# A MATRIX-FREE ALGORITHM FOR REDUCED-SPACE PDE-CONSTRAINED OPTIMIZATION

By

Pengfei Meng

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: AERONAUTICAL ENGINEERING

Approved by the
Examining Committee:

---

Jason E. Hicken, Thesis Adviser

---

Assad A. Oberai, Member

---

Lucy T. Zhang, Member

---

John E. Mitchell, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2018
(For Graduation May 2018)

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGMENT

I would like to thank my advisor, Professor Jason E. Hicken, for the numerous discussions he patiently had with me for the past five years. His broad knowledge and insights in numerical simulation and design field continue to motivate me. Beyond being professional, he is a very nice person to work with, and it is my pleasure to have him as my advisor.

I thank my doctoral committee members, Prof. Assad A. Oberai, Prof. Lucy T. Zhang, Prof. John E. Mitchell for the time they spent on my candidacy and thesis defense. Their valuable inputs greatly help improve the quality of my thesis work.

Many thanks to my fellow lab mates in CII lab for providing a lively and humorous environment: Alp Dener, Anthony Ashley, Jared Crean, Jianfeng Yan, Kinshuk Panda. In particular, thank you to Alp Dener, whose work in Kona laid the foundation for my thesis research, and who very generously gave me numerous valuable suggestions.

I gratefully acknowledge Dr. Gaetan Kenway, Dr. Charles (Sandy) Mader in MDO design lab at University of Michigan for their help with the CFD solver ADflow.

Lastly I am grateful to have my family with me, Yongjian, and most recently Emma, my mom and my sister.

This thesis is jointly funded by the NASA Learn project, Prof. Hicken's startup grant, and TA funding from the MANE department. Specially, thanks to OGE at RPI for supporting me with Childbirth Accommodation in fall 2017 after my baby was born, which enable me to spend more time with my baby and smoothly finish my PhD.

# ABSTRACT

This thesis present a matrix-free method for partial differential equation (PDE) constrained optimization problems formulated in the reduced space. When many state-based constraints are present in the reduced-space formulation, the constraint Jacobian can become prohibitively expensive to compute explicitly, because each constraint gradient requires the solution of a distinct adjoint PDE. This leads many practitioners to use constraint aggregation, which can produce overly conservative solutions. To avoid conservative solutions as well as the expense of forming the constraint Jacobian, we adopt a matrix-free inexact-Krylov optimization framework. This choice introduces additional challenges related to globalization and preconditioning. To address globlization, the proposed method uses a homotopy continuation approach and a predictor-corrector algorithm to trace the solution curve. The predictor and corrector linear systems are solved using a Krylov iterative method with the necessary matrix-vector products evaluated via second-order adjoints. To cope with the poorly conditioned primal-dual system, a matrix-free preconditioner is proposed that uses a low-rank approximation of the Schur complement of the primal-dual matrix; the low-rank approximation is constructed using a fixed number of iterations of the Lanczos method. The algorithm is verified using analytical problems, a subset of CUTEr problems, a stress-constrained mass minimization problem, and an aerodynamic shape optimization problem. The method shows promising performance relative to a state-of-the-art matrix-based active-set algorithm, particularly for large numbers of design variables.

# CHAPTER 1
# INTRODUCTION

## 1.1 Motivation

Global warming has been unequivocally proven by scientific evidence [1]. Furthermore, there is also strong evidence that human activities, especially anthropogentic emissions of carbon dioxide (CO2), are the major source of this warming.

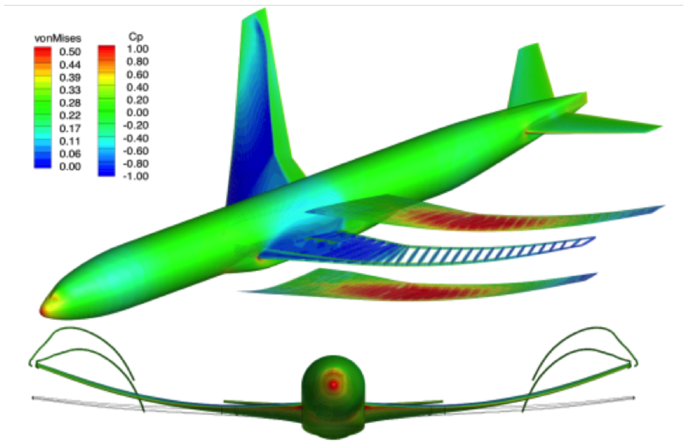
Every industry has an ethical responsibility to address climate change, including the air transport industry, which is responsible for about 2% of the manmade carbon dioxide (CO2) emissions [2], [3]. While this percentage may seem small, researchers suggest that aviation's share of CO2 emissions should be multiplied by 1.9 times [2], [3] to incorporate the impact of altitude and other emissions, like NOx and water vapors. Furthermore, with the number of passengers increasing at an average of 5% each year [4], [2], perhaps more in developing markets, the impact of aviation on the environment will only increase. It is estimated that approximately 27,000 new passenger aircraft will be demanded between now and 2030 [2]. In summary, the total contribution of aviation to human emissions of CO2 and other effects will likely rise to 5% and in a worst-case to 15% [2] by 2050.

In light of these figures, reducing the impact on the environment is becoming a driving factor for future aircraft design [5]. For example, the Advisory Council for Aeronautics Research in Europe (ACARE) is enforcing strict emission targets in order to reduce CO2 emissions per passenger kilometer by 75%, NOx by 90% and perceived noise by 65% by 2050 relative to the year 2000 [6], [7]. To design future aircraft that meet such targets, the aviation industry needs to consider a range of strategies, including improved efficiency through optimization.

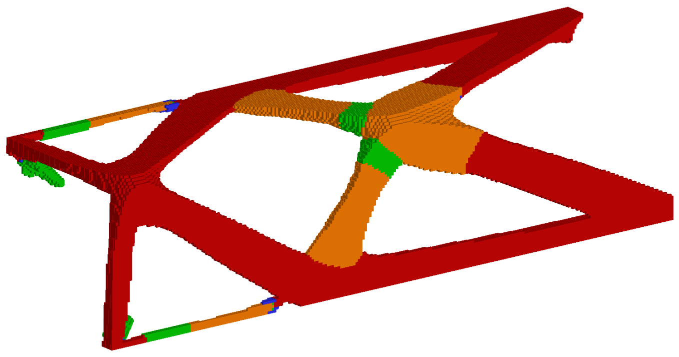
## 1.2 PDE-constrained Optimization

Numerical optimization is a powerful tool that can be used to inform the design of aircraft. In particular, in aircraft conceptual design stage, engineering design optimization can reveal valuable insights about the design trade-offs and help engineers make detailed and informed decisions. Moreover, optimization is increasingly used during detailed design to refine the shape and structural layout of aircraft. However, the optimization must be coupled with sufficiently accurate models if the results are to be reliable. For example, in this work we will consider partial-differential equations (PDEs) models that can capture the complex nonlinear physics present in flight.

Engineering design optimization problems that are governed by PDEs arise in many engineering applications including aerodynamic shape optimization [8], [9], [10], structural optimization [11], [8], [12], and thermodynamic optimization [13], [14], [15]. Figure 1.1 shows two examples of PDE-constrained design problems: the first one is an aero-structural optimization problem; the second one is a topology optimization problem.



(a) Aero-Structure Optimization [16]



(b) Topology Optimization [17]

**Figure 1.1. Large-scale PDE-constrained optimization**

In PDE-constrained optimization, an optimization library is coupled with one or more PDE models. Figure 1.2 illustrates the schematic diagram of the PDE-constrained optimization process[1]. As shown in Figure 1.2, a typical optimization iteration begins with updating the computational mesh (if necessary), followed by the primal PDE solve. The solution of PDE can then be used to evaluate the objective and constraints. If a gradient or Jacobian is requested by the optimization algorithm, then one or more adjoint PDEs must be evaluated.

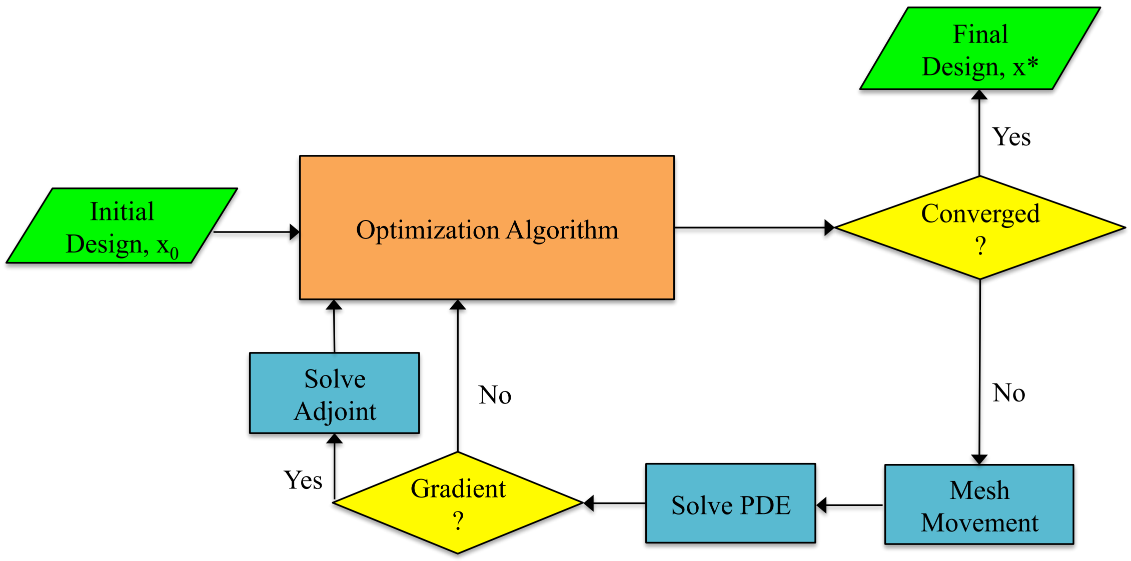


**Figure 1.2. Schematic process of PDE-constrained optimization**

Computational cost is an important consideration in PDE-constrained optimization. Again referring to Figure 1.2, we see that each optimization iteration requires the solution of the PDE. This subproblem itself can be a formidable task in high-performance computing. Furthermore, gradients and Jacobians require the solution of additional linearized PDEs, e.g. adjoints. In the next section, we will discuss how these differences cause significant challenges for conventional optimization algorithms.

[1]More precisely, Figure 1.2 illustrates a reduced-space PDE-constrained optimization.

## 1.3 Conventional Optimization Algorithms and Their Limitations

Conventional gradient-based optimization algorithms [18], [19], [20] have been used extensively in PDE-constrained optimizations, particularly for problems with relatively few state-based constraints. For example, [21] and [22] used SNOPT (Sparse Nonlinear OPTimizer) [20] through the Python interface pyOpt [23] for the investigations of the aerodynamic and aerostructural optimization on the Common Research Model based on RANS. Because there were only three state-based outputs of interest, drag, lift, and pitch moment coefficients, they use adjoint methods [24], [25], [26], [27], [28], [29], [30] to assemble the total gradients and feed them to SNOPT. The cost of an adjoint solution is independent of the number of design variables for each state-based output. Other general purpose optimization algorithms, such as IPOPT [31] and Knitro [32] have been used for aerodynamic design problems [33], [34] as well.

However, conventional optimization algorithms are not well suited for large-scale PDE-constrained optimization problems with many design variables and state-based constraints. Large-scale problems with many (thousands or more) design variables are a problem, because conventional optimization algorithms typically rely on limited-memory quasi-Newton methods, which have linear asymptotic convergence rates [35]. Furthermore, in the presence of many state-based constraints, assembling the total constraint Jacobians can become prohibitively expensive, as each constraint gradient requires the solution of an adjoint equation whose cost is comparable to that of the governing PDE.

This work is particularly concerned with addressing the costs associated with the constraint Jacobian. Conventional optimization methods require the explicit constraint Jacobian at every major iteration in order to factor the matrix and determine a basis for its null-space; this basis is required by many algorithms for constrained optimization [36]. As already explained, these matrix-based algorithms are not practical when many constraints are present. Therefore, a different class of algorithm is needed.

## 1.4 PDE-constrained Optimization Algorithms

### 1.4.1 Full-space and Reduced-space Approaches

There are two broad classes of algorithm used to solve PDE-constrained optimization problems: full-space methods and reduced-space methods. In order to describe these two approaches and their relative merits, consider the following generic PDE-constrained optimization problem,

$$\begin{aligned} \min_{x,u} \quad & f(x,u) & \\ \text{subject to} \quad & h(x,u) & = 0 \\ & g(x,u) & \geq 0 \\ \text{governed by} \quad & \mathcal{R}(x,u) & = 0, \end{aligned} \tag{1.1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^v$ are the design and state vectors, respectively, and $f : \mathbb{R}^n \times \mathbb{R}^v \to \mathbb{R}, h : \mathbb{R}^n \times \mathbb{R}^v \to \mathbb{R}^l, g : \mathbb{R}^n \times \mathbb{R}^v \to \mathbb{R}^m$ are the objective, equality and inequality constraints, respectively. We assume that $f$, $h$ and $g$ have continuous second derivatives. Finally, $\mathcal{R}(x,u)$ represents the PDE modeling the physical system.

A solution to (1.1) must satisfy the first-order necessary optimaltiy conditions [18]. These conditions are most easily expressed in terms of the Lagrangian, which is the scalar function defined below:

$$\mathcal{L}(x,u,\psi,s,\lambda_h,\lambda_g) = f(x,u) + \lambda_h^T h(x,u) + \lambda_g^T (g(x,u) - s) + \psi^T \mathcal{R}(x,u), \tag{1.2}$$

where $s \in \mathbb{R}^m$ are the so-called slack variables, and $\lambda_h \in \mathbb{R}^l$ and $\lambda_g \in \mathbb{R}^m$ are the Lagrangian multipliers for the equality and inequality constraints, respectively.

Using $\mathcal{L}$, the first-order necessary conditions, as known as the *Karush-Kuhn-*

*Tucker* (KKT) optimality conditions, for (1.1) can be expressed as

$$\begin{aligned}
\partial_x \mathcal{L} &= \partial_x f + \lambda_h^T \partial_x h + \lambda_g^T \partial_x g + \psi^T \partial_x \mathcal{R} = 0, \\
\partial_u \mathcal{L} &= \partial_u f + \lambda_h^T \partial_u h + \lambda_g^T \partial_u g + \psi^T \partial_u \mathcal{R} = 0, \\
\partial_\psi \mathcal{L} &= \mathcal{R} = 0, \\
\partial_{\lambda_h} \mathcal{L} &= h = 0, \\
\partial_{\lambda_g} \mathcal{L} &= g - s = 0, \\
-\mathsf{S}\mathsf{\Lambda}_g e &= 0, \\
s \geq 0, \quad \lambda_g &\leq 0.
\end{aligned} \tag{1.3}$$

For notational convenience, we have introduced $e = [1, 1, \ldots, 1]^T$ and the diagonal matrices

$$\mathsf{S} = \text{diag}\left(s_1, s_2, \ldots, s_m\right), \qquad \text{and} \qquad \mathsf{\Lambda}_g = \text{diag}\left(\lambda_{g1}, \lambda_{g2}, \ldots, \lambda_{gm}\right).$$

As mentioned earlier, the nonlinear system (1.3) can be solved in either the full space or the reduced space. Full-space methods [37], [38], [39] solve all the unknowns in (1.3) simultaneously. This results in a large nonlinear system whose size is more than double the number of PDE state variables (due to the adjoint). If Newton's method is used to solve (1.3), the resulting linear system is highly sparse, indefinite, and ill-conditioned. Nevertheless, effective iterative methods and preconditioners have been proposed for full-space methods [37], [38].

