This is the preprint version of the following article:
Bingran Wang, Anugrah Jo Joshy, and John T. Hwang. Equality-Constrained Engineering Design Optimization Using a Novel Inexact Quasi-Newton Method. AIAA Journal, 2022.

Published article: https://doi.org/10.2514/1.J061695
Preprint pdf: https://github.com/LSDOlab/lsdo_bib/blob/main/pdf/wang2022equality.pdf
Bibtex: https://github.com/LSDOlab/lsdo_bib/blob/main/bib/wang2022equality.bib

Equality-Constrained Engineering Design Optimization Using a Novel Inexact Quasi-Newton Method*

Bingran Wang[†], Anugrah Jo Joshy[‡], and John T. Hwang[§]

University of California San Diego, La Jolla, CA, 92093

Abstract

For engineering design optimization, the full-space formulation offers the potential for greater efficiency than the more commonly used reduced-space formulation. This potential is greater when the numerical model involves discretized partial differential equations or coupled disciplines. However, the full-space formulation results in a larger optimization problem with at least a factor of two increase in the number of optimization variables and equality constraints. Using Newton-type methods to solve such problems involves solving a large-scale and, often, ill-conditioned Karush–Kuhn–Tucker linear system at each optimization iteration. This can be time-consuming to solve even with a Krylov solver. If the number of iterations is reduced, the full-space formulation could be applied to a broader class of problems. This paper presents an inexact quasi-Newton algorithm with an adaptive extension for solving large-scale equality-constrained optimization problems. The new algorithm inexactly solves the Karush–Kuhn–Tucker system using new inexactness criteria that are derived to ensure a descent direction. The adaptive extension chooses the stopping condition of the Krylov solver by also taking its convergence rate into account. The paper presents results of numerical experiments applying this algorithm to three types of problems: six constrained optimization problems from the widely used CUTEst test suite, a bar thickness optimization problem, and a two-dimensional topology optimization problem. For all problems, the new algorithm consistently shows a roughly 50% reduction in the total number of Krylov solver iterations and a minimum of roughly 15% reduction in the optimization time. Moreover, the proposed approach for selecting the Krylov solver tolerance shows an improvement in all cases, whereas the existing forcing-parameter approach shows an increase in the number of Krylov iterations in some cases. These results indicate that this new method for selecting solver tolerances is effective and robust, and a good choice in algorithms that use a Krylov solver for solving the Karush–Kuhn–Tucker linear system.

*This paper was originally presented at the *AIAA AVIATION 2021 FORUM* held virtually on August 2-6, 2021. Paper number: AIAA 2021-3041. doi: https://doi.org/10.2514/6.2021-3041

[†]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.

1 Introduction

Engineering design optimization seeks improved design solutions using numerical optimization techniques. The problems arise from many domains such as fluids [1, 2], structures [3–5], aerodynamics [6, 7] and controls [8]. In these domains, the governing system of equations usually results from discretization of a partial differential equation (PDE). In more complex problems, the model could involve multiple disciplines, which is the domain of interest of multidisciplinary design optimization (MDO). A typical MDO problem is the conceptual design of an aircraft [9–11]. In this problem, disciplines such as aerodynamics, structures, and controls are tightly coupled. Optimizing the design of an aircraft 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 automobiles [12–14], wind turbines [15–17], launch vehicles [18], satellites [19, 20], electric aircraft [21], and electric vertical take-off and landing (eVTOL) aircraft [22].

The validity of the optimization result depends on the accuracy of the computational model of the system being designed. For disciplines like structures and aerodynamics, sufficient accuracy can be achieved using high-fidelity finite element solvers and computational fluid dynamics (CFD) solvers, respectively. In this case, we are solving a large-scale optimization problem with hundreds or thousands of design and even more state variables.

Many engineering design optimization problems are constrained nonlinear optimization problems with continuous variables. When these problems are large-scale, gradient-based methods like sequential quadratic programming (SQP) and interior-point methods are state-of-the-art methods to solve the optimization problems. Gradient-based methods require an efficient and accurate approach to calculate the derivatives. Common methods to compute the analytic derivatives are the complex-step method, algorithmic differentiation, direct method and adjoint method. The complex-step method [23] uses the Taylor series expansion with a purely imaginary step. This method can be accurate but it requires the model to support complex arithmetic. Algorithmic differentiation (AD) repeatedly applies the chain rule to each elementary function and operation in the model to compute the total derivatives. The direct and adjoint methods linearize the governing equations and formulate the problem as the solution of a linear system and subsequent insertion into an algebraic equation to compute the derivatives at a cost independent of the number of outputs and inputs, respectively. We often see adjoint method being used in PDE-constrained optimization and MDO problems as the number of functions is typically much less than the number of variables. In the MDO field, NASA's OpenMDAO software framework [24] enables a gradient-based approach using analytic derivatives to solve large-scale MDO problems. In OpenMDAO, the analytic derivatives are calculated using the unified derivative equation [25], which unifies the chain rule, adjoint method and other derivative computation methods.

The SQP and interior-point methods apply Newton's method to solve a linear Karush–Kuhn–Tucker (KKT) system for the update direction at each optimization iteration. When equality constraints are present, the KKT matrix is indefinite. For large-scale optimization problems, the KKT matrix is high-dimensional and can be ill-conditioned. Solving this KKT system necessitates using an iterative Krylov solver, but it can still be computationally expensive to solve this KKT system exactly (to a tight tolerance) at each iteration. One way to accelerate the computation is to solve the KKT system to a looser tolerance in intermediate optimization iterations, i.e., apply an inexact Newton approach. Compared with the exact Newton approach, the inexact Newton approach saves computing time for solving the KKT systems but loses accuracy in the computed update directions. If the inexact tolerance selection is too aggressive (i.e., the tolerances are too

large), the resulting errors in the linear solutions, which are the search directions, may cause an increase in the total number of optimization iterations, and there are no guarantees that the optimization algorithm will converge as it did with an exact linear solver.

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. Regarding the criterion used to compute the inexact tolerance, most of the inexact methods use a forcing parameter criterion to compute the inexact tolerance. However, for constrained optimization, they do not guarantee the update direction computed is a descent direction at each iteration. For those methods that guarantee a descent direction (e.g., the inexact Lagrange-Newton-Krylov-Schur (LNKS) method [26]), the criteria used are only applicable to a specific preconditioner, and the criteria are expensive to compute. For the termination of the Krylov solver, it stops as soon as the inexact 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 for the update direction.

