcal{F}(x) \\ N(x) &:= \partial_x C(x) \\ M(x, \lambda) &:= \partial_{xx} \mathcal{L}(x, \lambda). \end{aligned}$$

The first-order optimality conditions state that at a local minimum, the gradients of the Lagrangian function are equal to zero; that is,

$$\begin{bmatrix} \partial_x \mathcal{L}(x, \lambda) \\ \partial_\lambda \mathcal{L}(x, \lambda) \end{bmatrix} = \begin{bmatrix} g(x) + N(x)^T \lambda \\ C(x) \end{bmatrix} = 0 \quad (\text{or } b = 0). \tag{5}$$

At each iteration in Newton methods, we first compute the search direction for the optimization variables and Lagrange multipliers by solving the Karush–Kuhn–Tucker (KKT) system, which is given by

$$\begin{bmatrix} M(x, \lambda) & N(x)^T \\ N(x) & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \end{bmatrix} = - \begin{bmatrix} g(x) + N(x)^T \lambda \\ C(x) \end{bmatrix} \quad (\text{or } Ap = -b), \tag{6}$$

where p_x and p_λ are the search directions for the design variables and the Lagrange multipliers, respectively. The matrix A is called the KKT matrix. Solving (6) requires us to evaluate a Hessian matrix $M(x, \lambda)$, which is expensive. Typically, the model only provides accurate first-order derivatives; therefore, we use quasi-Newton approaches that approximate the Hessian matrix using recursive updates, e.g., the Broyden–Fletcher–Goldfarb–Shanno (BFGS) formula. These updates typically ensure hereditary positive definiteness of the Hessian matrix.

Once the search directions are obtained, we consider a line search method to find an acceptable step to update x and λ. This strategy globalizes the quasi-Newton method, meaning it will converge to a local minimum from any starting point. Another commonly used globalization strategy is the trust region method. For interested readers, summaries of trust region methods used for equality constrained optimization can be found in the literature [20–22].

The line search algorithm must achieve a sufficient decrease in the augmented Lagrangian merit function, given by

$$\phi(x, \lambda) := \mathcal{F}(x) + \lambda^T C(x) + \frac{\rho}{2} C(x)^T C(x), \tag{7}$$

where $\rho \in \mathbb{R}$ is a non-negative penalty parameter. Choosing the penalty parameter is crucial for the line search algorithm. Furthermore, the merit function is only exact when the penalty parameter is large enough, meaning the minimum for the merit function is also the minimum for the Lagrangian function. However, having a large penalty parameter would affect the convergence rate of the quasi-Newton algorithm, especially at the early iterations when the current point is not close to the minimum point. Gill et al. [23] suggest to keep it as small as possible and only increase it to assure the global convergence conditions.

We bound the penalty parameter in the same way as in the Lagrange-Newton-Krylov-Schur (LNKS) method [14], to ensure a descent direction. A search direction p is a descent direction if

$$\nabla \phi^T p < 0. \tag{8}$$

For the augmented-Lagrangian merit function, we have

$$\begin{aligned} \nabla \phi^T p &= (g + N^T \lambda + \rho N^T c)^T p_x + N^T p_\lambda \\ &= -g^T M g - \rho c^T c + c^T p_\lambda. \end{aligned} \tag{9}$$

Since M is calculated as a positive-definite matrix in quasi-Newton methods. A descent direction is ensured if the penalty parameter ρ satisfies

$$\rho > \frac{c^T p_\lambda}{c^T c}. \tag{10}$$

We consider a simple backtracking Armijo line search method to find the update step α^k, in which $\alpha^k \in (0, 1]$ is chosen to satisfy the Armijo condition, given by

$$\phi(x^k + \alpha p_x^k, \lambda^k + \alpha p_\lambda^k) \leq \phi(x^k, \lambda^k) + \alpha^k \eta \nabla \phi(x^k, \lambda^k)^T p^k. \tag{11}$$

Then, the optimization variables and the Lagrange multipliers are updated as

$$\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha \begin{bmatrix} p_x^k \\ p_\lambda^k \end{bmatrix}. \tag{12}$$

We refer to this line-search based quasi-Newton algorithm as the exact quasi-Newton method, an outline of this algorithm is shown in Alg. 1.