An advantage of the full-space approach is that, during the intermediate optimization iterations, the PDE state equation, $\mathcal{R} = 0$, and adjoint equation, $\partial_u \mathcal{L} = 0$, do not need to be solved exactly. This avoids the computational expense of tightly converging the PDE and adjoint equation residuals; however, this is also a potential disadvantage in practical engineering problems, because, if the optimization fails to converge, the intermediate solution may not be feasible with respect to the physics. Furthermore, for highly nonlinear PDEs, e.g. gas dynamics with shocks and boundary layers, practitioners have developed specialized globalization strategies that may be difficult to take advantage of in general-purpose full-space optimization algorithms. For theses reasons, no general purpose optimization libraries exist for full-space methods, to the best of our knowledge.

Reduced-Space algorithms for solving (1.3) treat the states $u$ and the adjoints $\psi$ as implicit functions of the design variables through $\mathcal{R}(x, u(x)) = 0$, and $\partial_u \mathcal{L} = 0$. Consequently, the reduced KKT conditions can be formulated as follows:

$$F(x, s, \lambda_h, \lambda_g) \equiv \begin{bmatrix} \nabla_x f + \lambda_h^T \nabla_x h + \lambda_g^T \nabla_x g \\ -\mathsf{S}\mathsf{\Lambda}_g e \\ h \\ g - s \end{bmatrix} = 0, \tag{1.4}$$

$$\text{subject to} \quad s_i \geq 0, \quad \text{and} \quad \lambda_{gi} \leq 0 \qquad \forall i = 1, 2, \ldots, m,$$

where the unknowns are $x^T, s^T, \lambda_h^T, \lambda_g^T$, and $F : \mathbb{R}^N \to \mathbb{R}^N, N = n + l + 2m$, is the vector-valued residual of the KKT conditions, excluding the inequalities on $s$ and $\lambda_g$.

The reduced-space approach to PDE-constrained optimization is attractive for a few reasons. First, it should be clear that the KKT system (1.4) is much smaller than (1.3). Second, reduced-space algorithms are more modular, since they can make direct use of existing PDE primal and adjont solvers. This modularity is one of the reasons that the reduced-space approach has remained the dominant approach in aerospace applications.

Of course, the reduced-space approach is not without difficulties. If conventional optimization algorithms are used to solve (1.4), then the aforementioned scaling issues arise, particularly the cost of evaluating the explicit constraint Jacobian. Instead, we need an algorithm that does not require the explicit constraint Jacobian.

### 1.4.2 Reduced-space Inexact-Newton Methods

One alternative to using conventional (matrix-based) optimization algorithms to solve the reduced-space KKT conditions (1.4) is to apply inexact-Newton methods [40], which are also know as truncated-Newton methods in the optimization literature [41].

To see how inexact-Newton methods can be used to solve 1.4, notice that, with the exception of the bounds on $s$ and $\lambda_g$, the KKT conditions (1.4) form a set of nonlinear algebraic equations, $F(q) = 0$, where $q \equiv (x^T, s^T, \lambda_h^T, \lambda_g^T)^T \in \mathbb{R}^N$ is the vector of unknowns. These equations can be solved, in principle, using Newton iterations of the form

$$(\nabla_q F)\Delta q^{(k)} = -F(q^{(k)}), \tag{1.5}$$

where $q^{(k)}$ is the solution at the $k$th iteration and $\Delta q^{(k)} = q^{(k+1)} - q^{(k)}$ is the solution update. Solving (1.5) exactly can be inefficient during early Newton iterates when the linear model is not a good approximation to $F(q) = 0$. Instead, truncated- and inexact-Newton methods find approximate solutions to (1.5) that, for example, satisfy the inexact-Newton condition

$$\left\|(\nabla_q F)\Delta q^{(k)} + F(q^{(k)})\right\| \leq \eta_k \left\|F(q^{(k)})\right\|, \tag{1.6}$$

for some parameter $\eta_k \in (0, 1)$.

There has been considerable success applying inexact-Newton methods to unconstrained optimization problems; see [41] and the references therein. On the other hand, inexact-Newton methods for general (nonconvex) constrained problems are much less common. Some notable exceptions include the efforts by Byrd and colleagues [42], [43] and by Heinkenschloss and Ridzal [44]; however, these algorithms make assumptions regarding the structure of the problem that favor full-space formulations, and our experience applying them to reduced-space PDE-constrainted optimization has been disappointing.

Applying Newton's method to (1.4), the KKT system, also called the primal-dual system, is obtained:

$$\begin{bmatrix} \nabla_{xx}\mathcal{L} & 0 & \nabla_x h^T & \nabla_x g^T \\ 0 & -\mathsf{\Lambda}_g & 0 & -\mathsf{S} \\ \nabla_x h & 0 & 0 & 0 \\ \nabla_x g & -\mathsf{I} & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_s \\ p_h \\ p_g \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L} \\ -\mathsf{S}\mathsf{\Lambda}_g e \\ h \\ g - s \end{bmatrix} \tag{1.7}$$

where $\mathcal{L}$ the Lagrangian is defined in (1.2), and $\mathsf{S}$ and $\mathsf{\Lambda}_g$ are as defined previously. A Newton-Krylov (NK) algorithm is a type of inexact-Newton method that solves (1.7) approximately using a Krylov iterative method, which only needs the matrix-vector product of the system matrix. The products can be formed in a matrix-free way by solving two second-adjoint systems; for details, see [45], [46] and the references therein. This is significant, because it means that NK algorithms do not require the constraint Jacobian (or Lagrangian Hessian) explicitly, unlike conventional optimization algorithms.

Prior to this work, the reduced-space Newton-Krylov method has been successfully applied to unconstrained problems and certain types of equality-constrained problems. In the case of unconstrained problems, a Newton-Conjugate-Gradient method can be used, in which the Steinhaug-Toint variant of CG is used to deal with nonconvex objectives; see, for example, [47], [48], [49], [50]. Reduced-space NK optimization algorithms have also shown promise for some types of equality-constrained problems, because, as already mentioned above, they do not require the constraint Jacobian to be form explicitly and, thus, avoid the scaling issue described earlier. For instance, [46] applied a matrix-free NK algorithm to a class of equality-constrained optimization problems that arise in multidisciplinary design optimization and would otherwise be intractable with conventional matrix-based algorithms.

### 1.4.3 Challenges in Using Reduced-space Newton-Krylov Methods

Motivated by its success in the unconstrained and equality-constrained cases, this thesis aims to extend the Newton Krylov methodology to more general, equality and inequality constrained problems. This extension of the reduced-space NK algorithm encounters two fundamental challenges that must be addressed.

**Nonconvexity:** The system $F(q) = 0$ does not distinguish between different types of stationary points, so Newton-type methods, including inexact-Newton-Krylov methods, may converge to local maximizers or saddle points. Conventional optimization algorithms often project onto the null-space of the (active) constraint Jacobian to detect directions of negative curvature and avoid undesirable stationary points; however, the null-space is not explicitly available for matrix-free inexact-Newton methods. Consequently, we must find a matrix-free approach to deal with nonconvexity.

**Preconditioning:** The number of iterations necessary to satisfy the inexact Newton condition (1.6) using a Krylov method is closely related to the condition number of the system. Unfortunately, it is well known that the primal-dual matrix $\nabla_q F$ is indefinite and highly ill-conditioned. A preconditioner is needed that is inexpensive to form, store, and apply. A general-purpose, inexpensive preconditioner is especially difficult to find in the reduced-space context, since approximations to $\nabla_q F$ are not readily available as they are in the full-space.

## 1.5 Contributions

This thesis proposes a matrix-free inexact-Newton-Krylov optimization algorithm that is specifically intended for large-scale, reduced-space PDE-constrained design problems. The primary contributions of this work are in addressing the challenges related to nonconvexity and matrix-free preconditioning.

The approach to addressing nonconvexity is to introduce a homotopy map that implicitly defines a solution curve that connects the solution to an easy problem to the solution of the desired problem. A predictor-corrector algorithm is proposed to follow the curve from the easy to the desired solution.

To address the conditioning of the KKT matrix, a low-rank approximation of the Schur complement of the KKT system is proposed. The low-rank approximation is computed by using the Lanczos method, which only involves matrix-vector products with the Schur complement. The matrix-vector products can be obtained using approximate state and adjoint solutions.

## 1.6 Thesis Outline

The remaining chapters in the thesis are structured as follows:

- Chapter 2 reviews the homotopy-based globalization and the homotopy map adopted in this work. Then it describes the predictor-corrector path-following algorithm that traces the homotopy zero curve. Finally, the proposed algorithm is verified and investigated using analytical problems.

- Chapter 3 is focused on describing the proposed matrix-free preconditioner. It begins by considering inequality constrained problems, and then generalizes the preconditioner to problems with both equality and inequality constraints. A synthetic quadratic problem with linear inequality constraints is used to investigate the effectiveness of the inequality preconditioner and the scalability performance of the algorithm.

- Chapter 4 presents the main numerical results. The chapter begins by describing the optimization environment Kona in which the proposed algorithms have been implemented. Subsequently, the chapter summarizes a state-of-the-art optimization algorithm (SNOPT) against which comparisons will be made. The predictor-corrector algorithm and the preconditioners are first tested on a subset of the CUTEr problems. Next, the proposed algorithms are applied to the stress-constrained mass minimization of a flat plate. Finally, we present the results of a three-dimensional aerodynamic shape optimization problem.

- Chapter 5 provides conclusions and some recommendations.

# CHAPTER 2
# HOMOTOPY-BASED GLOBALIZATION

## 2.1 Homotopy-based Globalization

Recall that Newton-Krylov (NK) root-finding algorithms cannot distinguish between (desired) local minimizers and other stationary points. Thus, the basic NK algorithm must be augmented with a globalization that avoids local maximizers and saddle points. This chapter describes one such globalization approach based on homotopy methods.

Homotopy methods are robust, numerically stable, and globally convergent methods for solving nonlinear algebraic equations; see, for example, [51] and [52]. These methods have been used to globalize nonlinear PDEs, including difficult computational aerodynamics problems [53], [54], [55], but globally convergent probability-one homotopy methods have also been successfully applied to solve engineering optimization problems [56]. Watson [57] reviewed and developed the general convergence theory of probability-one homotopies for nonlinear optimization problems, including unconstrained, bound-constrained, linear and nonlinear inequality constrained convex cases. He also discussed the extension of the theory to nonconvex problems, although the convergence theory for equality constraints remains an open problem. More recently, Huang *et al.* [58] transformed a general nonlinear optimization with equality and inequality into an inequality-only problem, and used a predictor-corrector method to track the homotopy interior-point map using the conjugate gradient method. While their method achieves global linear convergence under the normal cone condition, it is limited to convex objectives and constraint functions.

Portions of this chapter previously appeared as: P. Meng, A. Dener, J. Hicken, and G. Kennedy, "Matrix-free algorithm for reduced-space PDE-governed optimization with inequality constraints", unpublished.

### 2.1.1 An Example

Conceptually, the idea of homotopy methods is easy to understand. To find the solution of a difficult nonlinear equation, $F(q) = 0$, a homotopy map is constructed that relates the target problem to an easy-to-solve problem via a parameter. For example, a convex homotopy map $H : \mathbb{R}^N \times \mathbb{R}^N \times [0, 1) \to \mathbb{R}^N$ is given by

$$H(q, q_0, \mu) = (1 - \mu)F(q) + \mu G(q, q_0), \quad 0 \leq \mu \leq 1, \tag{2.1}$$

where $\mu$ is the homotopy parameter, and $G : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$ is chosen such that $G(q, q_0) = 0$ is easy to solve and has the solution $q = q_0$.

We will use a simple, unconstrained optimization example to illustrate the homotopy idea. Consider the problem

$$\min_x \quad f(x) = x^4 - x^2.$$

The first-order optimality condition for this problem is given by (identifying $q$ with $x$ here)

$$F(x) = \nabla_x f(x) = 2x(2x^2 - 1) = 0.$$

It is easy to see that there are three stationary points; a local maximizer at $x = 0$ and two local/global minimizers at $x = \pm 1/\sqrt{2}$. Newton's method may converge to any of these stationary points depending on the initial guess $x_0$, so we need some way to avoid the local maximizer at $x = 0$.

Now, consider the simple problem $\min_x \frac{1}{2}(x - x_0)^2$, whose first-order optimality is given by

$$G(x, x_0) = x - x_0 = 0.$$

This has the obvious solution $x = x_0$. We can take advantage of this simple optimization problem by constructing a convex homotopy that combines $F(x)$ and $G(x, x_0)$ as follows:

$$H(x, x_0, \mu) = (1 - \mu)F(x) + \mu G(x, x_0) = (1 - \mu)2x(2x^2 - 1) + \mu(x - x_0).$$

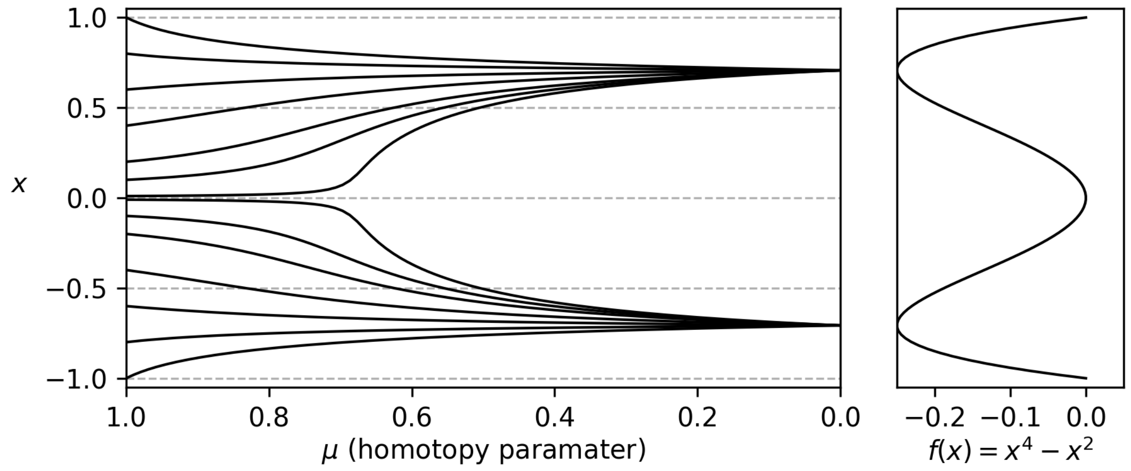


**Figure 2.1. Solution curves of $H(x, x_0, \mu) = 0$ (left side of figure) for different values of $x_0$. The paths begin at $\mu = 1$ and converge to the local minimizers at $\mu = 0$. The function to be minimized is plotted on the right-side of the figure**

Next, we trace out the set of solutions corresponding to $H(x, x_0, \mu) = 0$ from $\mu = 1$ to $\mu = 0$. Starting at $\mu = 1$ we have the solution $x = x_0$. If we change the value of $\mu$ slightly to $\mu = 1 - \Delta\mu$, then, for $\Delta\mu$ sufficiently small and by continuity, $x_0$ should remain in the basin of attraction for Newton's method applied to $H(x, x_0, 1-\Delta\mu) = 0$. The solution at $\mu = 1-\Delta\mu$ can then be used as an initial guess for the next value of $\mu$, and so on, until we reach $\mu = 0$. Example solution paths for this process are illustrated in Figure 2.1 starting from distinct $x_0$. Notice that all paths converge to the local minimizers, even those that begin near the maximizer $x = 0$.

### 2.1.2 Review of Convergence Theory for Homotopy Methods

The path-following process described above, while intuitive, is not guaranteed to succeed. In particular, it is not clear that $\nabla_q H$ remains non-singular along the path. A poor choice of $G(q)$ may produce a level set $H(q, q_0, \mu) = 0$ with an intersection, where $\nabla_q H$ is singluar and the path bifurcates. The Jacobian will also become rank deficient at so-called turning points, where following the path from $\mu = 1$ to $\mu = 0$ requires $\mu$ to increase at some point. Finally, the path may diverge before reaching $\mu = 0$.