In this paper, we propose an inexact quasi-Newton algorithm for large-scale equality-constrained optimization. This algorithm uses the new criteria we derived to compute the inexact tolerance for the Krylov solver. The new criteria, under certain approximations, can be easy to compute, and the tolerance computed assures a descent direction on the augmented Lagrangian merit function at each iteration. We also offer an extension for this algorithm to adaptively select the stopping criteria for the Krylov solver based on its convergence rate to capitalize whenever the convergence rate is high.

This paper is organized as follows. In Section 2, we give some background on optimization problem formulations for engineering design optimization problems, Newton's method to solve equality constrained optimization, and current inexact Newton methods. In Section 3, we present the outline of the inexact quasi-Newton method we proposed and the adaptive extension. In Section 4, we show the numerical results on several test problems. We make the conclusion in Section 5.

2 Background

2.1 Optimization Problem Formulation

The numerical model for an engineering system can, in general, be formulated using an implicit function that yields the solution of $\mathcal{R}(x, y) = 0$. Here, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ denote the design variables and state variables respectively, and $\mathcal{R} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^r$ is the vector-valued residual function. Depending on how to treat the residual function, there are two standard methods to formulate the optimization problems. In PDE-constrained optimization, they are called the reduced-space method and full-space method; in MDO, they are known as multidisciplinary feasible (MDF) [27] and simultaneous analysis and design (SAND) [28], respectively.

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. Considering only equality constraints, the reduced-space optimization problem is given by

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

where $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the objective function, $\mathcal{C} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^c$ is the vector-valued constraint function and $\mathcal{Y} : \mathbb{R}^n \to \mathbb{R}^m$ is the vector-valued function to the implicit solution of $\mathcal{R}(x, \mathcal{Y}(x)) = 0$.

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

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

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 optimization formulation of interest is the strong unification of reduced space and full space (SURF) method proposed by Joshy and Hwang [29, 30]. The SURF method is based on a full-space formulation and provides a hybrid version of reduced-space and full-space methods. Using a full-space formulation results in a high-dimensional optimization problem with a large number of equality constraints, which necessitates a gradient-based method and provides the motivation for the inexact approaches investigated here.

2.2 Newton's method for equality constrained optimization

We assume the optimization problems formulated in engineering design optimization are constrained nonlinear optimization problems with continuous variables. Newton-type methods (e.g. SQP and interior point methods) are the state-of-the-art to solve these optimization problems since they can achieve a second-order convergence rate.

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

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

For an optimization problem with only equality constraints, both SQP and interior point methods are equivalent to applying Newton's method to the KKT conditions. We first define the Lagrangian function as

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

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

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

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

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

At each iteration in Newton's method, it computes the search direction for the optimization variables and Lagrange multipliers by solving the Karush–Kuhn–Tucker (KKT) system, which is given by

$$\underbrace{\begin{bmatrix} M(x,\lambda) & N(x)^T \\ N(x) & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} p_x \\ p_\lambda \end{bmatrix}}_{p} = - \underbrace{\begin{bmatrix} g(x) + N(x)^T \lambda \\ \mathcal{C}(x) \end{bmatrix}}_{b}, \tag{6}$$

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; in this case, the quasi-Newton approach is attractive as it approximates 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. One could point out that using the inverse BFGS formula to update A^{-1} avoids solving the linear system in (6). However, the inverse BFGS formula is known to have an instability issue and may lead to convergence problems [31].

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 [32–34].

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

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

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. [35] 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 [36, 37], to ensure a descent direction. Ensuring a descent direction guarantees the existence of a step length result to satisfy the Armijo condition which will be mentioned later. A search direction p is a descent direction if

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

For the augmented-Lagrangian merit function, we have

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

assuming M is a positive-definite matrix. 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 standard quasi-Newton method. An outline of this algorithm is shown in Alg. 1.

Algorithm 1 Standard 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

2.3 Current inexact methods

In Alg. 1, we need to solve a linear KKT system as in (6). The KKT matrix, A, 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 can be severely ill-conditioned, which makes solving the KKT system difficult.

When the KKT system is large in size (say, 1,000 × 1,000 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 [38] and generalized minimal residual (GMRES) method [39]. Note that the CG method requires the KKT matrix to be positive definite. Most of the time, 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

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

where P is a preconditioner.

The effectiveness of a preconditioner is usually case-dependent. For instance, the active-set sequential quadratic programming (SQP) algorithm, Sparse Nonlinear OPTimizer (SNOPT) [40], uses a CG method without a preconditioner. In contrast, LNKS 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. $||r|| < 10^{-14}$. When we solve it inexactly, the tolerance is less tight, e.g. $||r|| < 10^{-6}$, 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 [41, 42]. However, choosing η is purely heuristic, and this criterion does not guarantee that the search direction calculated is a descent direction for the augmented Lagrangian merit function. Overall, albeit this criterion is easy to use and good performance is observed on many problems, this is not a rigorous way to compute the tolerance. Thus, it is not used in most widely used optimizers (e.g. SNOPT).

A more rigorous criterion is used in the inexact LNKS method [26], the criterion ensure a descent direction on the augmented Lagrangian merit function. However, it assumes a particular LU-equivalent preconditioner, and the criterion is not easy to compute.

3 Methodology

In this paper, we build on the idea of the inexact LNKS method to compute the inexact tolerance in a rigorous way to ensure a descent direction, but our proposed method is more generalized and can be used with any or without preconditioner. The criteria we used to compute the tolerance, under certain approximations, are easy to compute. We also propose an extension to the inexact methods to adaptively choose the stopping criteria for the Krylov solver and make best use of the Krylov solver's convergence rate.

3.1 Criteria for inexact tolerance

We first present the criteria we derived for inexact tolerance that ensures a descent direction on the augmented Lagrangian function. When we solve the KKT system inexactly with or without a preconditioner, the search direction we compute satisfies

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

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. It follows that

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

To ensure a descent direction, we choose to satisfy

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

and

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

To satisfy (19), we use the Cauchy-Schwarz inequality and bound the norm of r_x as

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

where $\sigma_{min}(M)$ denotes the minimum singular value of the matrix M. 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. 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

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

If we solve the KKT system exactly, namely, $r_\lambda = 0$, then this criterion to bound the penalty parameter is equivalent to the criterion for the exact quasi-Newton method in (10).