Algorithm 1 Exact quasi-Newton method

1: **loop**
2: Evaluate c, g, N at x^k
3: Assemble A^k and b^k
4: Solve $A^k p^k = b^k$
5: Update ρ to ensure a descent direction
6: Find α^k via a backtracking line search method
7: Update $\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha^k \begin{bmatrix} p_x^k \\ p_\lambda^k \end{bmatrix}$
8: Update M via the BFGS formula

C. Current Inexact Quasi-Newton Methods

The quasi-Newton method has been successfully applied in various PDE-constrained optimization problems and MDO problems. The main challenge of applying the quasi-Newton method to a very large-scale problem is to find an efficient way to solve the linear KKT system in (6). The KKT matrix contains a mixture of first-order and second-order gradient information. Even when the Hessian matrix is positive definite, the larger KKT matrix is indefinite and is often ill-conditioned, which makes solving the KKT system difficult.

When the KKT system is large in size (say, 10,000 × 10,000 or larger), Krylov iterative solvers are preferred over direct solvers. Widely used Krylov methods include the classic conjugate gradient (CG) method, biconjugate gradient stabilized (BiCGStab) method [24] and generalized minimal residual (GMRES) method [25]. Note that the CG method requires the KKT matrix to be positive definite. This can be satisfied by adding a small number to the diagonal elements. In many cases, a preconditioner is used to increase the convergence rate of the Krylov solver. With a preconditioner, the KKT system becomes

$$P^{-1} A p = P^{-1} b, \tag{13}$$

where P is a preconditioner.

The effectiveness of a preconditioner is usually case-dependent. For instance, the active-set sequential quadratic programming (SQP) algorithm, SNOPT [26], uses a CG method without a preconditioner. In contrast, LNKS [14] uses a preconditioner equivalent to the block LU factorization. Other possible preconditioners include block Jacobi, incomplete LU, and incomplete Cholesky.

Even with a preconditioner, a Krylov solver can still take tens or hundreds of iterations to solve the large-scale KKT system. One way to accelerate it is to solve the KKT system inexactly, which may take significantly fewer iterations. We define the residual vectors r as

$$r = Ap - b. \tag{14}$$

When we solve the KKT system exactly, we stop the Krylov solver when the norm of the residuals becomes small, e.g. $\|r\| < 1e-14$. When we solve it inexactly, the tolerance is less tight, e.g. $\|r\| < 1e-6$, which may save tens of iterations of the Krylov solver. The most commonly used inexactness criterion is

$$\|r\| \leq \eta \|r\|, \tag{15}$$

where $\eta \in [0, 1)$ is the forcing parameter that is chosen differently from iteration to iteration. This criterion has been used in many full-space Lagrange-Newton algorithms for solving PDE-constrained optimization problems [27, 28]. However, this criterion does not guarantee that the search direction calculated is a descent direction for the augmented Lagrangian merit function, and the performance of this criterion differs from case to case. The inexactness criteria used in the inexact LNKS method [14] ensure a descent direction on the augmented Lagrangian merit function. However, it assumes a particular LU-equivalent preconditioner and these criteria are not easy to compute.

III. Methodology

In our proposed method, we follow the line-search-based quasi-Newton algorithm outlined in Alg. 1, and the KKT system is solved inexactly. New inexactness criteria are used to compute the inexact tolerances in our new algorithm that adaptively chooses the stopping criteria for the Krylov solver.

A. New inexactness criteria

We first present the inexactness criteria we derived that ensures a descent direction. When we solve the KKT system inexactly with or without a preconditioner, the search direction we compute satisfies

$$\begin{bmatrix} M & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} \tilde{p}_x \\ \tilde{p}_\lambda \end{bmatrix} = -\begin{bmatrix} g + N^T \lambda \\ c \end{bmatrix} + \begin{bmatrix} r_x \\ r_\lambda \end{bmatrix} \quad (\text{or } A\tilde{p} = -b + r). \tag{16}$$