Many of the potential issues with the path-following approach can be avoided if we place some conditions on the map $H$. These conditions are described in the theorem below, which has been adapted from [59] to the present context.

**Theorem 1.** *Assume the following conditions hold:*

- *$G : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$ is a $C^2$ map, and $q = q_0$ is the unique solution to $G(q, q_0) = 0$;*
- *$F : \mathbb{R}^N \to \mathbb{R}^N$, the first-order optimality residual given by (1.4), is a $C^2$ map;*
- *For the homotopy map $H$ defined by (2.1), the Jacobian*

$$\nabla H \equiv \begin{bmatrix} \nabla_q H & \nabla_{q_0} H & \nabla_\mu H \end{bmatrix}$$

*is full-rank on the zero set $X \equiv \{(q, q_0, \mu) | H(q, q_0, \mu) = 0\}$.*

*Then, for almost all $q_0 \in \mathbb{R}^N$, there exists a zero curve of $H$ starting from $q = q_0$ at $\mu = 1$ along which the Jacobian $\nabla H$ has full rank. Furthermore, if the set $X$ is bounded, then the path includes a point $(q, q_0, \mu) = (q^*, q_0, 0)$, i.e., where $H(q^*, q_0, 0) = F(q^*) = 0$. Finally, if the Jacobian $\nabla_q F$ is invertible at $q^*$, the path has finite arc length.*

Theorem 1 is a powerful result. It implies that, for almost all choices of $q_0$, there exists a path from $q = q_0$ at $\mu = 1$ to a point $q = q^*$ at $\mu = 0$ that satisfies $F(q^*) = 0$, and along this path the Jacobian is full-rank. The phrase *"almost all choices of $q_0$"* means that the set of points for which there is no path has measure zero.

The drawback of Theorem 1 is that its assumptions, with the exception of the first, are difficult to guarantee or verify in practice[2]. The second condition, which cannot be relaxed [59], implies that the objective and constraints are $C^3$. While this level of smoothness exists for many engineering problems, it is certainly not true in general. The third assumption means that $H$ is *transversal to zero* for each choice of $q_0$, which can be verified for simple problems, but may be difficult to determine

[2]The first condition requires $G$ to be sufficiently smooth and have $q_0$ as its only solution; as we show in Section 2.1.3, it is straightforward to construct such a map.

for complex engineering design problems. Finally, as Watson points out in [59], the assumptions on the boundedness of the path and the invertibility of $\nabla_q F$ at $q^*$ are the most difficult to verify, since, taken together with the other conditions, they imply the exsistence of a solution to $F(q) = 0$.

Despite the potential difficulty of guaranteeing the assumptions of Theorem 1, the theorem hints at the robustness of the homotopy approach, and this is corroborated by our experience.

### 2.1.3 Homotopy Map for Constrained Optimization

For this work we use the convex homotopy defined by (2.1), therefore we need only define $G(q, q_0)$. Similar to the unconstrained case, we could use a map of the form

$$G(q, q_0) = q - q_0,$$

where $q_0 = \begin{bmatrix} x_0^T & s_0^T & \lambda_{h0}^T & \lambda_{g0}^T \end{bmatrix}^T$. However, this map produces a positive-definite (diagonal) Jacobian $\nabla_q G$, which would be inconsistent with the inertia of the Jacobian $\nabla_q F$ at a local solution to (1.4) [18]. Instead, we adopt the map

$$G(q, q_0) \equiv \begin{bmatrix} (x - x_0)^T & (s - s_0)^T & -\lambda_h^T & -\lambda_g^T \end{bmatrix}^T,$$

for which $q_0 = \begin{bmatrix} x_0^T & s_0^T & 0^T & 0^T \end{bmatrix}$. We note that $G(q, q_0)$ satisfies the requirements of Theorem 1.

For future reference, we restate the homotopy map that we use for solving the first-order necessary conditions in this work:

$$H(q, q_0, \mu) = (1 - \mu) \begin{bmatrix} \nabla_x f(x) + \lambda_h^T \nabla_x h(x) + \lambda_g^T \nabla_x g(x) \\ -\mathsf{S}\Lambda_g e \\ h(x) \\ g(x) - s \end{bmatrix} + \mu \begin{bmatrix} x - x_0 \\ s - s_0 \\ -\lambda_h \\ -\lambda_g \end{bmatrix}. \tag{2.2}$$

**Remark 1.** *The homotopy map* (2.2) *can be used to identify stationary points of $F(q)$, but it cannot, on its own, ensure that $s_i \geq 0$ and $\lambda_i \leq 0$. We use safeguards to ensure the correct signs for the slacks and inequality multipliers; see Section 2.2.3.*

The homotopy map (2.2) is favorable in the context of optimization for several reasons. First, the term $\mu(x-x_0)$ helps to address nonconvexity by adding a positive diagonal matrix to the Lagrangian Hessian during the early stages of convergence. Furthermore, the term $\mu(s-s_0)$ with $s_0 > 0$ generates a path that remains feasible with respect to the slack boundaries $s > 0$. Finally, the terms $-\mu\lambda_h$ and $-\mu\lambda_g$ improve the conditioning of the KKT matrix; even when the constraint Jacobian is rank deficient the homotopy map will remain invertible for $\mu > 0$.

## 2.2 Predictor-Corrector Path-following Algorithm

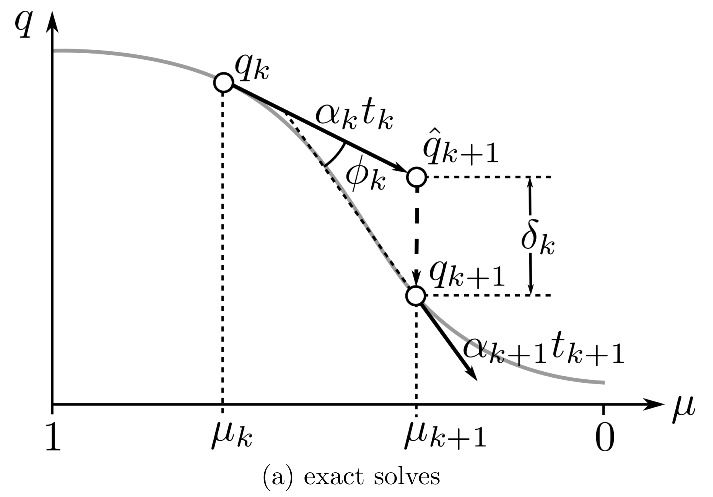
The theory presented in Section 2.1 tells us when a path exists between $q_0$ and a solution to $F(q)$, but it does not tell us how to traverse such a path in an efficient manner. In this section we describe the modified predictor-corrector algorithm [51] used to trace the zero-curve of the homotopy map.

### 2.2.1 Overview

Figure 2.2a illustrates the predictor and corrector phases at iteration $k$ when exact linear and nonlinear solutions are possible. The iteration begins by computing a tangent to the path, $q'_{k+1}$, and using this tangent to predict the next point on the path. Subsequently, the homotopy parameter $\mu$ is fixed and a correction is found that gives a root of the homotopy. This cycle is repeated until $\mu < \epsilon_\mu$, where $\epsilon_\mu > 0$ is a specificed tolerance. Once $\mu$ is sufficiently close to zero an approximate solution of the primal-dual system is recovered.

**Remark 2.** *We use a default value of $\epsilon_\mu = 10^{-9}$ in our algorithm, which we have found works well for most problems; however, for difficult problems, e.g., with rank-deficient constraint Jacobians, it may be necessary to use larger thresholds. For example, we use $\epsilon_\mu = 10^{-6}$ in the structural optimization problem presented in Chapter 4.3.*

In practice, the linear and nonlinear subproblems cannot be solved exactly. Thus, the true situation is more like that in Figure 2.2b, which depicts the path-following algorithm when inexact solutions are obtained. These inexact solutions



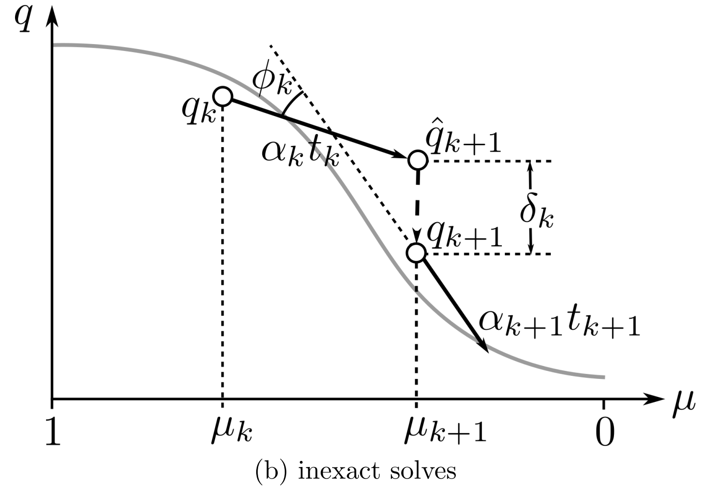


**Figure 2.2. Figure 2.2a illustrates the predictor-corrector algorithm in the case that exact solves are used, while Figure 2.2b depicts the situation when inexact solves are used**

are discussed in the following high-level description of the predictor and corrector phases.

**Predictor:** During the predictor phase we first need to compute an approximate tangent direction $q' = dq/d\mu$, which is defined by taking the total derivative of $H(q, q_0, \mu) = 0$ with respect to $\mu$, i.e. by applying the implicit function theorem:

$$\left(\nabla_q H\right)_k q'_k = -\nabla_\mu H_k = F(q_k) - G(q_k, q_0), \tag{2.3}$$

where $\nabla_q H$ and $\nabla_\mu H$ are evaluated at $q_k$, the previous homotopy iterate.

In practice, we solve (2.3) inexactly using a preconditioned Krylov iterative method. That is, we find a $q'_k$ that satisfies

$$\|\left(\nabla_q H\right)_k q'_k - F(q_k) + G(q_k, q_0)\| \leq \tau \|F(q_k) - G(q_k, q_0)\|,$$

where $\tau \in [0, 1)$ is the desired relative tolerance. Further details on the inexact solution of (2.3) are provided in Chapter 3.

After (inexactly) solving (2.3) for $q'_k$, the predictor step is given by

$$\begin{bmatrix} \hat{q}_{k+1} \\ \mu_{k+1} \end{bmatrix} = \begin{bmatrix} q_k \\ \mu_k \end{bmatrix} + \alpha_k t_k, \tag{2.4}$$

where $\alpha_k$ is the step length taken along the tangent direction at iteration $k$ (see Section 2.2.2), and $t_k$ is the normalized tangent given by

$$t_k \equiv \frac{1}{\sqrt{\|q'_k\|^2 + 1}} \begin{bmatrix} -q'_k \\ -1 \end{bmatrix}. \tag{2.5}$$

Note that the negative signs in the definition of $t_k$ account for decreasing $\mu$ as the path moves from $\mu = 1$ to $\mu < \epsilon_\mu$.

**Corrector:** In the corrector phase, we fix the homotopy parameter at $\mu = \mu_{k+1}$ based on the predictor, and use a Newton-Krylov method to inexactly solve $H(q_{k+1}, q_0, \mu_{k+1}) = 0$. This has the effect of "correcting" $\hat{q}_{k+1}$ to be closer to the path. More precisely, we seek $q_{k+1}$ that reduces the relative residual below some tolerance:

$$\|H(q_{k+1}, q_0, \mu_{k+1})\| \leq \epsilon_H \|H(\hat{q}_{k+1}, q_0, \mu_{k+1})\|. \tag{2.6}$$

A loose relative tolerance of $\epsilon_H \in [0.1, 0.5]$ is used for homotopy parameter values $\mu > 0$ to avoid oversolving the corrector step. Once the algorithm reaches $\mu \leq \epsilon_\mu$, the tolerance is tightened to the user-specified value for the first-order necessary conditions.

At each Newton subiteration within the corrector, we must (inexactly) solve a linear system of the form

$$\left(\nabla_q H\right)_{k+1} \Delta q_{k+1} = -H, \tag{2.7}$$

for $\Delta q_{k+1}$, where $(\nabla_q H)_{k+1}$ and $H_{k+1}$ are evaluated at $\mu_{k+1}$ and the current estimate for $q_{k+1}$. The Jacobian, $\nabla_q H$, that appears in (2.7) also appears in (2.3) during the predictor phase; again, further details regarding the solution of the linear systems that involve $\nabla_q H$ are provided in Chapter 3.

**Remark 3.** *The above predictor-corrector approach is a classical embedding method [51], since the solution path is assumed to be parameterized by $\mu$; that is, the path has the form $(q(\mu), \mu)$. This type of method cannot handle folds, or turning points, where the path reverses direction with respect to $\mu$ and the Jacobian $\nabla_q H$ becomes singular. Problems for which such turning points arise can be handled using more advanced predictor-corrector algorithms; see, for example, [60].*

### 2.2.2 Adaptive Step-Size Control

The step length $\alpha$ along the tangent direction is calculated using the asymptotic expansion method [61], which we briefly review here.

At the first iteration of the predictor-corrector algorithm when $\mu = 1$, a conservative step length is used, e.g., $\alpha_0 = 0.05$. Subsquent step lengths are then determined adaptively using a scaling factor and a couple safeguards on the maximum step size:

$$\alpha_k = \min\left(\sqrt{\|q_k'\|^2 + 1}\Delta\mu_{\max}, \alpha_{\max}, \alpha_{k-1}/\zeta_k\right),$$

where $\Delta\mu_{\max}$ is the maximum allowable change in $\mu$, $\alpha_{\max}$ is the allowable step size permitted by the fraction-to-the-boundary rule (see Section 2.2.3), and the scaling factor for $\alpha_{k-1}$ is given by

$$\zeta_k \equiv \max\left(\sqrt{\delta_k/\delta_{\text{targ}}}, \phi_k/\phi_{\text{targ}}\right).$$

The scaling factor $\zeta_k$ is controlled by two quantities that measure how nonlinear the path is, namely the previous corrector update size, $\delta_k$, and the angle between successive tangents, $\phi_k$. Referring to Figure 2.2, the corrector update size is the magnitude of the difference between the predictor step and the corrector step:

$$\delta_k \equiv \|q_k - \hat{q}_k\|. \tag{2.8}$$

A large value of $\delta_k$ indicates that the linear predictor does not approximate the path well, so a smaller step size is warranted. The angle between the tangents is

$$\phi_k \equiv \arccos\left(t_{k+1}^T t_k\right). \tag{2.9}$$

The angle $\phi_k$ is a measure of the curvature in the path, so a relatively large value of $\phi_k$ also suggests that $\alpha_k$ should be reduced. For locally linear paths both $\delta_k$ and $\phi_k$ are zero, and the scaling factor will lead to an unbounded $\alpha_k$; hence the need for the maximum allowable step $\alpha_{\max}$.

**Remark 4.** *While the adaptive step-size control is automated, performance depends on the parameters $\alpha_0$, $\Delta\mu_{\max}$, $\delta_{targ}$ and $\phi_{targ}$. The optimal choice for these parameters is problem dependent.*

### 2.2.3 Safeguards On The Slacks and Inequality Multipliers

One of the differences between solving (1.4) and solving a generic nonlinear system of equations is that the slacks and inequality multipliers must remain nonnegative and nonpositive, respectively. To cope with this additional set of requirements, the following safeguards are implemented in the predictor-corrector scheme.

- During the predictor phase, the maximum allowable step size is bounded using a fraction-to-the-boundary-like rule [18] defined below.

$$\alpha_{\max} = \max\left\{\alpha \in (0,1] \mid s + \alpha s' \geq \tau_s.\right\}. \tag{2.10}$$

  where $\tau_s = 10^{-6}$. The fraction-to-the-boundary rule we use is slightly different than [18] in that we use a fixed absolute value $\tau_s$ to define the boundary instead of a fixed fraction boundary.