Note that at the k th iteration, we do not know $\left|\left|p_x^k\right|\right|$ before we solve the KKT system. Therefore, in our method, we approximate it using the value from the previous iteration. We set the tolerances for the Krylov solver at k th optimization iteration as

$$\begin{aligned} \left|\left| r_x^k \right|\right| &< \sigma_{min}(M) \left|\left| \tilde{p}_x^{k-1} \right|\right| , \\ \left|\left| r_\lambda^k \right|\right| &< \eta \, ||c|| \,, \quad \eta \in (0, 1). \end{aligned} \tag{25}$$

We decide on the penalty parameter after we solve the KKT system, and we can use p_λ^k to bound the penalty parameter as

$$\rho > \frac{2c^T \tilde{p}_\lambda^k - \tilde{p}_\lambda^{kT} r_\lambda^k}{c^T (c - r_\lambda^k)}. \tag{26}$$

This criterion to bound the penalty parameter does not greatly increase the penalty parameter value, and it will not affect the convergence rate of the optimization algorithm. The outline for the inexact quasi-Newton algorithm is shown in Alg. 2.

In our numerical experiments, we used the singular value decomposition (SVD) to find the smallest singular value of the matrix M, and the computation time is tolerable. However, in larger problems, applying SVD could require a significant amount of time. One alternative method to be considered is to use the inverse BFGS method to approximate M^{-1}. The M matrix is real and symmetric. Therefore, by the spectral theorem, we know that the eigenvalues are equal to the singular values. Thus,

$$\sigma_{min}(M) = \lambda_{min}(M), \tag{27}$$

Algorithm 2 Inexact quasi-Newton method

1: **loop**
2: Evaluate c, g, N at x^k
3: Assemble A^k and b^k
4: Compute the inexact tolerance as in (25)
5: Solve $A^k p^k \approx b^k$
6: Update ρ to satisfy (26)
7: Find α^k via a backtracking line search method
8: Update $\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha^k \begin{bmatrix} p_x^k \\ p_\lambda^k \end{bmatrix}$
9: Update M via the BFGS formula

where λ_{min} denotes the minimum eigenvalue. Since $\lambda_{min}(M) > 0$, the minimum eigenvalue of M is equal to the inverse of the maximum eigenvalue of M^{-1}, namely,

$$\lambda_{min}(M) = \frac{1}{\lambda_{max}(M^{-1})}. \tag{28}$$

With M^{-1} being approximated by the inverse BFGS method, we can apply the power iteration method to find $\lambda_{max}(M^{-1})$.

3.2 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 the 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 to extend the inexact quasi-Newton methods to also consider the convergence rate and adaptively select the stopping criteria for the Krylov solver.

We want to design our algorithm in this way: the earliest iteration in which 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 achieve this by defining a value n_1 as the number of extra Krylov iterations we are willing to execute 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, $\left|\left|r^j\right|\right|$, $\left|\left|r^{j-1}\right|\right|$, $\left|\left|r^{j-2}\right|\right|$ and estimate the number of additional iterations required to satisfy the lower bound tolerance as n_2. If

the expected number of additional iterations is more than tolerable, 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 algorithm is outlined in Alg. 3.

Algorithm 3 Krylov solver with adaptive stopping criteria

Specify the number of extra iterations we are willing to execute, n_1
while lower bound tolerance is not satisfied **do**
  if upper bound tolerance is satisfied **then**
    Estimate the convergence rate using $\left|\left|r^j\right|\right|, \left|\left|r^{j-1}\right|\right|, \left|\left|r^{j-2}\right|\right|$
    Estimate the number of extra iterations required to satisfy 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. 3 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 $n_1 = 20 - j$, where j is the current iteration number. We refer to the inexact quasi-Newton algorithm with the adaptive stopping criteria for the Krylov solver as the adaptive quasi-Newton algorithm. The algorithm is outlined in Alg. 4.

Algorithm 4 Adaptive inexact quasi-Newton method

Specify the lower bound tolerance
loop
  Evaluate c, g, N at x^k
  Assemble A^k and b^k
  Compute the upper bound tolerance as in (25)
  Solve $A^k p^k \approx b^k$ using Alg. 3
  Update ρ to satisfy (26)
  Find α^k via a backtracking line search method
  Update $\begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \lambda^k \end{bmatrix} + \alpha^k \begin{bmatrix} p_x^k \\ p_\lambda^k \end{bmatrix}$
  Update M via the BFGS formula

4 Numerical Results

4.1 Analytical optimization test problems

We first show the results from applying our method to six analytical, equality-constrained optimization problems with at least hundreds of optimization variables. All problems are selected from the CUTEst [43] test suite. The properties of the problems are shown in Tab. 1, where n

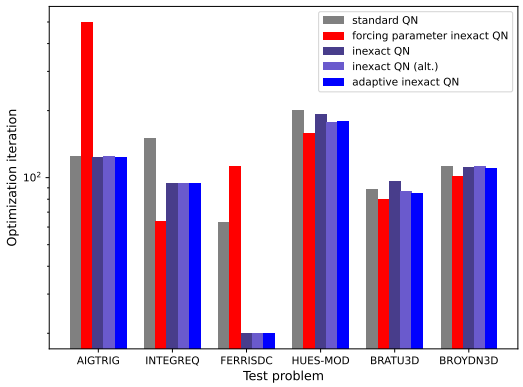
denotes the number of optimization variables and m denotes the number of equality constraints. The analytical test problems are solved using the standard quasi-Newton method, forcing parameter inexact quasi-Newton method, inexact quasi-Newton method, inexact quasi-Newton method using the alternative method to compute the tolerance and the adaptive inexact quasi-Newton method.