We refer to $\tilde{p}$ as the inexact search direction. We want to ensure that the inexact search direction is a descent direction for the augmented Lagrangian merit function, which means

$$\nabla\phi^T \tilde{p} < 0. \tag{17}$$

We can express $\nabla\phi^T$ as a function of ρ, r_x and r_λ,

$$\begin{aligned} \nabla\phi^T \tilde{p} &= (g + N^T \lambda + \rho N^T c)^T \tilde{p}_x + c^T \tilde{p}_\lambda \\ &= (-M\tilde{p}_x - N^T \tilde{p}_\lambda + r_x)^T \tilde{p}_x + \rho c^T N \tilde{p}_x + c^T \tilde{p}_\lambda \\ &= -\tilde{p}_x^T M^T \tilde{p}_x + r_x^T \tilde{p}_x + \rho c^T (r_\lambda - c) + (2c^T - r_\lambda)\tilde{p}_\lambda. \end{aligned} \tag{18}$$

To ensure a descent direction, we can set

$$-\tilde{p}_x^T M^T \tilde{p}_x + r_x^T \tilde{p}_x < 0 \tag{19}$$

and

$$\rho c^T (r_\lambda - c) + (2c - r_\lambda)^T \tilde{p}_\lambda < 0. \tag{20}$$

We choose r_x as

$$\|r_x\| < \sigma_{min}(M) \|\tilde{p}_x\| \tag{21}$$

to satisfy (19). We write (20) as

$$\rho c^T (c - r_\lambda) > (2c - r_\lambda)^T \tilde{p}_\lambda. \tag{22}$$

We satisfy (22) in two steps. At first, we choose r_λ to ensure

$$c^T (c - r_\lambda) > 0, \tag{23}$$

this can be satisfied by choosing $\|r_\lambda\| < \|c\|$. Then we choose ρ as

$$\rho > \frac{2c^T \tilde{p}_\lambda - \tilde{p}_\lambda^T r_\lambda}{c^T (c - r_\lambda)}. \tag{24}$$

Note that at the k th iteration, we do not know $\left\|p_x^k\right\|$ before we solve the KKT system. Therefore, in our method, we approximate it as

$$\left\|\tilde{p}_x^k\right\| = \left\|\tilde{p}_x^{k-1}\right\|. \tag{25}$$

Using these criteria we derived, in our algorithm, we set the tolerances for the Krylov solver at k th optimization iteration as

$$\begin{aligned} \left\|r_x^k\right\| &< \sigma_{min}(M) \left\|\tilde{p}_x^{k-1}\right\|, \\ \left\|r_\lambda^k\right\| &< \eta \|c\|, \quad \eta \in (0,1), \end{aligned} \tag{26}$$

and the penalty parameter is chosen to satisfy

$$\rho > \frac{2c^T \tilde{p}_\lambda^k - \tilde{p}_\lambda^{kT} r_\lambda^k}{c^T (c - r_\lambda^k)}. \tag{27}$$

B. Adaptive Stopping Criteria for Krylov Solver

For the inexact quasi-Newton methods, we can think of the inexactness as a trade-off between efficiency and accuracy. Tighter tolerance can lead to more accurate search direction but it takes more time for the Krylov solver to converge. For all of the current inexact quasi-Newton methods, the Krylov solver stops when the inexact tolerance is satisfied. However, in some cases, we observe a large convergence rate at the iteration where the inexact tolerance is satisfied, and by running the solver for a few additional iterations, the norm of residuals can decrease dramatically, resulting in a much more accurate search direction.

In practice, the convergence of the Krylov solver differs from case to case, it depends on the dimension of the problem and also the preconditioner used to solve the KKT system. We want to propose a general idea that also considers the convergence rate and adaptively selects the stopping criteria for the Krylov solver.

The earliest iteration our algorithm stops is when the inexact tolerance (the tolerance used for the inexact method) is first satisfied. The latest iteration our algorithm stops is when the exact tolerance (the tolerance used for the exact method) is first satisfied. In this way, at each optimization iteration, the number of iterations our Krylov solver takes is always in between that of the regular inexact method and exact method. We refer to the inexact tolerance as the upper bound tolerance and exact tolerance as the lower bound tolerance.

We define a value n_1 as the number of extra Krylov iterations we are willing to afford in order to get an exact search direction. At the jth Krylov solver iteration when the upper bound tolerance is satisfied, we measure the current convergence rate of $\|r\|$ by fitting a linear regression using the values from the previous three iterations, $\left\|r^j\right\|$, $\left\|r^{j-1}\right\|$, $\left\|r^{j-2}\right\|$ and estimate the required number of additional iterations it takes to satisfy the lower bound tolerance as n_2. If the expected number of additional iterations is more than what we can afford, meaning $n_2 > n_1$, we stop the Krylov solver at the current iteration. If not, the Krylov solver proceeds one more iteration, and reduce n_1 by 1. The Krylov solver terminates when either the lower bound tolerance is satisfied or $n_1 < n_2$. The outline of the algorithm is in Alg. 2.