The maximum step size is enforced for all variables, including the design variables $x$. We have found that synchronizing the step size across the design, slack and multipliers improves the performance of the algorithm.

- In addition, the following clipping rule is applied to the slack variable:

$$s \leftarrow \max(s, \tau_s). \tag{2.11}$$

This clipping rule is applied at two points in the algorithm: at the very beginning to the initial slack $s_0$, and at the last point of the corrector phase.

- At the end of both the predictor and corrector phases, the inequality multipliers are clipped to ensure they remain non-positive:

$$\lambda_g \leftarrow \min(\lambda_g, 0) \tag{2.12}$$

where the min function in the above expression is to be interpreted elementwise.

While the slacks and inequality multipliers both have similar bound constraints, their treatment is slightly different in the above safeguards. Our motivation for bounding the slacks away from zero, in both the predictor and corrector phases, is to avoid severe ill-conditioning in the system matrix and its preconditioner; see Section 3.3 for further details. The inequality multipliers, in contrast, do not lead to conditioning problems if they vanish, so we can clip the multipliers to zero.

### 2.2.4 Algorithm Summary

With most of its elements described, we summarize the predictor-corrector method in Algorithm 1. The solution of the tangent step, line 19, and the solution of the Newton step, line 14, are the only components of the algorithm that require further explanation. Note that the initial tangent step on line 4 is a linear system that is trivial to solve, because $\nabla_q H$ is diagonal when $\mu = 1$.

**Algorithm 1:** Inexact-Newton predictor-corrector algorithm for reduced-space PDE-Constrained optimization.

```
Parameters: $K_{\max}$, $J_{\max}$, $\epsilon_F$, $\tau$, $\epsilon_H$, $\alpha_0$, $\delta_{\text{targ}}$, $\phi_{\text{targ}}$, $\Delta\mu_{\max}$
Input      : $x_0$, $s_0$
Output     : $q^* = \begin{bmatrix} x^{*T} & s^{*T} & \lambda_h^{*T} & \lambda_g^{*T} \end{bmatrix}^T$, the solution of (1.4)
Clip $s_0$ if necessary: $s_0 \leftarrow \max(s_0, \tau_s)$
Set $q_0 = \begin{bmatrix} x_0^T & s_0^T & 0^T & 0^T \end{bmatrix}$, $q \leftarrow q_0$
compute state $u$ and adjoint $\psi$ at $q_0$
4 solve $\left(\nabla_q H\right)_k q'_k = -\nabla_\mu H_k$ for $q'_k$
compute the normalized tangent direction $t_k$ using (2.5)
for $k = 0, 1, 2, \ldots, K_{\max}$ do
    compute $\alpha_k (\alpha_0 \text{ if } k \equiv 0) = \min\left(\sqrt{\|q'_k\|^2 + 1}\Delta\mu_{\max}, \alpha_{\max}, \alpha_{k-1}/\zeta_k\right)$
    use (2.4) to update $\hat{q}_{k+1}$ and $\mu_{k+1}$
    clip $\lambda_{g,k+1}$ if necessary: $\lambda_{g,k+1} = \min(\lambda_{g,k+1}, 0)$
    update state $u$ and adjoint $\psi$ at $\hat{q}_{k+1}$
    set $q_{k+1} \leftarrow \hat{q}_{k+1}$
    for $j = 0, 1, 2, \ldots, J_{\max}$ do
        if $\|H(q_{k+1}, q_0, \mu_{k+1})\| \leq \epsilon_H \|H(\hat{q}_{k+1}, q_0, \mu_{k+1})\|$ then break
14      inexactly solve $\left(\nabla_q H\right)_{k+1} \Delta q_{k+1} = -H_{k+1}$ for $\Delta q_{k+1}$
        $q_{k+1} \leftarrow q_{k+1} + \Delta q_{k+1}$
        update state $u$ and adjoint $\psi$ at $q_{k+1}$
    end
    if $\mu_k < \epsilon_\mu$ then return $q_k$
19  inexactly solve $\left(\nabla_q H\right)_k q'_k = -\nabla_\mu H_k$ for $q'_k$ to required tolerance $\tau$
    compute the normalized tangent direction $t_k$ using (2.5)
    update $\delta_k$ and $\phi_k$ according to (2.8) and (2.9)
    compute the step scaling factor: $\zeta_k = \max\left(\sqrt{\delta_k/\delta_{\text{targ}}}, \phi_k/\phi_{\text{targ}}\right)$
    clip if necessary: $s_{k+1} \leftarrow \max(s_{k+1}, \tau_s)$ and $\lambda_{k+1} \leftarrow \min(\lambda_{k+1}, 0)$
end
```

## 2.3 Globalization Numerical Experiments

In theory, Newton-based methods can successfully solve the first-order necessary optimality conditions, but cannot guarantee that the solution also satisfies the second-order sufficient optimality conditions. However, the added homotopy term in the homotopy map (2.2) functions like a regularization in the optimization problem, and thus provides the proposed algorithm the capacity to handle a certain amount of nonconvexity. To investigate this capacity, two numerical examples are tested: the first one is a 3D sphere constrained problem with a linear objective, and the second example is an indefinite quadratic problem with bound constraints. The goal is to check whether the proposed algorithm can avoid converging to local maximizers and saddle points.

### 2.3.1 Sphere Problem Description

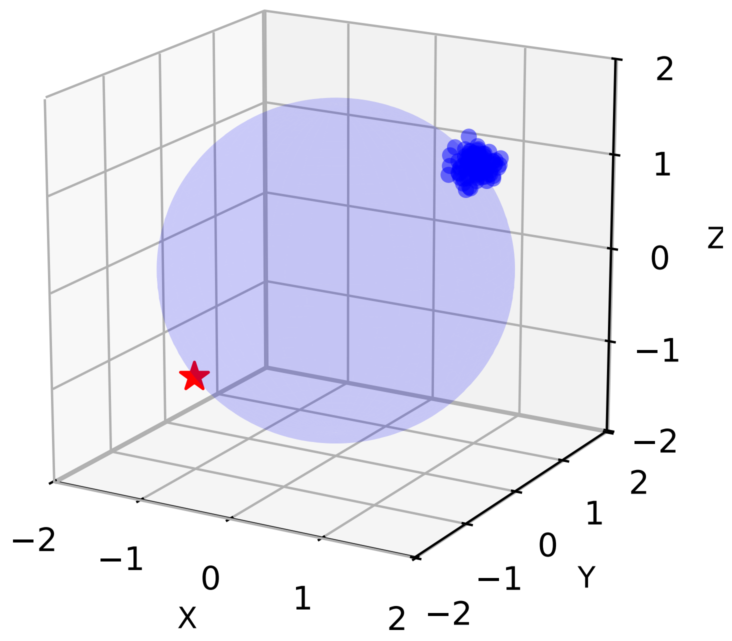
The first test problem to verify the homotopy-based globalization has a linear objective and a feasible domain that is the inside of a sphere:

$$\begin{aligned} \min_{x,y,z} \quad & x + y + z \\ \text{subject to} \quad & x^2 + y^2 + z^2 \leq 3. \end{aligned}$$

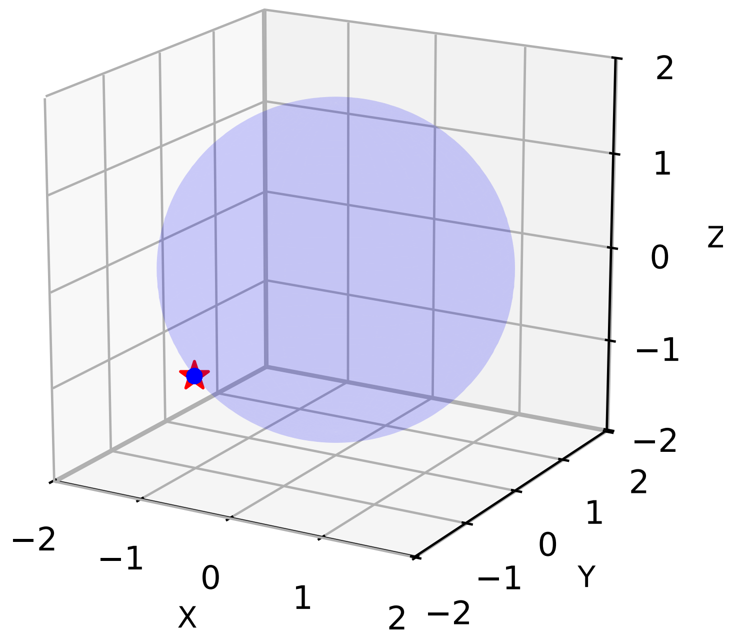
The solution to this problem is at $(-1, -1, -1)^T$. There is also a local maximum point at $(1, 1, 1)^T$, where the first-order optimality condition (KKT condition) is satisfied but not the second-order optimality conditions. The presence of this local maximizer provides a test for the homotopy-based globalization algorithm.

### 2.3.2 Sphere Problem Results

To demonstrate that the algorithm can bypass the local maximizer, 100 random starting points around the point $(1, 1, 1)^T$ where generating by adding Gaussian perturbations, $\Delta x_i \sim \mathcal{N}(0, 0.1^2)$, to each coordinate. Figure 2.3a shows 100 random initial points as blue circles and the exact solution as a red star. Figure 2.3b shows the final iterates as blue circles and the exact solution as a red star. As can be seen, even when the 100 starting points are near the local maximizer, they all converge to the minimum point.

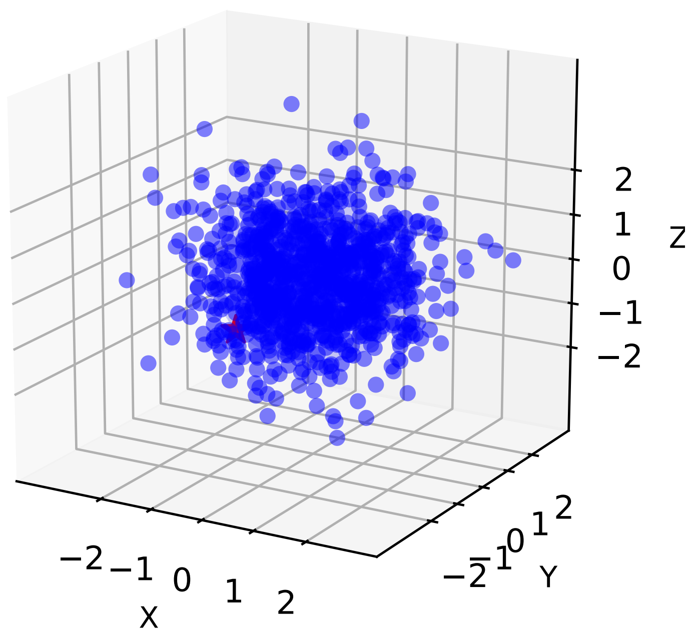


(a) 100 initial points, $x_0$, around the local maximizer.

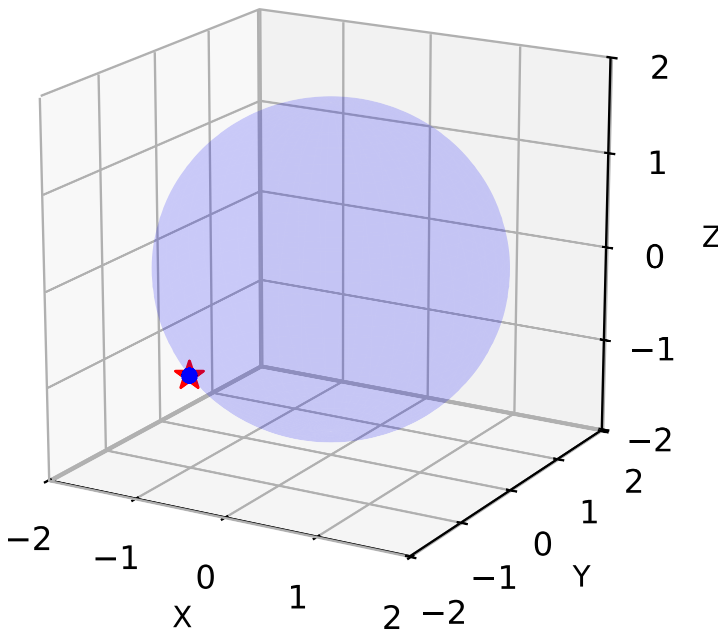


(b) 100 converged points, $x^*$.

**Figure 2.3. 100 initial and converged points**



(a) 1000 initial points, $x_0$, distributed about the origin.



(b) 1000 converged points, $x^*$.

**Figure 2.4. 1000 initial and converged points**

To further test the robustness of the globalization, we seeded the algorithm with 1000 initial guesses whose $x_0$ were drawn from a standard normal distribution. Figure 2.4a shows a scatter plot of these 1000 points. All $x_0$ values converge to the local minimizer, as shown in Figure 2.4b.

### 2.3.3 Non-convex Problem Description

For our second numerical experiment, we consider the following simple 100-dimensional non-convex optimization problem:

$$\begin{aligned} \min_{x \in R^{100}} \quad & \frac{1}{2} x^T \mathsf{Q} x \\ \text{subject to} \quad & -1 \leq x_i \leq 1 \qquad \forall i = 1, 2, \ldots, 100, \end{aligned}$$

where Q is a diagonal matrix whose entries are $-1$ or 1 with equal probability, i.e. according to a Rademacher distribution. As the objective function can be separated into individual dimensions, it is easy to see that valid minimizers for this problem have the following dependence on Q:

$$\begin{aligned} x_i &= 0 \quad \text{if } \mathsf{Q}_{ii} = 1, \\ x_i &= \pm 1 \quad \text{if } \mathsf{Q}_{ii} = -1. \end{aligned} \tag{2.13}$$

We would like to investigate the ability of the algorithm to avoid stationary points that are not local minimizers, e.g. $x_i = 0$ when $\mathsf{Q}_{ii} = -1$ and $x_i = \pm 1$ when $\mathsf{Q}_{ii} = 1$. Note that when $\mathsf{Q}_{ii} = 1$, the upper and lower bound $x_i = \pm 1$ are individual maximums where the gradient of the Lagrangian is zero in that dimension.

### 2.3.4 Non-convex Problem Results

We ran the optimization algorithm on the non-conex problem with 1000 randomly generated Q matrices. For each case, the arrangement of 1 and $-1$ in $\mathsf{Q}$ was randomly generated as described above. The initial point $x_0$ was also generated randomly with uniform probability in the domain $\Omega = \{x \in \mathbb{R}^{100} \mid -2 \leq x_i \leq 2\}$. We call a solution successful if its pattern is exact as given by (2.13). Table 2.1 lists the success rate for different combinations of $\epsilon_{\text{krylov}}$ and $\tau$, $\epsilon_H$. Note that $\epsilon_{\text{krylov}}$

is the tolerance of the Krylov solver introduced in Chapter 3.1; $\tau$ and $\epsilon_H$ are the tolerances for the predictor and the corrector phases as explained in Section 2.2.1.