For all five methods, we add 10^{-3} on the diagonal elements of the KKT matrix to increase its condition number, and the KKT systems are solved using the CG method without a preconditioner. We use 10^{-14} for the exact tolerance for the standard quasi-Newton and the adaptive inexact quasi-Newton methods. For the forcing parameter method, we use $\eta = \min(0.5, \sqrt{||r||})$, which is a commonly used inexact method described in [44]. We choose the parameters introduced in our algorithms based on the parameter tuning results shown in Fig. 2. In the parameter tuning results, we compare the average percent reduction in total CG iterations relative to the standard quasi-Newton method. The results show that the optimal values are $\eta = 0.7$ and $n_1 = 3$. The results of the five methods on the six analytical test problems are shown as bar graphs in Fig. 1, where we show detailed comparison on each test problem in terms of total optimization iterations, total CG iterations, and total optimization time, along with the average percentage reduction in total CG iterations among all six test problems The three inexact methods proposed in this paper perform better than the standard quasi-Newton method on each of the test problems. In contrast, the forcing parameter inexact quasi-Newton method performs better than the standard quasi-Newton method on only four test problems. This is because the forcing parameter approach does not guarantee a descent direction on the augmented Lagrangian function for the equality-constrained optimization problems. Thus, the performance can be unstable—it can greatly reduce the CG iterations on certain problems but may lead to significantly more optimization iterations on other problems. On average, the forcing parameter inexact quasi-Newton method results in a 52.8% increase in CG reduction, while all the inexact methods we proposed show significant decrease. The inexact quasi-Newton method using the alternative approach shows an average of 47.9% reduction in CG iterations, which is slightly worse than the inexact quasi-Newton method which has 59.9% reduction. However, the computation time is slightly better than the inexact quasi-Newton method on most of the problems, especially the ones with larger size. This indicates a great potential of this method on very large-scale problems. Considering now the adaptive inexact quasi-Newton method, we see an average reduction of 61.5% in CG iterations. This is only a 1.6% improvement compared to the non-adaptive inexact quasi-Newton method; however, we note that all cases considered show an improvement. As these results suggest, the addition of the adaptive criterion does not always provide a significant improvement, but generally, it does not hurt the total computation time.

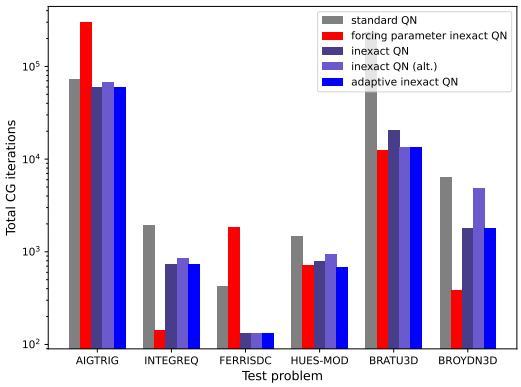
Table 1: Properties of the CUTEst test problems

	AIGTRIG	INTEGREQ	FERRISDC	HUES-MOD	BRATU3D	BROYDN3D
(n, m)	(200, 300)	(502, 500)	(2200, 210)	(5000, 2)	(3375, 3375)	(5000, 5000)

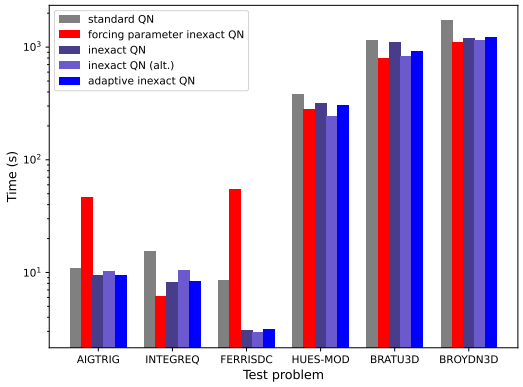
n is the number of optimization variables, m is the number of equality constraints.



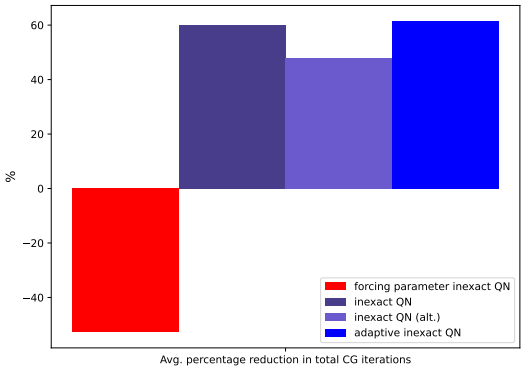
(a) Total optimization iterations



(b) Total CG iterations



(c) Total optimization time



(d) Average percentage reduction in total CG iterations

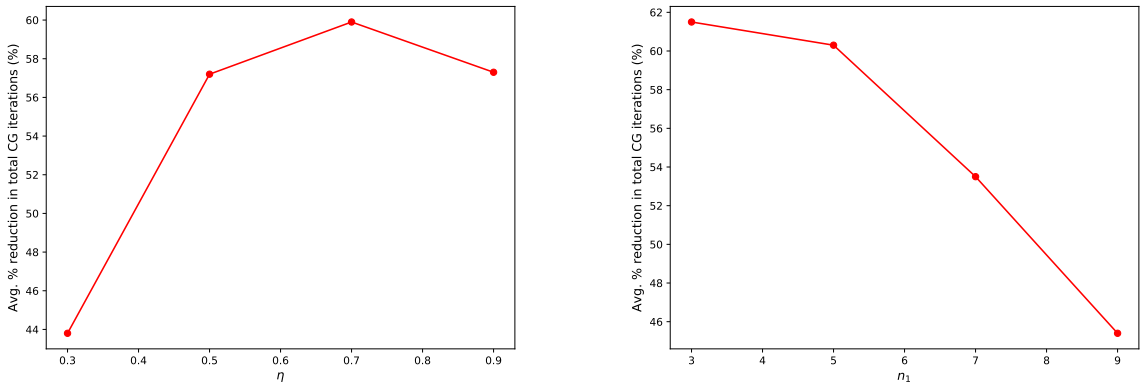
Figure 1: Comparison of results of the five methods applied to the CUTEst test problems

4.2 Bar thickness optimization

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

$$\begin{aligned} \min_{h} \quad & q^T d \\ \text{s.t.} \quad & \mathcal{V}(h) = v_0 \\ \text{with} \quad & \mathcal{K}(h,d)d = q, \end{aligned} \tag{29}$$

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 formulated using the full-space method,



(a) η for inexact quasi-Newton method

(b) n_1 for adaptive quasi-Newton method

Figure 2: Parameter tuning results on the CUTEst test problems

and the optimization problem is solved using standard quasi-Newton, inexact quasi-Newton and adaptive inexact quasi-Newton methods.

Table 2 shows the results for various problem sizes. The results are also plotted in Figure 4. The total number of optimization iterations, total number of CG iterations and the total optimization time are compared for the three methods. For this problem, the total CG iterations for the two inexact quasi-Newton methods are 2-3 times less than the standard quasi-Newton method. However, for the 1000-element and 1500-element cases, the total computing time of the inexact quasi-Newton methods are slightly greater than the standard quasi-Newton method. This is because, for certain problems that are not large-scale, the time spent to perform one CG iteration is too 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 when the problem size is not large enough.