Algorithm 2 Krylov solver with adaptive stopping criteria

```
Specify the number of extra iterations we can afford as n_1.
while lower bound tolerance is not satisfied do
    if upper bound tolerance is satisfied then
        Estimate the convergence rate using ||r^j||, ||r^{j-1}||, ||r^{j-2}||
        Estimate the number of extra iterations required to meet the lower bound tolerance, n_2
        if n_2 > n_1 then
            stop
        end if
    end if
    n_1 = n_1 − 1
    Update r using the Krylov method.
```

One difficulty with Alg. 2 is defining a heuristic for n_1. From our numerical experiments with the algorithm, if solving the exact KKT system takes 30-40 Krylov iterations, it is effective to define n_1 as $n_1 = 30 - j$, where j is the current iteration number.

C. Adaptive inexact quasi-Newton algorithm

The complete outline of the adaptive inexact quasi-Newton algorithm is shown in Alg. 3.

Algorithm 3 Adaptive inexact quasi-Newton method

1: Specify the lower bound tolerance
2: **loop**
3: Evaluate c, g, N at x^k
4: Assemble A^k and b^k
5: Compute the upper bound tolerance as in (26)
6: Solve $A^k p^k \approx b^k$ using Alg. 2
7: Update ρ to satisfy (27)
8: find α^k via a backtracking line search method
9: update $\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha^k \begin{bmatrix} p_x^k \\ p_\lambda^k \end{bmatrix}$
10: update M via the BFGS formula

IV. Numerical Results

We use a cantilever bar design optimization problem to demonstrate the effectiveness of our algorithm. In this problem, we optimize the thickness (height) distribution of a cantilever bar under a load assuming a circular section. The cantilever bar is assumed to have a non-linear stress strain behavior, and is subject to an equality constraint to bound the volume of the cantilever bar. The optimization problem can be formulated as

$$\begin{aligned} \min_h \quad & q^T d \\ \text{s.t.} \quad & \mathcal{V}(h) = v_0 \\ \text{with} \quad & \mathcal{K}(h, d) d = q, \end{aligned} \tag{28}$$

where h is the thickness distribution vector; d is the displacement vector; q is the force vector; $\mathcal{V}$ is the function that computes the volume of the cantilever bar; v_0 is the allowable volume; $\mathcal{K}$ is the function that computes the stiffness matrix.

This problem is solved using three quasi-Newton (QN) methods, all using the full-space formulation. In the exact-QN method, we use a generic quasi-Newton method described in Alg. 1. In the inexact-QN method, we use the inexact quasi-Newton method with the inexact tolerance in (26). In the adaptive-inexact-QN method, we use the adaptive inexact quasi-Newton method we proposed as in Alg. 3, with a pre-selected n_1. For all three methods, we add a small number on the diagonal elements of the KKT matrix, and the KKT systems are solved using the CG method with an LU preconditioner.

Table 1 shows the results for various problem sizes. The results are also plotted in Figure 1. The total number of optimization iterations, total number of CG iterations and the total optimization time are compared for exact-QN, inexact-QN, and adaptive-inexact-QN methods. For this problem, the total CG iterations for the inexact methods are 2-3 times less than the exact method. However, for the 1000-element and 1500-element cases, the total computing time of the inexact methods are slightly greater than the exact method. This is because, for problem sizes not large enough, the time spent to perform one CG iteration is small, and for the inexact methods, they require extra time to compute for the inexact tolerance in each optimization iteration. Therefore, applying inexact methods does not give any benefits for this size of the problem.