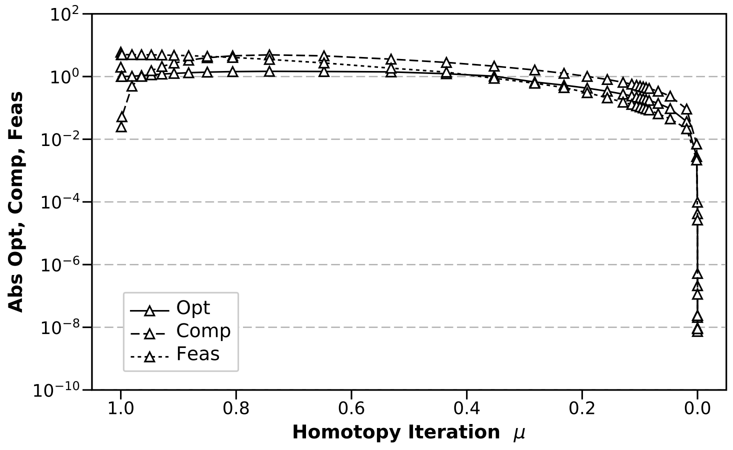
**Table 2.1. Success rate with different parameters**

| | | $\epsilon_{\text{krylov}}$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ |
| $\tau$ and $\epsilon_H$ | $10^{-1}$ | 51% | 90.0% | 94.2% | 94.6% | 93.9% | 93.8% |
| | $10^{-2}$ | 47.2% | 93.1% | 94.4% | 93.9% | 94.2% | 94.5% |

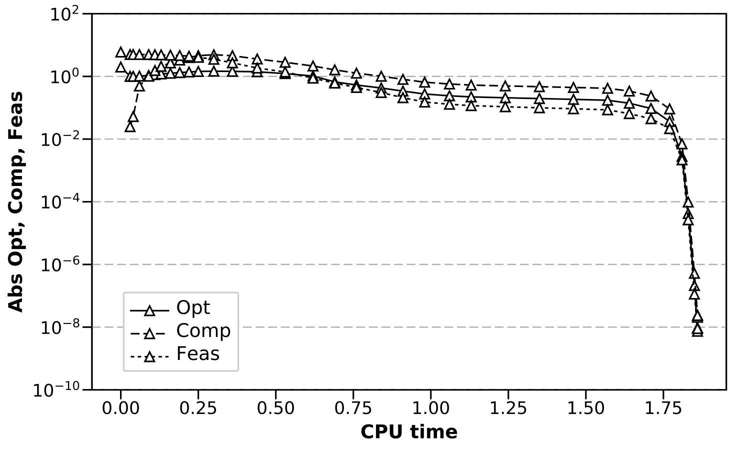
As can be seen from Table 2.1, the robustness of the algorithm for handling non-convexity is obviously impacted by the tolerances, particularly $\epsilon_{\text{krylov}}$. The results show that $\epsilon_{\text{krylov}}$ has to be below $10^{-2}$ for effective non-convexity handling, while $\tau$ and $\epsilon_H$ plays a smaller role than $\epsilon_{\text{krylov}}$.

Admittedly, this nonconvex test problem is relatively simple, with no additional complications such as bad-scaling or ill-conditioning. The focus here is solely to see whether the added regularization from the homotopy term can help Newton's method bypass local maximizers and saddle points. The results show that the optimization method can start from infeasible points, and can handle nonconvexity provided the tolerances are sufficiently tight. Further work is needed to automatically detect nonconvex problems and adjust the tolerances as necessary.

We conclude this chapter by presenting typical optimization convergence plots for this problem; see Figure 2.5. Figure 2.5a shows the absolute optimality, complementarity and feasibility at each homotopy iteration of different $\mu$. Note that, for simplicity, only the predictor points are displayed when $\mu \geq \epsilon_\mu$, while the corrector points are displayed for $\mu < \epsilon_\mu$. Figure 2.5b shows the convergence criteria with respect to CPU time. These convergence plots indicate that, at least asymptotically, superlinear convergence is maintained by the algorithm.



(a) Convergence criteria vs. $\mu$



(b) Convergence criteria vs. CPU time

**Figure 2.5. Convergence history from one of the nonconvex cases. This history is typical of 1000 cases considered**

# CHAPTER 3
# MATRIX-FREE SOLUTION OF THE LINEAR SYSTEMS

During the execution of Algorithm 1, the tangent linear system and Newton update are solved many times. Therefore, in order for the predictor-corrector algorithm to be competitive, these systems must be (inexactly) solved with high efficiency. Both systems, (2.3) and (2.7), take the form

$$(\nabla_q H)x = b, \tag{3.1}$$

where $b \in \mathbb{R}^N$ is either $F - G$, in the case (2.3) for the predictor step or $-H$ in the case of (2.7) for a corrector step. We inexactly solve these systems using a preconditioned Krylov-iterative method. The primary challenge with this approach, as discussed in Chapter 1, is that the preconditioner itself must be matrix free.

To address this requirement, a matrix-free preconditioner has been proposed as part of this thesis. The bulk of this chapter focuses on describing the matrix-free preconditioner and exercising it on some numerical experiments. However, we begin with a brief review of the Krylov iterative solver that needs the preconditioner.

## 3.1 Krylov Iterative Solver

We use the flexible generalized minimal residual method, FGMRES [62], to solve (3.1). One advantage of FGMRES, compared to most Krylov-based methods, is that it permits nonstationary preconditioners that vary from one Krylov iteration to the next. While we do not take advantage of nonstationary preconditioners in this work, we have found this flexibility invaluable in the solution of multidisciplinary optimization problems [46], [63].

To find an approximate solution to (3.1), FGMRES orthonormalizes a sequence of matrix-vector products using a generalized form of Arnoldi's method [64]. Starting with $v_1 = b/\|b\|$, the generalized Arnoldi's method produces the following relation at the $i$th iteration:

$$(\nabla_q H)\mathsf{Z}_i = \mathsf{V}_{i+1}\bar{\mathsf{H}}_i, \tag{3.2}$$

where $\mathsf{V}_{i+1} \in \mathbb{R}^{N\times(i+1)}$ has orthogonal columns and $\bar{\mathsf{H}}_i \in \mathbb{R}^{(i+1)\times i}$ is upper Hessenberg. The vectors in $\mathsf{Z}_i \in \mathbb{R}^{N\times i}$ form the subspace from which the approximate solution to (3.1) is drawn (see below). These vectors are related to the vectors in $\mathsf{V}_{i+1}$ by

$$z_j = P_j(v_j), \qquad \forall j = 1, 2, \ldots, i,$$

where $P_j(\cdot)$ denotes the preconditioning operation at iteration $j$.

The FGMRES solution is given by $x_i = \mathsf{Z}_i y_i$, where $y_i$ is chosen to minimize the 2-norm of the residual, $r_i \equiv b - (\nabla_q H)x_i = b - (\nabla_q H)\mathsf{Z}_i y_i$. The solution to this minimization problem is

$$\begin{aligned} y_i = \underset{y\in\mathbb{R}^i}{\operatorname{argmin}}\|b - (\nabla_q H)\mathsf{Z}_i y\| &= \underset{y\in\mathbb{R}^i}{\operatorname{argmin}}\|\mathsf{V}_{i+1}(\|b\|e_1 - \bar{\mathsf{H}}_i y)\| \\ &= \underset{y\in\mathbb{R}^i}{\operatorname{argmin}}\|\|b\|e_1 - \bar{\mathsf{H}}_i y\|, \qquad (\text{since } \mathsf{V}_{i+1}^T\mathsf{V}_{i+1} = \mathsf{I}) \end{aligned}$$

where $e_1 \in \mathbb{R}^{i+1}$ is the first column of the $(i+1)\times(i+1)$ identity. The least-squares minimization problem on the final line is inexpensive to solve in our applications, since $i$ is usually less than 100.

Like most Krylov iterative methods, the FGMRES algorithm described above does not require the Jacobian $(\nabla_q H)$ explicitly. From the user's perspective, the algorithm only requires matrix-vector products and preconditioning operations. In this work, matrix-vector products involving $(\nabla_q H)$ are computed using second-order adjoints [65], [45]. Briefly, each product with $(\nabla_q H)$ requires the solution of two discretized linear PDEs: a linear forward problem and a linear adjoint problem. See [66] for further details regarding second-order adjoints in the context of reduced-space problems with state constraints. The second required operation, preconditioning, is described in subsection 3.3.

## 3.2 Review on Preconditioners for Optimization

Using Krylov methods to solve the linear systems saves the effort of computing and storing the constraint Jacobian and Lagrangian Hessian, but the convergence rate of Krylov solvers depends on the distribution of the system's eigenvalues. If the eigenvalues are clustered in a small radius, the convergence rate is generally better, and poor convergence often arises when the ratio of the largest to the smallest eigenvalue modulus is large, e.g. $10^5$ to $10^9$.

In the presence of ill-conditioning, a nonsingular matrix can transform the linear system to a better conditioned one with the same solutions. This nonsingular matrix is called the preconditioner, and it exists in the form of mathematical operations in matrix-free methods. A preconditioner is an operator that is designed to cluster a system's eigenvalues; it usually does this indirectly by approximating the action of the matrix inverse and should be inexpensive to apply. When the preconditioner is stationary, it can be represented as a matrix $P \in \mathbb{R}^{N \times N}$. Thus, in the present case we would like

$$P_j(u) \approx (\nabla_q H)^{-1} u$$

in some sense where $u \in \mathbb{R}^N$ is arbitrary.

Many general preconditioners have been developed for saddle-point problems [67], such as incomplete LU factorizations [68], [69], [64], incomplete null-space factorization [70], [71], and approximate Schur complement decompositions [72], [73]. These preconditioners all involve direct operations to the entries of the system matrix, which would be infeasible for matrix-free methods. Furthermore, the diagonal entries of the system matrix in interior-point type optimizations could approach zero or infinity for inequality constraints, causing linear algebraic error in direct factorization methods.

In addition, many specialized preconditioners have been investigated for full-space PDE-constrained optimization; see, for example, [74]. However, most full-space PDE-constrained optimization preconditioners also rely on the availability of a matrix-based preconditioner for the state Jacobian. There is no analogous matrix-based preconditioners for the total Jacobian $\nabla_x g$ in the reduced-space, and, as explained in Chapter 1, forming $\nabla_x g$ explicitly is not feasible. In the making of this thesis, a Uzawa type preconditioner [75] has been investigated but without success. The preconditioner proposed in this thesis is also based on the approximate Schur complement decompositions.

## 3.3 Matrix-free Preconditioner

As $\mu$ approaches zero, the system (3.1) becomes an increasingly ill-conditioned saddle-point problem. Consequently, solving this problem with an iterative Krylov solver, like FGMRES, requires an effective preconditioner. One of the primary contributions of this work is a matrix-free[3] preconditioner for reduced-space PDE-constrained optimization with state-based constraints.

To motivate our preconditioner, we begin by examining the exact Jacobian $\nabla_q H$ in (2.3) and (2.7):

$$\begin{aligned}
\nabla_q H &= (1-\mu)\nabla_q F(q) + \mu \nabla_q G(q, q_0) \\
&= (1-\mu)\begin{bmatrix} \nabla_{xx}\mathcal{L} & \mathbf{0} & \nabla_x h^T & \nabla_x g^T \\ \mathbf{0} & -\mathsf{\Lambda}_g & \mathbf{0} & -\mathsf{S} \\ \nabla_x h & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \nabla_x g & -\mathsf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} + \mu \begin{bmatrix} \mathsf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathsf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathsf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathsf{I} \end{bmatrix} \\
&= \begin{bmatrix} \mathsf{W}_\mu & 0 & \mathsf{A}_{h,\mu}^T & \mathsf{A}_{g,\mu}^T \\ 0 & -\mathsf{\Lambda}_\mu & 0 & -\mathsf{S}_\mu \\ \mathsf{A}_{h,\mu} & 0 & -\mu\mathsf{I} & 0 \\ \mathsf{A}_{g,\mu} & -(1-\mu)\mathsf{I} & 0 & -\mu\mathsf{I} \end{bmatrix}
\end{aligned} \tag{3.3}$$

where $\mathsf{I}$ is the identity matrix, whose size can be inferred from the context; and the

[3]In this context, matrix-free means that we do not require a matrix whose size is the same size as $\nabla_x g$; however, we do use low-rank matrices.

sub-Jacobians are defined by

$$\mathsf{W}_\mu \equiv (1-\mu)\left[\nabla_x^2 f + \lambda_h^T \nabla_x^2 h + \lambda_g^T \nabla_x^2 g\right] + \mu \mathsf{I}, \qquad \mathsf{A}_{h,\mu} \equiv (1-\mu)\nabla_x h,$$
$$\mathsf{A}_{g,\mu} \equiv (1-\mu)\nabla_x g, \qquad \mathsf{S}_\mu \equiv (1-\mu)\mathsf{S}, \qquad \mathsf{\Lambda}_\mu \equiv (1-\mu)\mathsf{\Lambda}_g - \mu \mathsf{I}.$$

In the ideal case, a preconditioner application corresponds to solving the linear system

$$(\nabla_q H)v = u \tag{3.4}$$

for $v \in \mathbb{R}^N$ given an arbitrary $u \in \mathbb{R}^N$, where

$$u = \begin{bmatrix} u_x^T & u_s^T & u_h^T & u_g^T \end{bmatrix},$$
$$v = \begin{bmatrix} v_x^T & v_s^T & v_h^T & v_g^T \end{bmatrix},$$

and $u_x, v_x \in \mathbb{R}^n$, $u_s, v_s \in \mathbb{R}^m$, $u_h, v_h \in \mathbb{R}^l$, and $u_g, v_g \in \mathbb{R}^m$. To simplify the presentation, we will first consider the case when only inequality constraints are present, in which case the third row and column of (3.4) are removed. Subsequently, we will reintroduce the equality constraints.

### 3.3.1 Inequality-only Constrained Problems

When only inequality constraints are present, the linear system (3.4) simplifies to

$$\begin{bmatrix} \mathsf{W}_\mu & 0 & \mathsf{A}_{g,\mu}^T \\ 0 & -\mathsf{\Lambda}_\mu & -\mathsf{S}_\mu \\ \mathsf{A}_{g,\mu} & -(1-\mu)\mathsf{I} & -\mu\mathsf{I} \end{bmatrix} \begin{bmatrix} v_x \\ v_s \\ v_g \end{bmatrix} = \begin{bmatrix} u_x \\ u_s \\ u_g \end{bmatrix}. \tag{3.5}$$

If at least one constraint is active, it is easy to show that the lower right $2m \times 2m$ block in the above matrix will become singular as $\mu \to 0$; however, for the moment, we consider the case $\mu > 0$ and invert this block to express $v_s$ and $v_g$ as functions of $v_x$:

$$\begin{bmatrix} v_s \\ v_g \end{bmatrix} = \begin{bmatrix} \mathsf{C}_\mu^{-1} & 0 \\ 0 & \mathsf{C}_\mu^{-1} \end{bmatrix} \begin{bmatrix} -\mu\mathsf{I} & \mathsf{S}_\mu \\ (1-\mu)\mathsf{I} & -\mathsf{\Lambda}_\mu \end{bmatrix} \begin{bmatrix} u_s \\ u_g - \mathsf{A}_{g,\mu} v_x \end{bmatrix}, \tag{3.6}$$

where

$$\mathsf{C}_\mu \equiv \mu\Lambda_\mu - (1-\mu)\mathsf{S}_\mu = \mu(1-\mu)\Lambda_g - \mu^2\mathsf{I} - (1-\mu)^2\mathsf{S}$$

is a diagonal matrix. Substituting the expression for $v_g$ from (3.6) into the first row of (3.5), we find

$$\left[\mathsf{W}_\mu + \mathsf{A}_{g,\mu}^T\mathsf{C}_\mu^{-1}\Lambda_\mu\mathsf{A}_{g,\mu}\right] v_x = u_x - \mathsf{A}_{g,\mu}^T\mathsf{C}_\mu^{-1}\left[(1-\mu)u_s - \Lambda_\mu u_g\right]. \tag{3.7}$$

**Remark 5.** *To derive* (3.6)*, note that the matrices involved in the lower* $2m \times 2m$ *block in* (3.5) *are all diagonal. Consequently, we can apply the formula for the inverse of a* $2\times 2$ *matrix. This yields*

$$\begin{bmatrix} -\Lambda_\mu & -\mathsf{S}_\mu \\ -(1-\mu)\mathsf{I} & -\mu\mathsf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \mathsf{C}_\mu^{-1} & 0 \\ 0 & \mathsf{C}_\mu^{-1} \end{bmatrix} \begin{bmatrix} -\mu\mathsf{I} & \mathsf{S}_\mu \\ (1-\mu)\mathsf{I} & -\Lambda_\mu \end{bmatrix}$$

*where* $\mathsf{C}_\mu$ *holds the determinant of each* $2\times 2$ *matrix on its diagonal.*

**Remark 6.** *Consider the case where all the inequality constraints are active.* $\mathsf{S}_\mu = 0$ *and the matrix in* (3.7) *becomes the Hessian for an augmented-Lagrangian-like function with linearized constraints:*

$$\begin{aligned} \lim_{\mu\to 0} W_\mu + \mathsf{A}_{g,\mu}^T\mathsf{C}_\mu^{-1}\Lambda_\mu\mathsf{A}_{g,\mu} &= \nabla_x^2 f + \lambda^T\nabla_x^2 g + \mathsf{S}^{-1}\mathsf{A}_g^T\Lambda_g\mathsf{A}_g \\ &= \nabla_x^2 f + \lambda^T\nabla_x^2 g + \frac{1}{\tau_s}\mathsf{A}_g^T\Lambda_g\mathsf{A}_g \end{aligned}$$

*Notice that the fixed-value fraction-to-the-boundary rule applied to the slack variable in this thesis would impose the value of* $\tau_s$ *on all slack variables.*

The boundedness of the system (3.7) depends on the behavior of the matrix $\mathsf{C}_\mu$. Assuming strict complementarity, as $\mu \to 0$ we have two situations:

1. If the $i$th constraint is inactive at the solution, then $\lambda_{g,i} = 0$ and $\lim_{\mu\to 0}\left[\mathsf{C}_\mu^{-1}\Lambda_\mu\right]_{ii} = 0$. This has the effective of eliminating the corresponding constraint from the reduced system (3.7).