In the larger problems, with 2000 and 3000 elements, the inexact quasi-Newton methods take 15% - 30% less computing time than the standard quasi-Newton method. Comparing the inexact quasi-Newton method with the adaptive inexact quasi-Newton method, the inexact quasi-Newton method takes around two times more optimization iterations to converge than the adaptive inexact quasi-Newton method. In the inexact quasi-Newton 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. In this case, this results in taking more optimization iterations to converge. In the adaptive inexact quasi-Newton method, 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.

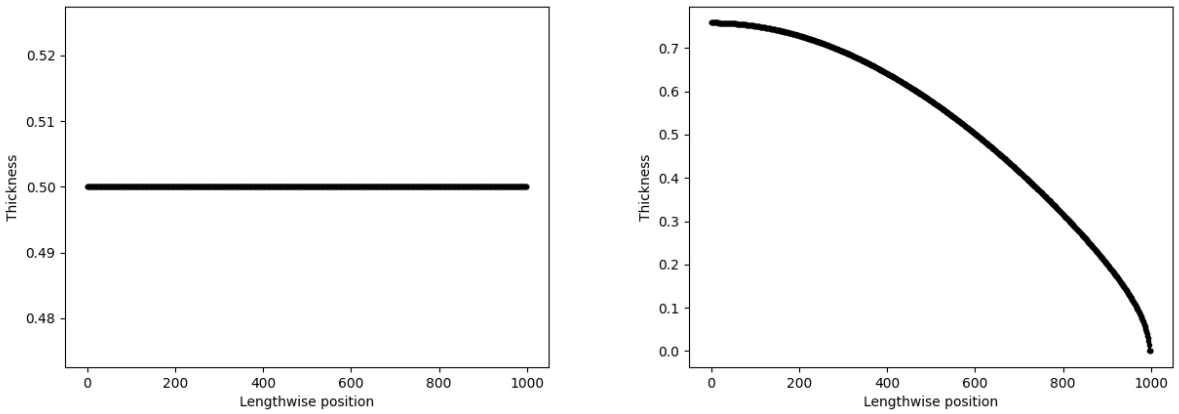
In our implementation, the majority of the running time is spent on model evaluations. Thus, a better metric to demonstrate the effectiveness of our algorithm is to look at the total computing time for solving the KKT system, which is directly related to the total number of CG iterations. The inexact quasi-Newton method takes 50% less CG iterations than the standard quasi-Newton method, while the adaptive inexact quasi-Newton method takes 65% less CG iterations than the

standard quasi-Newton method.

For all the cases with different number of elements, the three quasi-Newton methods converged to the same solution. Figure 3 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 5.

Table 2: Comparison of results of the three methods applied to the bar thickness optimization problem of various problem sizes

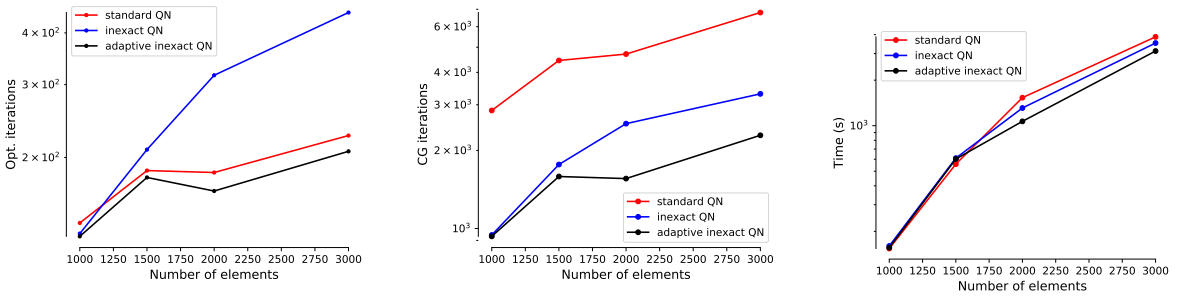
Design variables	State variables	KKT matrix size	Method	Optimization iteration	Total CG iterations	Time (s)
1000	1002	3006 × 3006	standard quasi-Newton	139	2852	154
			inexact quasi-Newton	131	944	160
			adaptive inexact quasi-Newton	129	933	156
1500	1502	4506 × 4506	standard quasi-Newton	186	4445	557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



(a) Initial Design

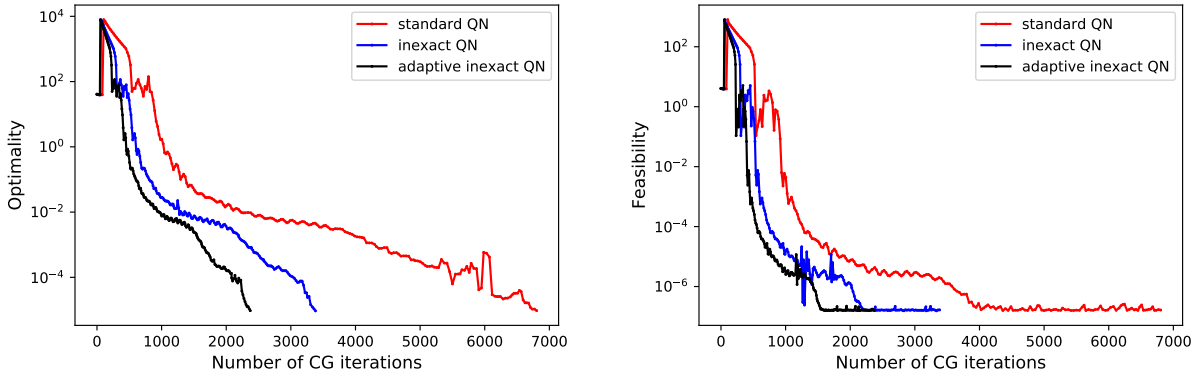
(b) Optimized Design

Figure 3: Thickness distribution plots of the initial and optimized designs for the bar thickness optimization problem with 1000 elements



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

Figure 4: Comparison of the three methods across various number of elements on the bar thickness optimization problem



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

Figure 5: Convergence history with number of CG iterations for the 3000-element bar thickness optimization problem

4.3 Topology optimization

The second engineering design optimization problem we use is a 2D topology optimization problem. In this problem, we optimize the material layout of a linear-elastic cantilever beam under a traction load on the right face of the beam. The beam is modeled in FEniCS [45, 46], with its domain modeled using uniform quadrilateral elements. We optimize the compliance of the beam subject to a constraint to bound the average density of the material. The optimization problem is formulated as

$$\begin{aligned} \min_{h} \quad & q^T d \\ \text{s.t.} \quad & \rho_{\text{avg}}(h) \leq 0.4\rho_{\max} \\ \text{with} \quad & \mathcal{K}(h)d = q, \end{aligned} \tag{30}$$

where h is the density distribution vector; d is the displacement vector; q is the force vector; ρ_{avg} is the average material density; $\rho_{\max}$ maximum material density allowed; $\mathcal{K}$ is the function that computes the stiffness matrix.