In the larger problems, with 2000 and 3000 elements, inexact methods take 15% - 30% less computing time than the exact method. Comparing the inexact-QN method with the adaptive-inexact-QN method, the inexact-QN method takes around two times more optimization iterations to converge than the adaptive-inexact-QN method. In the inexact-QN

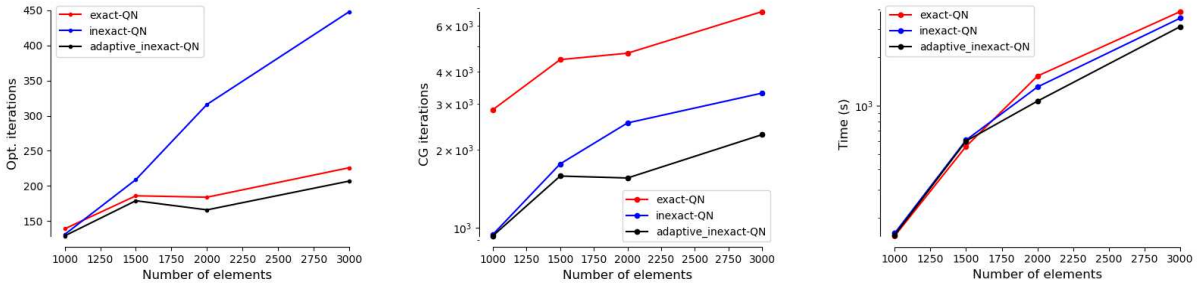
Table 1 Comparison of results of the exact-QN, inexact-QN and adaptive-inexact-QN methods applied to the cantilever bar optimization problem of various problem sizes.

Design variables	State variables	KKT matrix size	Method	Optimization iteration	Total CG iterations	Time (s)
1000	1002	3006×3006	exact-QN	139	2852	154
			inexact-QN	131	944	160
			adaptive-inexact-QN	129	933	156
1500	1502	4506×4506	exact-QN	186	4445	557
			inexact-QN	209	1765	610
			adaptive-inexact-QN	179	1585	602
2000	2002	6006×6006	exact-QN	184	4705	1532
			inexact-QN	316	2537	1309
			adaptive-inexact-QN	166	1558	1068
3000	3002	9006×9006	exact-QN	226	6803	3868
			inexact-QN	448	3305	3520
			adaptive-inexact-QN	207	2288	3117

method, we only use the inexactness tolerance as the stopping criteria for the CG solver, this tolerance guarantees that the search direction is a descent direction, but it does not assure a good convergence rate; this could result in taking more optimization iterations to converge. In adaptive-inexact-QN, we use the inexact tolerance with the adaptive stopping criteria algorithm. The adaptive stopping criteria ensure the robustness of the inexact method as it also makes the best use of the CG solver's performance.

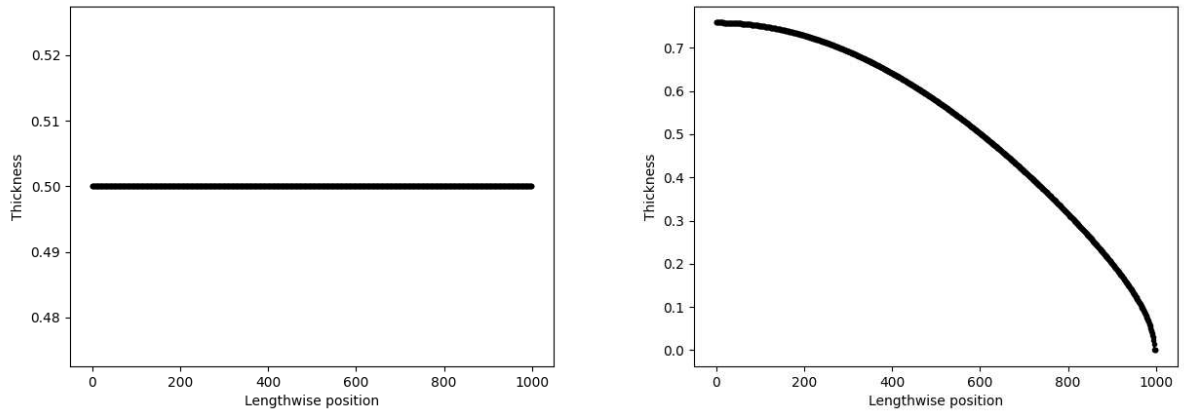
For the 3000-element case, the inexact-QN method takes 10% less time than the exact-QN method, and the adaptive-inexact-QN method takes 20% less time than the exact-QN method. In our implementation, the majority of the running time is spent on model evaluations. A better metric to demonstrate the effectiveness of our algorithm is to look at the total computing time for solving the KKT system, which is directly related to the total number of CG iterations. The inexact-QN method takes 50% less CG iterations than the exact-QN method, while the adaptive-inexact-QN method takes 65% less CG iterations than the exact-QN method.