2. If the $i$th constraint is active, then $s_i = 0$ and $\lim_{\mu\to 0}\left[\mathsf{C}_\mu^{-1}\Lambda_\mu\right]_{ii} = \infty$.

The first situation is desirable, since the constraint is inactive and should not influence the step $v_x$. The second situation is obviously undesirable; however, the safeguards in place during the predictor-corrector phases bound the slacks away from zero, so the inverse of $\mathsf{C}_\mu$ remains well defined in practice. In particular, we use the fraction-to-the-boundary rule at the end of the predictor step and explicit clipping at the end of the corrector step to make sure the slack variables remain larger than $\tau_s$; see Section 2.2.3 for details.

Having established that (3.7) is well defined for all iterates, we now seek to approximately invert this system. To this end, we replace $\mathsf{W}_\mu$ by an approximation denoted by $\mathsf{B}_\mu$:

$$\mathsf{W}_\mu \approx \mathsf{B}_\mu,$$

where $\mathsf{B}_\mu$ is either a scaled identity, $\mathsf{B}_\mu = \beta \mathsf{I}$, or an L-BFGS quasi-Newton approximation [35]. In addition, we use the Lanczos algorithm [76] to construct a low-rank approximation of $\mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu}$ for the nonlinear constraints; that is

$$\mathsf{A}_\mu^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_\mu \approx \mathsf{U} \Sigma \mathsf{V}^T, \tag{3.8}$$

where $\Sigma = \mathsf{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{n_\Sigma}) \in \mathbb{R}^{n_\Sigma \times n_\Sigma}$ is a diagonal matrix holding estimates for the $n_\Sigma$ largest singular values, and $\mathsf{U} \in \mathbb{R}^{m \times n_\Sigma}$ and $\mathsf{V} \in \mathbb{R}^{m \times n_\Sigma}$ are the corresponding approximations to the left and right singular vectors.

**Remark 7.** *The Lanczos algorithm is advantageous in this context, because it only requires matrix-vector products with* $\mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu}$*. Such products can be evaluated using second-order adjoints; see [45] or [66] for further details regarding second-order adjoints. Furthermore, by replacing accurate second-order adjoint PDE solves with corresponding preconditioner applications, the cost of the Lanczos-based SVD can be significantly reduced. This idea is explored in the numerical experiments in Chapter 4.*

Based on the above approximations, the (approximate) inverse of the matrix in (3.7) can be found using the Sherman-Morrison-Woodbury formula as follows:

$$\begin{aligned}\left[\mathsf{W}_\mu + \mathsf{A}_\mu^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_\mu\right]^{-1} &\approx \left[\mathsf{B}_\mu + \mathsf{U}\Sigma\mathsf{V}^T\right]^{-1} \\ &= \mathsf{B}_\mu^{-1} - \mathsf{B}_\mu^{-1}\mathsf{U}\left(\mathsf{I}_{n_\Sigma} + \Sigma\mathsf{V}^T\mathsf{B}_\mu^{-1}\mathsf{U}\right)^{-1}\Sigma\mathsf{V}^T\mathsf{B}_\mu^{-1}, \qquad (3.9)\end{aligned}$$

where $\mathsf{B}_\mu^{-1}$ is either $(1/\beta)\mathsf{I}$ or the L-BFGS approximation. Note that, in the above expression, $\left(\mathsf{I}_{n_\Sigma} + \Sigma\mathsf{V}^T\mathsf{B}_\mu^{-1}\mathsf{U}\right)$ is an $n_\Sigma \times n_\Sigma$ matrix. In the proposed preconditioner, we assume that the number of singular values in the truncated SVD, i.e. $n_\Sigma$, is sufficiently small that we can invert this matrix explicitly using an $LU$ factorization.

In summary, to approximately solve (3.7) we evaluate

$$\begin{aligned}v_x = &\left[\mathsf{B}_\mu^{-1} - \mathsf{B}_\mu^{-1}\mathsf{U}\left(\mathsf{I}_{n_\Sigma} + \Sigma\mathsf{V}^T\mathsf{B}_\mu^{-1}\mathsf{U}\right)^{-1}\Sigma\mathsf{V}^T\mathsf{B}_\mu^{-1}\right] \\ &\left\{u_x - \mathsf{A}_\mu^T\mathsf{C}_\mu^{-1}\left[(1-\mu)u_s - \Lambda_\mu u_g\right]\right\}. \qquad (3.10)\end{aligned}$$

After obtaining $v_x$, we use it in (3.6) to find $v_s$ and $v_\lambda$;

$$v_s = \mathsf{C}_\mu^{-1}\left[-\mu u_s + \mathsf{S}_\mu\left(u_g - \mathsf{A}_\mu v_x\right)\right], \qquad (3.11)$$

$$v_\lambda = \mathsf{C}_\mu^{-1}\left[(1-\mu)u_s - \Lambda_\mu\left(u_\lambda - \mathsf{A}_\mu v_x\right)\right]. \qquad (3.12)$$

We conclude this section by summarizing the inequality-only matrix-free preconditioner in Algorithm 2. Note that the approximate SVD can be performed before each Krylov iterative solve, or it can be performed periodically to reduce cost, possibly at the risk of descreasing the effectiveness of the preconditioner.

### 3.3.2 Preconditioner for Problems with both Inequality and Equality Constraints

When both equality and inequality constraints are present, it is straightforward to extend the preconditioner to handle the general constrained case, since the equality-constraint rows in (3.4) do not involve $v_s$ or $v_g$. Indeed, in the general case

**Algorithm 2:** Matrix-free, approximate SVD preconditioner.

```
Parameters: values at which to evaluate matrices, $x$, $s$, $\lambda_g$, and $\mu$
; $n_\Sigma$
Input       : vectors to precondition, $u_x$, $u_s$, $u_{\lambda_g}$
Output      : preconditioned vectors, $v_x$, $v_s$, $v_{\lambda_g}$
Build the truncated and approximate SVD (3.8), using preconditioner
applications in place of exact second-order adjoint solves
Solve for $v_x$ using (3.10)
Solve for $v_s$ using (3.11)
Solve for $v_{\lambda_g}$ using (3.12)
return $v_x$, $v_s$, and $v_{\lambda_g}$
```

the reduced system becomes

$$\begin{bmatrix} \mathsf{W}_\mu + \mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu} & \mathsf{A}_{h,\mu}^T \\ \mathsf{A}_{h,\mu} & -\mu \mathsf{I} \end{bmatrix} \begin{bmatrix} v_x \\ v_h \end{bmatrix} = \begin{bmatrix} u_x - \mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \left[ (1-\mu) u_s - \Lambda_\mu u_g \right] \\ u_h \end{bmatrix}. \quad (3.13)$$

When both equality and inequality constraints present, we need to solve (3.13) rather than (3.7). Subsequently, $v_s$ and $v_g$ can be recovered by (3.6) the same way as in the inequality-only case. Thus, the problem becomes how to solve (3.13) using matrix-free methods. Once again we take advantage of the Schur complement.

Recall that, if $\mathsf{D}$ and $\left(A - BD^{-1}C\right)$ are invertible, then the solution of the linear system

$$\begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix},$$

is given by,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} I & 0 \\ -D^{-1}C & I \end{bmatrix} \begin{bmatrix} \left(A - BD^{-1}C\right)^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \quad (3.14)$$

To use the above factorization in (3.13), we have to consider two situations:

1. When $\mu$ is not small, e.g. $\mu > \tau_\mu$, then $-\mu\mathsf{I}$ is invertible and we can use (3.14).

2. When $\mu$ gets close to zero, e.g. $\mu < \tau_\mu$, we enforce the clipping rule $\bar{\mu} = \max(\mu, \tau_\mu)$ to prevent $\mu$ from being too small in the preconditioner. We use

$\tau_\mu = 10^{-4}$ for all the numerical experiments in this work. With this safeguard, (3.14) can still be used as a preconditioner.

**Remark 8.** *The choice $\tau_\mu = 10^{-4}$ is made based on numerical experiences. Theoretically, smaller values such as $\tau_\mu = 10^{-6}, 10^{-7}, 10^{-8}$ will make the preconditioner linear system resemble the original linear system better but also make it more ill-conditioned as in the original system. In contrast, larger values such as $\tau_\mu = 10^{-1}, 10^{-2}$ will help with the conditioning of the preconditioner system but will reduce the effectiveness of the preconditioner. Choosing $\tau_\mu = 10^{-4}$ can balance the trade-off between the aforementioned scenarios.*

To summarize, in the general constrained case, $v_x$ and $v_h$ can be obtained as follows:

$$\begin{bmatrix} v_x \\ v_h \end{bmatrix} = \begin{bmatrix} I & 0 \\ \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu} & I \end{bmatrix} \begin{bmatrix} \left(\mathsf{W}_\mu + \mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu} + \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu}^T \mathsf{A}_{h,\mu}\right)^{-1} & 0 \\ 0 & -\frac{1}{\bar{\mu}}\mathsf{I} \end{bmatrix} \begin{bmatrix} I & \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu}^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{u}_x \\ u_h \end{bmatrix}, \tag{3.15}$$

where $\hat{u}_x$ denotes the $x$-component of the right-hand side of (3.13). The most involved operation in (3.15) is the application of the inverse Schur complelement:

$$\left(\mathsf{W}_\mu + \mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu} + \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu}^T \mathsf{A}_{h,\mu}\right)^{-1} \left(\hat{u}_x + \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu}^T u_h\right). \tag{3.16}$$

By stacking $\mathsf{A}_{h,\mu}$ and $\mathsf{A}_{g,\mu}$ together, the Schur complement can be expressed as

$$\mathsf{W}_\mu + \mathsf{A}_\mu^T \Sigma_\mu \mathsf{A}_\mu = \mathsf{W}_\mu + \mathsf{A}_{g,\mu}^T \mathsf{C}_\mu^{-1} \Lambda_\mu \mathsf{A}_{g,\mu} + \frac{1}{\bar{\mu}}\mathsf{A}_{h,\mu}^T \mathsf{A}_{h,\mu}, \tag{3.17}$$

where

$$\mathsf{A}_\mu = \begin{bmatrix} \mathsf{A}_{h,\mu} \\ \mathsf{A}_{g,\mu} \end{bmatrix} \quad \text{and} \quad \Sigma_\mu = \begin{bmatrix} \frac{1}{\bar{\mu}}\mathsf{I} & 0 \\ 0 & \mathsf{C}_\mu^{-1}\Lambda_\mu \end{bmatrix}. \tag{3.18}$$

Inspecting (3.17), we see that it has the same form as the matrix on the left-hand side of (3.9). Therefore, as before, we use the Lanczos method to construct an approximate singular-value decomposition of $\mathsf{A}_\mu^T \Sigma_\mu \mathsf{A}_\mu$, and replace $\mathsf{W}_\mu$ with a diagonal or quasi-Newton approximation. Subsequently, we can use the Sherman-Morrison-Woodbury formula to invert the approximation in the same manner as in the previous section.

## 3.4 Scalable Quadratic Optimization Problem

### 3.4.1 Problem Description

We conclude this chapter with a numerical experiment to test the effectiveness of the approximate SVD preconditioner defined in Algorithm 2. In particular, we are interested in the performance of the algorithm with the preconditioner as the size of the problem increases. To this end, we consider the following scalable optimization problem in which we can independently control the size and conditioning of the Hessian and constraint Jacobian:

$$\begin{array}{rl} \min\limits_{x \in R^n} & \frac{1}{2} x^T \mathsf{Q} x + g^T x \\ \text{subject to} & \mathsf{A}x \geq b. \end{array} . \tag{3.19}$$

The vector $g \in \mathbb{R}^n$ is randomly sampled from a uniform distribution on $[0, 1)$, while $b \in \mathbb{R}^n$ is randomly sampled from $[0, 0.1)$.

The Hessian $\mathsf{Q}$ is diagonal with entries

$$\mathsf{Q}_{ii} = \begin{cases} \frac{1}{i}, & i = 1, 2, ..., \kappa, \\ \frac{1}{\kappa}, & i = \kappa + 1, ..., n, \end{cases}$$

where $\kappa \leq n$. This definition produces a Hessian with a condition number of $\kappa$, so that its condition number can be controlled independently of the problem dimension.

The constraint Jacobian $\mathsf{A} \in \mathbb{R}^{n \times n}$ is constructed as follows. First, a diagonal matrix $\mathsf{D} \in \mathbb{R}^{n \times n}$ of singular values is defined similar to $\mathsf{Q}$:

$$\mathsf{D}_{ii} = \begin{cases} \frac{1}{i^2}, & i = 1, 2, ..., \nu, \\ \frac{1}{\nu^2}, & i = \nu + 1, ..., n, \end{cases}$$

where $\nu \leq n$. Next, matrices $\mathsf{A}_L \in \mathbb{R}^{n\times n}$ and $\mathsf{A}_R \in \mathbb{R}^{n\times n}$ of random integers from the discrete uniform distribution on the interval [0,10), are factorized using a QR decomposition:

$$\mathsf{A}_L = \mathsf{Q}_L\mathsf{R}_L,$$
$$\mathsf{A}_R = \mathsf{Q}_R\mathsf{R}_R.$$

Finally, the constraint Jacobian is formed as $\mathsf{A} = \mathsf{Q}_L\mathsf{D}\mathsf{Q}_R$. Consequently, based on this construction, the condition number of $\mathsf{A}$ is $\nu^2$ and is also controllable independent of the problem dimension.