This is a practical design optimization problem to demonstrate our algorithm, but our current method is only applicable for equality-constrained optimization problems. Thus, we use SNOPT to compute a local optimum and use the corresponding active set to convert the inequality constraints into equality constraints. This problem is formulated in the full-space method and solved using standard quasi-Newton, inexact quasi-Newton and adaptive inexact quasi-Newton methods.

Table 3: Comparison of results of the three methods applied to the topology optimization problem of various problem sizes

Design variables	KKT matrix size	Method	Optimization iteration	Total CG iterations	Time (s)
40×20	4275×4275	exact quasi-Newton	84	1544	266
		inexact quasi-Newton	84	834	241
		adaptive inexact quasi-Newton	84	850	245
60×30	9440×9440	exact quasi-Newton	120	4117	1001
		inexact quasi-Newton	149	3364	947
		adaptive inexact quasi-Newton	123	2588	780
80×40	19646×19646	exact quasi-Newton	137	12911	3000
		inexact quasi-Newton	142	4311	2123
		adaptive inexact quasi-Newton	137	4087	2011

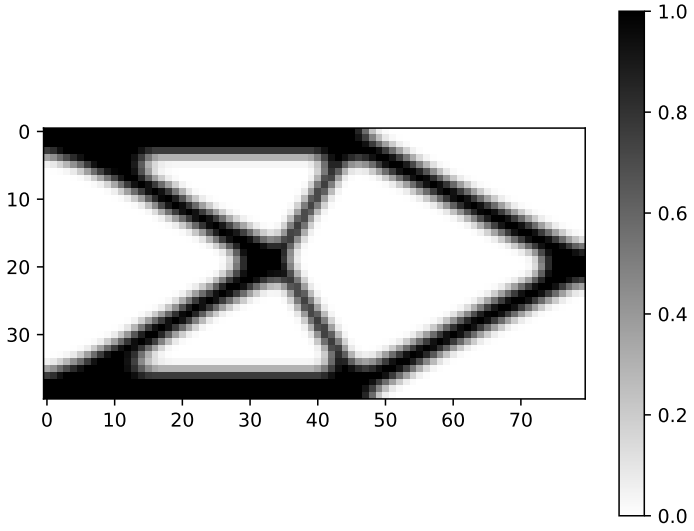
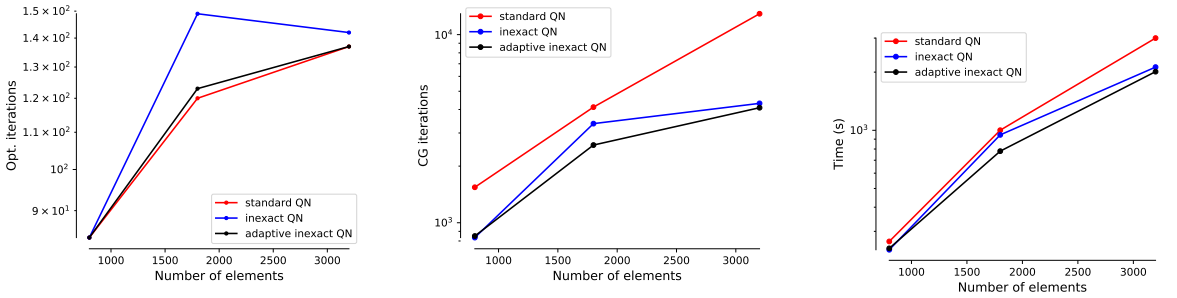
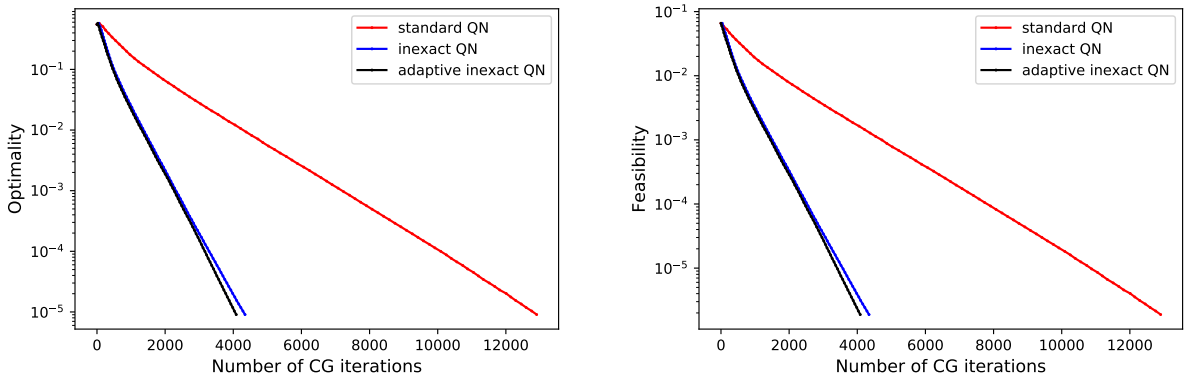


Figure 6: Material layout plot of the optimized design for the topology optimization problem with 3200 elements.



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

Figure 7: Comparison of the three methods for the topology optimization problem across various number of elements



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

Figure 8: Convergence history with number of CG iterations for the 3200-element topology optimization problem.

Table 3 shows the results for various problem sizes. The results are also plotted in Figure 7. The three methods are compared on total number of optimization iterations, total number of CG iterations and total optimization time. For this problem, we observe similar performance results for the two inexact quasi-Newton methods as in the bar thickness optimization problem. The two inexact quasi-Newton methods demonstrate better performance than the standard quasi-Newton method in terms of total computation time. For the 800-element case, the total computation time of the inexact methods is only 10% less than the exact method. In the larger-size problems, with 1800 and 3200 elements, the inexact methods become more effective. In the 3200-element case, the adaptive inexact quasi-Newton method can save more than 30% of the computation time. However, in the 1800-element case, the inexact quasi-Newton method takes 29 more optimization iterations to converge. As a result, it only saves 5% of the computing time compared to the exact quasi-Newton method. However, with the use of the adaptive criteria, the adaptive inexact quasi-Newton method, takes fewer optimization iterations to converge than the inexact quasi-Newton method. Particularly in the 1800-element case, an extra 15% computing time is saved as a result.