For all the cases with different number of elements, the three quasi-Newton methods converged to the same solution. Figure 2 shows the thickness distribution of the initial design and the optimized design when the bar is modeled with 1000 elements. The convergence history of the three methods for the 3000-element case is shown in Figure 3.



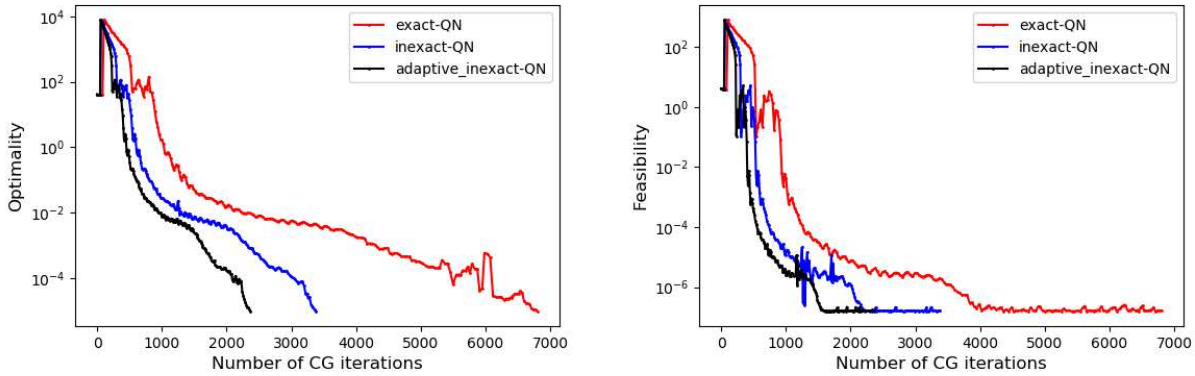
(a) Total optimization iterations **(b) Total CG iterations** **(c) Total optimization time**

Fig. 1 Comparison of the three methods across various number of elements



(a) Initial Design **(b) Optimized Design**

Fig. 2 Thickness distribution plots of the initial and optimized designs for the cantilever bar optimization problem with 1000 elements.



(a) Optimality convergence history **(b) Feasibility convergence history**

Fig. 3 Convergence history with number of CG iterations for the 3000-element case.

V. Conclusion

We presented an adaptive, inexact quasi-Newton algorithm for solving equality-constrained MDO problems using the SAND architecture. With the SAND architecture and other similar full-space methods, the size of the KKT system that must be solved in each iteration of the optimization algorithm is large, and thus preconditioned Krylov solvers are often used. We derived a new method to compute inexactness criteria for these iterative solvers, in a manner that ensures a descent direction for the line search. We also proposed a method for adaptively selecting the stopping criteria in a manner that takes into account the convergence rate of the Krylov solver—thus, going beyond the minimum requirement to guarantee a descent direction when the convergence rate is high.

We applied this adaptive, inexact quasi-Newton algorithm to a cantilever bar thickness optimization problem with a controllable problem size. We compared the traditional quasi-Newton method to the inexact quasi-Newton method with the descent-direction guarantee, and to the adaptive, inexact quasi-Newton method that considers both the descent-direction guarantee and convergence rate. In terms of total number of CG iterations, the inexact quasi-Newton method achieves a 50% reduction and the adaptive, inexact method achieves a 65% reduction for the problem configuration with 3000 elements, which is the largest one considered.

A limitation of these results is that they are obtained solely from optimization problems with only equality constraints. In the case of inequality-constrained optimization, this approach does not generalize directly to sequential quadratic

programming; however, it extends naturally to interior point methods for inequality-constrained problems. The expected significance of this work is the potential to make the SAND architecture and similar full-space methods feasible in a wider class of problems by reducing the cost of solving the larger systems that arise.

VI. Acknowledgments

The first author was partially supported by the First Year Fellowship from the Department of Mechanical and Aerospace Engineering at the University of California San Diego. The second author was supported by the National Science Foundation under grant no. 1917142.

References

[1] Ashley, H., "On making things the best-aeronautical uses of optimization," *Journal of Aircraft*, Vol. 19, No. 1, 1982, pp. 5–28.

[2] Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary optimization methods for aircraft preliminary design," *5th symposium on multidisciplinary analysis and optimization*, 1994, p. 4325.