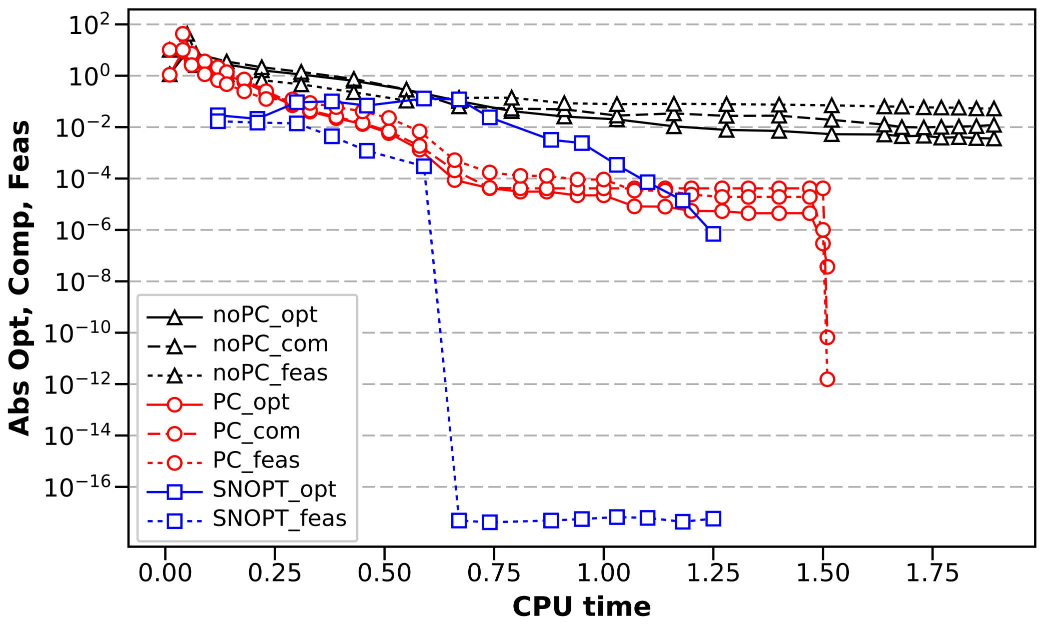
For this study we set $\kappa = \nu = 9$, which gives condition numbers of $\text{cond}(\mathsf{Q}) = 9$, $\text{cond}(\mathsf{A}) = 81$. While these are modest, even small condition numbers, we emphasize that the conditioning of the KKT matrix $\nabla_q H$ remains high, between $10^4$ and $10^9$.

**Remark 9.** *Although the matrices for this synthetic problem are available explicitly, our algorithm does not exploit this and remains matrix-free, i.e. it only uses matrix-vector products.*

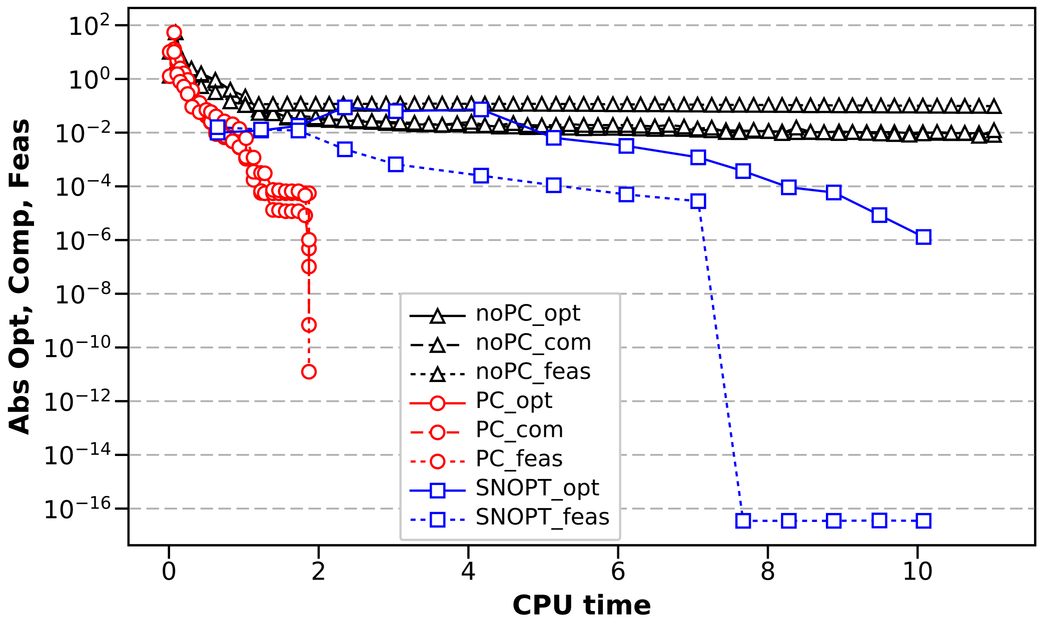
### 3.4.2 Results

We consider two problems sizes, $n = 200$ and $n = 500$. Figure 3.1 plots the absolute optimality, complementarity and feasibility histories versus CPU time. The predictor-corrector algorithm is applied both with and without the preconditioner. In addition, the plots include the results from SNOPT [20], a well-validated active-set SQP optimization library. See Section 4.1 for further details about the convergence criteria and SNOPT.

Without the preconditioner, the predictor-corrector algorithm is not competitive and does not converge within the time shown. This illustrates the need for preconditioning Newton-Krylov optmization algorithms, even for modest sized problems. The results also indicate that the proposed algorithm is competitive with SNOPT when $n = 200$ and the proposed algorithm outperforms SNOPT when $n = 500$. In both cases, the feasibility of SNOPT sharply decreases once when the correct set of active constraints are determined. It is notable that when $n = 500$, SNOPT takes significantly longer time than the new algorithm to converge.



(a) $n = 200$



(b) $n = 500$

**Figure 3.1. Convergence histories for the scalable quadratic problem with $n = 200$ and $n = 500$. The results for the proposed algorithm, with and without preconditioning, are plotted together with the results from SNOPT**

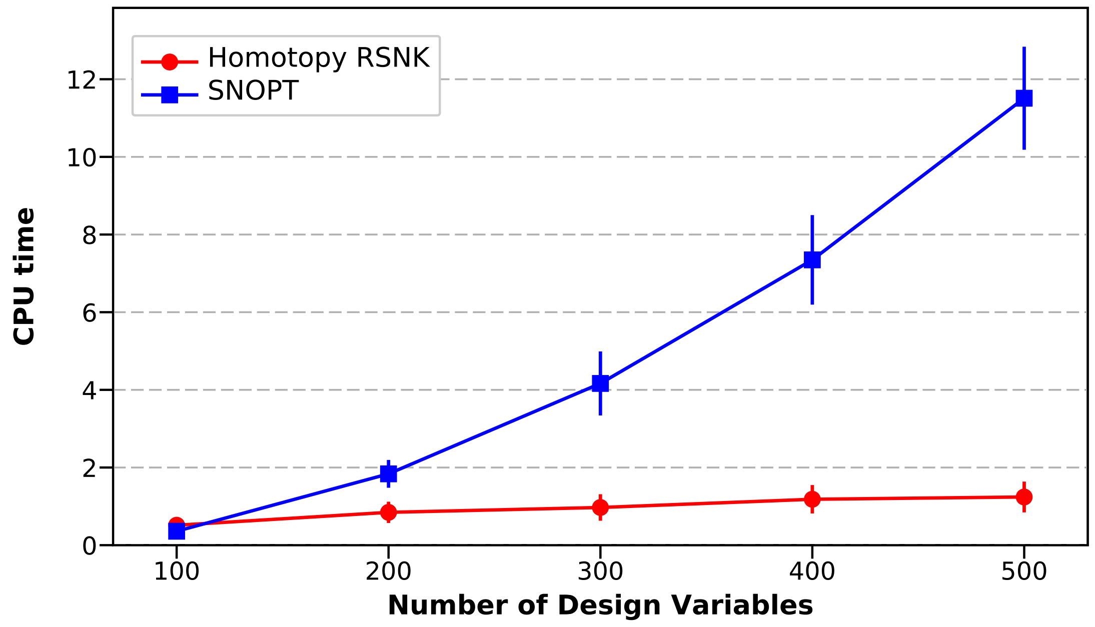


**Figure 3.2. CPU cost versus number of design variables for the quadratic optimization problem**

To further explore the scalability of the proposed algorithm, we ran 100 random cases for each fixed sized problem, for $n = 100, 200, 300, 400, 500$. In each random sample, the gradient of the objective function $g$, and the right hand side vector $b$ in the linear constraint in (3.19) were randomly generated as described earlier. The diagonal matrices $\mathsf{Q}$ and $\mathsf{D}$ remain fixed, while $\mathsf{A}_L$ and $\mathsf{A}_R$ are randomly generated. Thus the constraint Jacobian $\mathsf{A}$ also varies for each case. The starting point for each run is also randomly generated. Figure 3.2 shows the two standard deviations in time while the problem size increases. As can be seen, the new method exhibits good scalability relative to SNOPT.

We should emphasize that the proposed SVD preconditioner is particularly effective for this synthetic problem, because it is designed in such a way that the constraint Jacobian can be approiximated effectively using the Lanzcos-based SVD approximation. Put another way, the SVD approximation can capture the main characteristics of the matrix in (3.8). For real-world problems, the distribution of the singular values of the constraint Jacobian is unknown, and may possess an unfavorable distribution pattern for the proposed preconditioner to remain effective.

# CHAPTER 4
# NUMERICAL RESULTS: TESTS AND APPLICATIONS

The goal of this chapter is to verify and apply the matrix-free RSNK optimization algorithm described in Chapters 2 and 3. We begin by summarizing the software implementation of the proposed algorithm. We also describe the established algorithm we will compare against. Next, we use the well-established CUTEr test set to verify the algorithm. We then present the results of applying the algorithm to two large-scale PDE-constrained optimization problems.

## 4.1 Software Implementation and Benchmarking

This section begins with an overview of Kona, the software library in which the algorithms from Chapters 2 and 3 are implemented. This is followed by a descrption of the convergence criteria used in the matrix-free algorithm. Lastly, we provide a description of SNOPT, an independent and popular optimization library against which we will compare Kona.

### 4.1.1 The Kona Optimization Library

The globalized RSNK method and preconditioners are part of an in-house optimization package, Kona [77]. Kona is a matrix-free, architecture-agnostic optimization package designed to solve reduced-space PDE-constrained optimization problems. Kona implements several optimization algorithms including an unconstrained reduced-space quasi-Newton method, an unconstrained reduced-space Newton-CG method, an equality constrained reduced-space Newton-Krylov method based on FLECS [78], and an equality-constrained composite-step RSNK algorithm. These high-level optimization algorithms are built from modular components that implement various optimization subroutines including: iterative matrix-free solvers like FGMRES, STCG, FLECS for solving linear systems; globalization techniques like trust-region, backtracking line search, and a line search based on the strong Wolfe conditions; and merit functions like the augmented Lagrangian and the $l_2$ merit function. In addition, Kona can evaluate Hessian- and Jacobian-vector products by solving the approriate adjoint equations.

Architecturally, Kona uses an abstract interface to separate the optimization algorithms from PDE-solver specific implementations, and this allows the development of new optimization algorithms independent of the PDE solver. From a user's perspective, one has to write a solver interface to Kona providing the function operations listed in Table 4.1 of [79]. In particular, this solver interface provides the objective, constraint, gradient, Jacobian-vector, and vector-Jacobian products. These products allow Kona to evaluate more complex total sensitivities (e.g. Hessian- and Jacobian-vector products) that are needed for RSNK algorithms.

### 4.1.2 Convergence Criterion and Default Parameters

To measure the progress of the optimization iterations, Kona evaluates the infinity norm of each block row in $F(x)$. Specifically, we will refer to optimality, complementarity, and feasibility as defined below:

$$\begin{aligned}
\text{Optimality} &= \max_j |(\nabla_x \mathcal{L})_j|, \\
\text{Complementarity} &= \max_j |s_j \lambda_j|, \\
\text{Feasibility} &= \max_j |c_j|, \quad \text{where} \quad c = \left[ h(x)^T, (g(x) - s)^T \right].
\end{aligned}$$

In the subsequent tests, the convergence plots display the above metrics versus computational cost. In the past we have reported the relative convergence criteria, which normalizes the above quantities by their initial values; however, in the tests below we report the absolute criteria. The relative convergence criteria is not appropriate in the current context, because the initial complementarity products are zero due to the zero initial multipliers, and the initial feasibility could be zero if $s_0 = g(x_0)$ is used for inequality-only constrained problems.

Table A.1 in Appendix shows the complete list of the parameters used in the test problems, including the default values set in the algorithm and their recommended ranges.

### 4.1.3 SNOPT

We will benchmark the performance of Kona against the software library SNOPT, which is short for Sparse Nonlinear OPTimizer [20]. SNOPT is a gradient-based sequential quadratic optimization method for solving large-scale nonlinear problems with thousands of constraints and design variables. It uses an augmented Lagrangian merit function, and the Hessian of the Lagrangian is approximated using a limited-memory quasi-Newton method.

SNOPT uses scaled feasibility and optimality criteria to measure the progress of the optimization iterations. The feasibility criterion measures the maximum nonlinear constraint violation, and is normalized by the norm of the current estimated solution, denoted $x^{(k)}$ below:

$$\text{SNOPT feasibility} = \max_i c_i/\|x^{(k)}\|, \quad \text{where} \quad c = \left[h(x)^T, (g(x)-s)^T\right].$$

where $x$ is the current iterative point, and $c_i$ is the violation of the $i$th nonlinear constraint [80]. SNOPT's feasibility tolerance, $\epsilon_r$, has a default value of $10^{-6}$.

The optimality criterion measures the maximum complementarity slackness for the design variables, and it is calculated using the following formulas:

$$\text{SNOPT optimality} = \max_j \text{Comp}_j/\|\pi\|$$

where $\pi = [\lambda_h^T, \lambda_g^T]^T$ is the Lagrangian multiplier vector; $\text{Comp}_j$ measures the complementarity slackness for the $j$th variable and is defined by:

$$\text{Comp}_j = \begin{cases} (\nabla_x \mathcal{L})_j \min(x_j - l_j, 1) & \text{if } (\nabla_x \mathcal{L})_j \geq 0 \\ -(\nabla_x \mathcal{L})_j \min(u_j - x_j, 1) & \text{if } (\nabla_x \mathcal{L})_j < 0 \end{cases}$$

where $l_j$ and $u_j$ denote the lower and upper bounds on the $j$th variable.

## 4.2 CUTEr Test Problems

We have chosen to verify the algorithms presented in Chapters 2 and 3 using the CUTEr test problem set [81], [82]. This section begins with a brief description of the CUTEr test suite, and then it presents the results of applying Kona and SNOPT to a subset of the CUTEr problems. The CUTEr problem set, its interface, and its classification methods are discussed in Appendix A.1, A.2 and A.3.

### 4.2.1 Problem Description

The CUTEr set is a collection of test problems to test new optimization codes and develop new algorithms. The problem set ranges from small differentiable unconstrained problems to large-scale dense and sparse problems with both equality and inequality nonlinear constraints. Some of the test problems exhibit challenges and numerical difficulties observable in practice, such as bad scaling in the objective and/or constraint functions, multiple local solutions, non-regular solutions where the linearly-independent constraint qualification is not satisfied, etc. A number of optimization packages have interfaced with CUTEr [83], including Ipopt, Knitro, Minos, and SNOPT, to name a few.

The problems are written in the so-called Standard Input Format (SIF) [84]. A decoder translates the problems written in SIF to a library written in Fortran 77 that provide the tools to access the function values, Jacobians, and, sometimes, Hessians to the optimization packages.

### 4.2.2 Results

As the CUTEr test problem set contains a large collection of problems with assorted features, some of which are inappropriate for the RSNK algorithm, only a subset of the problems are considered here. The following criteria were used to remove certain problems from being considered in the tests; see Appendix A.3 for further details on the classification of the CUTEr problems.