In terms of the total CG iterations, the inexact quasi-Newton method takes 18% and 66% less CG iterations than the standard quasi-Newton method in the 1800-element and 3200-element cases, respectively. The adaptive quasi-Newton method takes 37% and 68% less CG iterations than the exact quasi-Newton method in the 1800-element and 3200-element cases, respectively.

The three quasi-Newton methods converged to the same solution for all the cases with different problem sizes. Figure 6 shows the material layout of the optimized design when the beam is modelled with 3200 elements. The convergence history of the three methods for the 3200-element case is shown in Figure 8.

5 Conclusion

In this paper, we presented an inexact quasi-Newton method with an adaptive extension. This method targets large-scale optimization problems with thousands or more optimization variables and equality constraints. This kind of optimization problem often appears when solving engineering design problems in the full-space formulation. In our proposed method, we derived new criteria to compute inexactness tolerance for the Krylov solver in a manner that ensures a descent direction for the line search. We also proposed an extension to this method that adaptively selects the stopping criteria to consider the Krylov solver's convergence rate. Thus, it goes beyond the minimum required to guarantee a descent direction when the convergence rate is high.

We used several test problems with various problem sizes to show indicative results. The test problems include six constrained optimization problems from the CUTEst test suite, a bar thickness optimization problem, and a 2D topology optimization problem. For all the test problems, using the new inexactness tolerance can consistently reduce the total number of CG iterations by roughly 50% and additionally using the adaptive extension can further reduce the total number of CG iterations in certain problems. In the largest-size topology optimization problem, using the new inexactness tolerance with the adaptive extension saves more than 30% of the computing time and 65% of the total CG iterations.

A limitation of our method is that our algorithm only applies to equality-constrained optimization problems, but many practical engineering design optimization problems have inequality constraints. The future work is to extend this idea to the inequality-constrained optimization algorithms such as the sequential quadratic programming and interior point methods. The expected significance of this work is the potential to make the full-space methods and similar the SAND architecture feasible on a broader class of problems by reducing the cost of solving the larger systems that arise.

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. The material presented in this paper is, in part, based upon work supported by NASA under award No. 80NSSC21M0070.

References

[1] Bijan Mohammadi and Olivier Pironneau. "Shape optimization in fluid mechanics". In: *Annu. Rev. Fluid Mech.* 36 (2004), pp. 255–279. DOI: 10.1146/annurev.fluid.36.050802.121926.

[2] Yusuke Tahara, Satoshi Tohyama, and Tokihiro Katsui. "CFD-based multi-objective optimization method for ship design". In: *International journal for numerical methods in fluids* 52.5 (2006), pp. 499–527. DOI: 10.1002/fld.1178.

[3] Garret N Vanderplaats. "Structural design optimization status and direction". In: *Journal of Aircraft* 36.1 (1999), pp. 11–20. DOI: 10.2514/6.1997-1407.

[4] Bret K Stanford and Peter D Dunning. "Optimal topology of aircraft rib and spar structures under aeroelastic loads". In: *Journal of Aircraft* 52.4 (2015), pp. 1298–1311. DOI: 10.2514/6.2014-0633.

[5] Kai A James, Graeme J Kennedy, and Joaquim RRA Martins. "Concurrent aerostructural topology optimization of a wing box". In: *Computers & Structures* 134 (2014), pp. 1–17. DOI: 10.1016/j.compstruc.2013.12.007.

[6] Antony Jameson. "Aerodynamic shape optimization using the adjoint method". In: *Lectures at the Von Karman Institute, Brussels* (2003).

[7] Marian Nemec, David W Zingg, and Thomas H Pulliam. "Multipoint and multi-objective aerodynamic shape optimization". In: *AIAA journal* 42.6 (2004), pp. 1057–1065. DOI: 10.2514/1.10415.

[8] Vassilis Sakizlis, John D Perkins, and Efstratios N Pistikopoulos. "Parametric controllers in simultaneous process and control design optimization". In: *Industrial & Engineering Chemistry Research* 42.20 (2003), pp. 4545–4563. DOI: 10.1021/ie0209273.

[9] Holt Ashley. "On making things the best-aeronautical uses of optimization". In: *Journal of Aircraft* 19.1 (1982), pp. 5–28. DOI: 10.2514/3.57350.

[10] Ilan Kroo, Steve Altus, Robert Braun, Peter Gage, and Ian Sobieski. "Multidisciplinary optimization methods for aircraft preliminary design". In: *5th symposium on multidisciplinary analysis and optimization*. 1994, p. 4325. DOI: 10.2514/6.1994-4325.

[11] Nicolas E Antoine and Ilan M Kroo. "Framework for aircraft conceptual design and environmental performance studies". In: *AIAA journal* 43.10 (2005), pp. 2100–2109. DOI: 10.2514/6.2004-4314.

[12] Charles D McAllister and Timothy W Simpson. "Multidisciplinary robust design optimization of an internal combustion engine". In: *J. Mech. Des.* 125.1 (2003), pp. 124–130. DOI: 10.1115/detc2001/dac-21124.

[13] S Kodiyalam, RJ Yang, L Gu, and C-H Tho. "Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment". In: *Structural and Multidisciplinary Optimization* 26.3-4 (2004), pp. 256–263. DOI: 10.1007/s00158-003-0343-2.

[14] Namwoo Kang, Michael Kokkolaras, Panos Y Papalambros, Seungwon Yoo, Wookjin Na, Jongchan Park, and Dieter Featherman. "Optimal design of commercial vehicle systems using analytical target cascading". In: *Structural and Multidisciplinary Optimization* 50.6 (2014), pp. 1103–1114. DOI: 10.2514/6.2012-5524.

[15] P Fuglsang and H Aagaard Madsen. "Optimization method for wind turbine rotors". In: *Journal of Wind Engineering and Industrial Aerodynamics* 80.1-2 (1999), pp. 191–206. DOI: 10.1016/s0167-6105(98)00191-3.

[16] Gaetan Kenway and Joaquim RRA Martins. "Aerostructural shape optimization of wind turbine blades considering site-specific winds". In: *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 2008, p. 6025. DOI: 10.2514/6.2008-6025.

[17] Andrew Ning and K Dykes. "Understanding the benefits and limitations of increasing maximum rotor tip speed for utility-scale wind turbines". In: *Journal of physics: conference series*. Vol. 524. 1. IOP Publishing. 2014, p. 012087. DOI: 10.1088/1742-6596/524/1/012087.