[3] Antoine, N. E., and Kroo, I. M., "Framework for aircraft conceptual design and environmental performance studies," *AIAA journal*, Vol. 43, No. 10, 2005, pp. 2100–2109.

[4] Hwang, J. T., and Ning, A., "Large-scale multidisciplinary optimization of an electric aircraft for on-demand mobility," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018, p. 1384.

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

[6] Kenway, G., and Martins, J. R., "Aerostructural shape optimization of wind turbine blades considering site-specific winds," *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2008, p. 6025.

[7] Ning, A., and Dykes, K., "Understanding the benefits and limitations of increasing maximum rotor tip speed for utility-scale wind turbines," *Journal of physics: conference series*, Vol. 524, IOP Publishing, 2014, p. 012087.

[8] Balesdent, M., Bérend, N., Dépincé, P., and Chriette, A., "A survey of multidisciplinary design optimization methods in launch vehicle design," *Structural and Multidisciplinary optimization*, Vol. 45, No. 5, 2012, pp. 619–642.

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

[10] McAllister, C. D., and Simpson, T. W., "Multidisciplinary robust design optimization of an internal combustion engine," *J. Mech. Des.*, Vol. 125, No. 1, 2003, pp. 124–130.

[11] Kodiyalam, S., Yang, R., Gu, L., and Tho, C.-H., "Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment," *Structural and Multidisciplinary Optimization*, Vol. 26, No. 3-4, 2004, pp. 256–263.

[12] Gray, J. S., Hwang, J. T., Martins, J. R., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104.

[13] Hwang, J. T., and Martins, J. R., "A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 44, No. 4, 2018, pp. 1–39.

[14] Biros, G., and Ghattas, O., "Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part II: The Lagrange–Newton solver and its application to optimal control of steady viscous flows," *SIAM Journal on Scientific Computing*, Vol. 27, No. 2, 2005, pp. 714–739.

[15] Martins, J. R., and Lambe, A. B., "Multidisciplinary design optimization: a survey of architectures," *AIAA journal*, Vol. 51, No. 9, 2013, pp. 2049–2075.

[16] Cramer, E. J., Dennis, J. E., Jr, Frank, P. D., Lewis, R. M., and Shubin, G. R., "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, 1994, pp. 754–776.

[17] Haftka, R. T., "Simultaneous analysis and design," *AIAA journal*, Vol. 23, No. 7, 1985, pp. 1099–1103.

[18] Joshy, A. J., and Hwang, J. T., "A new architecture for large-scale system design optimization," *AIAA AVIATION 2020 FORUM*, 2020, p. 3125.

[19] Joshy, A. J., and Hwang, J. T., "Unifying Monolithic Architectures for Large-Scale System Design Optimization," *AIAA Journal*, 2021, pp. 1–11.

[20] Byrd, R. H., Schnabel, R. B., and Shultz, G. A., "A trust region algorithm for nonlinearly constrained optimization," *SIAM Journal on Numerical Analysis*, Vol. 24, No. 5, 1987, pp. 1152–1170.

[21] Powell, M., and Yuan, Y., "A trust region algorithm for equality constrained optimization," *Math. Program.*, Vol. 49, No. 1, 1991, pp. 189–211.

[22] Celis, M. R., "A trust region strategy for nonlinear equality constrained optimization," Tech. rep., 1985.

[23] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Some Theoretical Properties of an Augmented Lagrangian Merit Function." Tech. rep., STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB, 1986.

[24] Van der Vorst, H. A., "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on scientific and Statistical Computing*, Vol. 13, No. 2, 1992, pp. 631–644.

[25] Paige, C. C., and Saunders, M. A., "Solution of sparse indefinite systems of linear equations," *SIAM journal on numerical analysis*, Vol. 12, No. 4, 1975, pp. 617–629.

[26] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.

[27] Yang, H., Hwang, F.-N., and Cai, X.-C., "Nonlinear preconditioning techniques for full-space Lagrange–Newton solution of PDE-constrained optimization problems," *SIAM Journal on Scientific Computing*, Vol. 38, No. 5, 2016, pp. A2756–A2778.

[28] Hicken, J., and Alonso, J., "Comparison of reduced-and full-space algorithms for PDE-constrained optimization," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013, p. 1043.