[18] Mathieu Balesdent, Nicolas Bérend, Philippe Dépincé, and Abdelhamid Chriette. "A survey of multidisciplinary design optimization methods in launch vehicle design". In: *Structural and Multidisciplinary optimization* 45.5 (2012), pp. 619–642. DOI: 10.1007/s00158-011-0701-4.

[19] Cyrus Jilla and David Miller. "A multiobjective, multidisciplinary design optimization methodology for the conceptual design of distributed satellite systems". In: *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 2002, p. 5491. DOI: 10.2514/6.2002-5491.

[20] John T Hwang, Dae Young Lee, James W Cutler, and Joaquim RRA Martins. "Large-scale multidisciplinary optimization of a small satellite's design and operation". In: *Journal of Spacecraft and Rockets* 51.5 (2014), pp. 1648–1663. DOI: https://doi.org/10.2514/1.A32751.

[21] John T Hwang and Andrew Ning. "Large-scale multidisciplinary optimization of an electric aircraft for on-demand mobility". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018, p. 1384. DOI: https://doi.org/10.2514/6.2018-1384.

[22] Tae H Ha, Keunseok Lee, and John T Hwang. "Large-scale multidisciplinary optimization under uncertainty for electric vertical takeoff and landing aircraft". In: *AIAA Scitech 2020 Forum*. 2020, p. 0904. DOI: 10.2514/6.2020-0904.

[23] Joaquim RRA Martins, Peter Sturdza, and Juan J Alonso. "The complex-step derivative approximation". In: *ACM Transactions on Mathematical Software (TOMS)* 29.3 (2003), pp. 245–262. DOI: 10.1145/838250.838251.

[24] Justin S Gray, John T Hwang, Joaquim RRA Martins, Kenneth T Moore, and Bret A Naylor. "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization". In: *Structural and Multidisciplinary Optimization* 59.4 (2019), pp. 1075–1104. DOI: https://doi.org/10.1007/s00158-019-02211-z.

[25] John T Hwang and Joaquim RRA Martins. "A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives". In: *ACM Transactions on Mathematical Software (TOMS)* 44.4 (2018), p. 37. DOI: https://doi.org/10.1145/3182393.

[26] George Biros and Omar Ghattas. "Inexactness issues in the Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization". In: *Large-scale PDE-constrained optimization*. Springer, 2003, pp. 93–114. DOI: 10.1007/978-3-642-55508-4_6.

[27] Evin J Cramer, John E Dennis Jr, Paul D Frank, Robert Michael Lewis, and Gregory R Shubin. "Problem formulation for multidisciplinary optimization". In: *SIAM Journal on Optimization* 4.4 (1994), pp. 754–776. DOI: 10.1137/0804044.

[28] Raphael T Haftka. "Simultaneous analysis and design". In: *AIAA journal* 23.7 (1985), pp. 1099–1103. DOI: 10.2514/3.9043.

[29] Anugrah Jo Joshy and John T Hwang. "A new architecture for large-scale system design optimization". In: *AIAA AVIATION 2020 FORUM*. 2020, p. 3125. DOI: 10.2514/6.2020-3125.

[30] Anugrah Jo Joshy and John T Hwang. "Unifying Monolithic Architectures for Large-Scale System Design Optimization". In: *AIAA Journal* (2021), pp. 1–11. DOI: 10.2514/1.j059954.

[31] Yonathan Bard. *On a Numerical Instability of Davidon-like Methods*. IBM Data Processing Division (New York), 1967. DOI: 10.1090/s0025-5718-1968-0232533-5.

[32] Richard H Byrd, Robert B Schnabel, and Gerald A Shultz. "A trust region algorithm for nonlinearly constrained optimization". In: *SIAM Journal on Numerical Analysis* 24.5 (1987), pp. 1152–1170. DOI: 10.1137/0724076.

[33] MJD Powell and Y Yuan. "A trust region algorithm for equality constrained optimization". In: *Math. Program.* 49.1 (1991), pp. 189–211. DOI: 10.1007/bf01588787.

[34] Maria Rosa Celis. *A trust region strategy for nonlinear equality constrained optimization*. Tech. rep. 1985. DOI: 10.21236/ada454933.

[35] Philip E Gill, Walter Murray, Michael A Saunders, and Margaret H Wright. *Some Theoretical Properties of an Augmented Lagrangian Merit Function*. Tech. rep. STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB, 1986.

[36] George Biros and Omar Ghattas. "Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part I: The Krylov–Schur solver". In: *SIAM Journal on Scientific Computing* 27.2 (2005), pp. 687–713. DOI: 10.1137/s106482750241565x.

[37] George Biros and Omar Ghattas. "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". In: *SIAM Journal on Scientific Computing* 27.2 (2005), pp. 714–739. DOI: 10.1137/s1064827502415661.

[38] Henk A Van der Vorst. "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". In: *SIAM Journal on scientific and Statistical Computing* 13.2 (1992), pp. 631–644. DOI: 10.1137/0913035.

[39] Christopher C Paige and Michael A Saunders. "Solution of sparse indefinite systems of linear equations". In: *SIAM journal on numerical analysis* 12.4 (1975), pp. 617–629. DOI: 10.1137/0712047.

[40] Philip E Gill, Walter Murray, and Michael A Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". In: *SIAM review* 47.1 (2005), pp. 99–131. DOI: https://doi.org/10.1137/S0036144504446096.

[41] Haijian Yang, Feng-Nan Hwang, and Xiao-Chuan Cai. "Nonlinear Preconditioning Techniques for Full-Space Lagrange–Newton Solution of PDE-Constrained Optimization Problems". In: *SIAM Journal on Scientific Computing* 38.5 (2016), A2756–A2778. DOI: https://doi.org/10.1137/15M104075X.

[42] Jason Hicken and Juan Alonso. "Comparison of reduced-and full-space algorithms for PDE-constrained optimization". In: *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. 2013, p. 1043. DOI: 10.2514/6.2013-1043.

[43] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. "CUTEst: a Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization". In: *Computational Optimization and Applications* 60.3 (2015), pp. 545–557. DOI: https://doi.org/10.1007/s10589-014-9687-3.

[44] Nocedal Jorge and J Wright Stephen. *Numerical optimization*. 2006. DOI: 10.1007/b98874.

[45] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. "The FEniCS project version 1.5". In: *Archive of Numerical Software* 3.100 (2015). DOI: 10.11588/ans.2015.100.20553.

[46] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*. Vol. 84. Springer Science & Business Media, 2012. DOI: 10.1007/978-3-642-23099-8.