- Objective functions with the following type: **U**, **C**, and **L**
- Constraint functions with the following type: **U**, **X**
- Irregular problems
- Problems with no correct solutions provided

Table 4.1 lists the results on the selected subset of the Cuter problems. The first column, 'Name' , lists the name of the problems as found in the ASCII files from [85]. The files from [85] also include the problem origin, authors, classifications, SIF problem cards, and, sometimes, the solutions. The second column 'n, $m_{\text{CUTEr}}$, l, m' are the number of design variables, the number of constraints as described in Appendix A.1, and the number of equality and inequality constraints as defined in this thesis. The third column provides the origin of the problem, if available. The fourth column lists the parameters used in the homotopy RSNK algorithm that were changed for some of the problems: the initial step size $\alpha_0$, the nominal distance $\delta_{\text{targ}}$ and nominal angle $\phi^{\circ}_{\text{targ}}$ as defined in Section 2.2.2, and the rank of the SVD approximation in Lanczos method used in the preconditioner 3.8. The fifth column is the optimal value of the objective function found using Kona, while the sixth column is optimal objective value found using SNOPT. The last column is the optimized objective function value as provided by Reference [85].

The results in Table 4.1 suggest that the globalized RSNK algorithm and the preconditioners proposed in this thesis can deliver accurate solutions. They even succeed in a few cases where SNOPT fails. Despite this success, several points are worth highlighting:

1. CUTEr is only used as verification here, and it is not intended to demonstrate the efficiency of the RSNK algorithm. Indeed, the CUTEr test problems are sufficiently small in size and have explicit Jacobians, so these problems do not benefit from the matrix-free RSNK algorithm.
2. The parameters $\alpha_0$, $\delta_{\text{targ}}$, $\phi^{\circ}_{\text{targ}}$, $n_{\Sigma}$ play important roles in the robustness and efficiency of the new algorithm. In some cases the parameters have to be changed manually to get convergence. In contrast, the parameters for SNOPT remained unchanged for all the test problems.
3. The globalized RSNK algorithm is based on the constraint qualification assumption; therefore, it will fail for problems with redundant constraints.
4. If the null space of the inequality constraint Jacobian includes a vector of ones, the preconditioner will fail. Because Kona starts the Lanczos algorithm (used to build the preconditioner) with a vector of ones, for problems whose inequality constraint Jacobian contain certain structures, e.g. $[-1, -3, 3, 1]$ on each row, the Lanczos method will crash as the matrix vector product is zero.

**Table 4.1. Subsets of CUTEr problem results**

| Name | n, $m_{\text{CUTEr}}$, l, m | Origin | $\alpha_0$, $\delta_{\text{targ}}$, $\phi^\circ_{\text{targ}}$, $n_\Sigma$ | $f_{\text{kona}}$ | $f_{\text{snopt}}$ | $f^*$ |
|---|---|---|---|---|---|---|
| AIRPORT | 84, 42, 0, 210 | | 0.05, 20, 20, 40 | 47952.976 | 47952.702 | 47952.696 |
| BIGGSC4 | 4, 7, 0, 21 | | 0.05, 1, 5, 2 | -24.49999 | -24.375 | -24.5 |
| BLOCKQP1 | 25, 11, 10, 71 | | 0.05, 1, 5, 2 | 2.403846 | 2.4038461 | -6.4988 |
| BLOCKQP2 | 25, 11, 10, 71 | | 0.05, 1, 5, 2 | -6.20165 | -2.5e+14 | -6.2017 |
| BLOWEYA | 22, 12, 12, 46 | | 0.05, 1, 5, 10 | -4.56931 | -4.56931 | -4.56932 |
| BT1 | 2, 1, 1, 2 | | 0.05, 10, 20, - | -0.99999 | -0.99999 | -1 |
| BT2 | 3, 1, 1, 2 | | 0.05, 10, 20, - | 0.032568 | 0.032568 | 0.0325682 |
| BT3 | 5, 3, 3, 6 | | 0.05, 10, 20, - | 4.093013 | 4.093023 | 4.093011 |
| BT4 | 3, 2, 2, 4 | | 0.05, 10, 20, - | -45.5106 | -45.5105 | -45.5105 |
| BT5 | 3, 2, 2, 4 | non-convex | 0.05, 10, 20, - | 961.7152 | 961.7152 | 961.7152 |
| BT6 | 5, 2,2, 4 | | 0.05, 10, 20, - | 0.277045 | 0.277045 | 0.277045 |
| BT7 | 5,3, 3,6 | | 0.05, 1, 10, - | 306.4957 | 360.3797 | 306.4964 |
| BT8 | 5,2,2,4 | | 0.05, 1, 10, - | 1.000000 | 1.000000 | 1 |
| BT11 | 5,3,3,6 | | 0.05, 1, 10, - | 0.824892 | 0.824892 | 0.824892 |
| BT12 | 5,3, 3,6 | | 0.05, 1, 10, - | 6.188113 | 6.188119 | 6.188119 |
| CHAIN | 102, 51, 51, 106 | | 0.05, 10,20,50 | 5.07047 | -25473973 | 5.07226 |
| CHANDHEQ | 10, 10, 10, 30 | | 0.05, 10,20,10 | 0 | 0 | 0 |
| DIPIGRI | 7,4,0,4 | | 0.05,10,20, 7 | 680.6301 | 680.6301 | 680.63 |

| | | | | | | |
|---|---|---|---|---|---|---|
| DTOC1L | 58, 36, 36, 80 | DTOC | 0.05,10,20,30 | 0.0735945 | 0.0307271 | 0.0735931 |
| DTOC1NA | 58, 36, 36, 80 | non-convex | 0.05,10,20,30 | 0.0753143 | 0.0320512 | 0.0753126 |
| DTOC1NB | 58, 36, 36, 80 | non-convex | 0.05,10,20,30 | 0.0984835 | 0.0524561 | 0.0984812 |
| DTOC1NC | 58, 36, 36, 80 | non-convex | 0.05,10,20,30 | 0.3123065 | 0.2703142 | 0.3123101 |
| DTOC1ND | 58, 36, 36, 80 | non-convex | 0.05,10,20,30 | 0.4155301 | 0.3845229 | 0.4142563 |
| DTOC2 | 58, 36, 36, 80 | non-convex | 0.05,10,20,30 | 0.4859889 | 1.5e-09 | 0.4859839 |
| DTOC3 | 29, 18, 18, 40 | convex | 0.05,10,20,30 | 224.59038 | 8.2e-13 | 224.59038 |
| DTOC4 | 29, 18, 18, 40 | non-convex | 0.05,10,20,30 | 3.7508222 | 1.6e-13 | 3.7507839 |
| DTOC5 | 19, 9, 9, 20 | convex | 0.05,10,20,30 | 1.451900 | 4.4e-13 | 1.4518939 |
| DTOC6 | 19, 9, 9, 20 | convex | 0.05,10,20,30 | 19.80411 | 9.740196 | 19.80411 |
| DUAL2 | 96, 1, 1, 192 | | 0.05,10,20,2 | 3.37546E-2 | 3.36831E-2 | 3.37337E-2 |
| DUAL3 | 111, 1, 1, 222 | | 0.05,10,20,1 | 1.35542E-1 | 1.35544E-1 | 1.35756E-1 |
| EQC | 9, 3, 0, 21 | Quality Control | 0.05,10,20,5 | -4789.377 | -4789.377 | 1138.416 |
| GILBERT | 2,1,1,1 | | 0.05,1,10,2 | 0.251624 | 0.251624 | 0.251626 |
| GILBERT | 5,1,1,1 | | 0.05,1,10,2 | 1.339713 | 1.339713 | 1.339710 |
| GILBERT | 10,1,1,1 | | 0.05,1,10,2 | 3.345201 | 3.345201 | 3.345196 |
| GMNCASE1 | 175, 300, 0, 300 | optimized control | 0.5, 5, 5, 100 | 0.267087 | 0.266973 | 0.266733 |
| GMNCASE4 | 175, 350, 0, 350 | optimized control | 0.05, 10, 20, 170 | 5.9132e3 | 5.9469e3 | 5.9468e3 |
| HS100 | 7, 4, 0, 4 | | 0.05,10,20,- | 680.6300 | 680.6300 | 680.6300 |
| HS100LNP | 7, 2, 2, 0 | | 0.05,10,20,- | 680.6300 | 680.6300 | 680.6300 |

| | | | | | | |
|---|---|---|---|---|---|---|
| HS104 | 8,5, 0,22 | | 0.05,10,20,3 | 3.951162 | 3.951163 | 3.951163 |
| HS11 | 2,1,0,1 | | 0.05,10,20,1 | -8.498464 | -8.498465 | -8.49846 |
| HS112 | 10,3,3,10 | | 0.05,1,5,3 | -47.7611 | -47.7611 | -47.7075 |
| HS113 | 10, 8,0, 8 | | 0.05,10,20,1 | 24.30621 | 24.30621 | 24.30621 |
| HS118 | 15, 17,0, 59 | | 0.05,10,20,- | 664.7021 | -1748.637 | 664.8204 |
| HS12 | 2, 1,0, 1 | | 0.05,10,20,5 | -30 | -30 | -30 |
| HS14 | 2,2,1,3 | | 0.05,10,20,1 | 1.393465 | 1.393461 | 1.423224 |
| HS15 | 2,2,0,3 | | 0.05,10,20,1 | 306.5000 | 6.8e-17 | 306.5 |
| HS16 | 2,2,0,5 | | 0.05,10,20,1 | 0.25 | 3.98 | 0.25 |
| HS21 | 2,1,0,5 | Schittkowski | 0.5,5,10,1 | -99.3434 | -99.9900 | -99.96 |
| HS35 | 3,1,0,4 | | 0.05,1,5,1 | 0.111111 | 0.111111 | 0.111111 |
| HS35I | 3,1, 0,7 | | 0.05,1,5,1 | 0.111111 | 0.111111 | 0.111111 |
| HS44 | 4,6, 0,10 | | 0.05,1,20,3 | -13.000 | -4.01e+14 | -13.0 |
| HS44NEW | 4,6,0,10 | | 0.05,1,20,3 | -13 | -3.20e+14 | -13.0 |
| HS51 | 5,3,3,6 | | 0.05,1,20,3 | 4.46e-19 | 5.86E-14 | 0.0 |
| HS52 | 5,3,3,6 | | 0.05,1,20,3 | 5.326634 | 5.326647 | 5.326643 |
| HS53 | 5,3,3,16 | | 0.05,1,5,3 | 4.093023 | 4.093023 | 4.093023 |
| HS118 | 15,17,0,59 | | 0.05,50,5,3 | 664.8205 | -1748.638 | 664.8204 |
| HATFLDH | 4, 7,0, 21 | | 0.05,1,5,2 | -24.5002 | -24.375 | 24.5 |
| MOSARQP1 | 36, 10, 0, 46 | convex quadratic | 0.05,1,5,4 | -24.1436 | -52.0492 | -24.1377 |

| | | | | | | |
|---|---|---|---|---|---|---|
| MOSARQP2 | 36,10, 0, 46 | | 0.05,1,5,4 | -35.6982 | -55.1623 | -35.6981 |
| STCQP1 | 17, 8, 8, 50 | convex quadratic | 0.05,1,5,5 | 494.4054 | 494.5208 | 494.5209 |
| SOSQP1 | 20, 11, 11, 62 | non-convex | 0.05,1,5,2 | 5.9e-07 | -4E-16 | 0.0 |
| SOSQP2 | 20, 11, 11, 62 | | 0.05,1,5,5 | -3.99779 | -4.045649 | -3.99781 |
| YAO | 22, 20, 0, 25 | | 0.05,1,5,10 | 2.398829 | 0.0037148 | 2.39883 |

## 4.3 Stress-Constrained Mass Minimization

### 4.3.1 Problem Description

Our next numerical experiment is a PDE-constrained structural sizing problem with state-based constraints. The problem was previously considered by the authors in [77], and the geometry and boundary conditions are illustrated in Figure 4.1. Informally, the problem consists of minimizing the plate mass with respect to the thickness distribution, subject to bound constraints and the von Mises stress criteria.

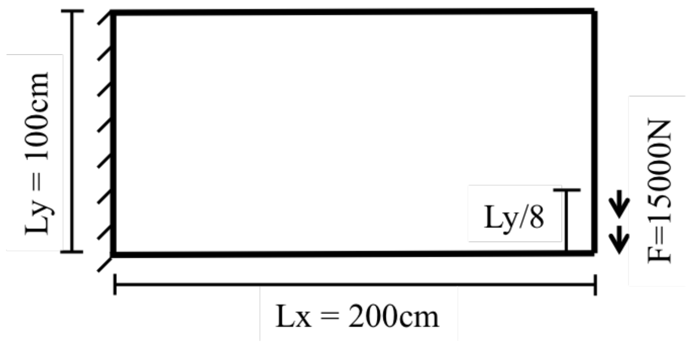


**Figure 4.1. Plate thickness design problem**

The response of the plate to the applied force is modeled using the equations of linear elasticity assuming 2D plane stress. The governing equations are discretized using the finite element method with bilinear elements. A uniform mesh of $n_x \times n_y$ elements is employed, and the thickness distribution is piecewise constant over each element. These thickness values are taken to be the design variables, so there are $n = n_x n_y$ design variables.

The formal optimization statement is

$$\begin{aligned} \min_{x} \quad & \mathsf{mass}(x) \\ \text{subject to} \quad & \mathsf{stress}_i(x) \leq \sigma_{\max}, \forall\, i = 1, 2, \ldots, n, \\ & x_l \leq x_i \leq x_u, \qquad \forall\, i = 1, 2, \ldots, n, \end{aligned}$$

where $\mathsf{mass}(x)$ is the plate mass, and $\mathsf{stress}_i(x)$ is the von Mises stress criterion on the $i$th element. The thickness of each element is bounded by the lower and upper values, $x_l = 0.02$ and $x_u = 0.98$.

The objective function, $\mathsf{mass}(x)$, is a sum of all the elements' mass. Each element's mass is obtained by a constant material density times the filtered thickness of that element. The filtered thickness of each element is a weighted sum of the design variables of the adjacent elements within a conic filter, with the weights determined by $1 - \frac{r}{r_0}$ and normalized so that the sum is 1; the radius of the conic $r_0 = 2$, and $r$ is the distance of the adjacent element to the central element. In summary, the mass is a linear function of the design variables and is given by

$$\mathsf{mass}(x) = \sum_{i=1}^{n} m_i \sum_{j} w_{ij} x_j,$$

where $m_i$ is the mass per unit thickness of the $i$th element of the structure, and $w_{ij}$ is the weight from the filter.

The von Mises stress criterion on the $i$th element can be expressed as

$$\mathsf{stress}_i(x) = 1 - u(x)^T \mathsf{B}^T \mathsf{G} \mathsf{B} u(x),$$

where $\mathsf{B}u(x)$ gives the strain as a function of the displacement, $u(x)$, and $\mathsf{G}$ is a positive-definite matrix. The displacement is a function of the design variables through the discretized governing equation

$$R(x, u) = \mathsf{K}(x) u - b,$$

where $\mathsf{K}(x)$ is the stiffness matrix and $b$ is the force vector.

We consider a set of three mesh sizes by varying $n_x$ and $n_y$. Table 4.2 lists the mesh dimensions and number of design variables for the three cases. We will informally refer to these three cases as the small, medium, and large cases.

Table 4.2 also lists the number of approximate singular values used in Algorithm 2 to construct the preconditioner. As the number of design variables increases in this structural optimization problem, the number of singular values in the SVD approximation has to increase accordingly, at about $\frac{1}{6}$ of the number of design variables.

**Table 4.2. Mesh dimensions ($n_x$ and $n_y$), number of design variables ($n = n_x n_y$), and size of the approximate SVD preconditioner ($n_\Sigma$) for the plate-thickness optimization problem**

| **Case** | $\mathbf{n_x}$ | $\mathbf{n_y}$ | $\mathbf{n}$ | $\mathbf{n_\Sigma}$ |
|---|---|---|---|---|
| Small | 16 | 8 | 128 | 20 |
| Medium | 32 | 16 | 512 | 80 |
| Large | 64 | 32 | 2048 | 320 |

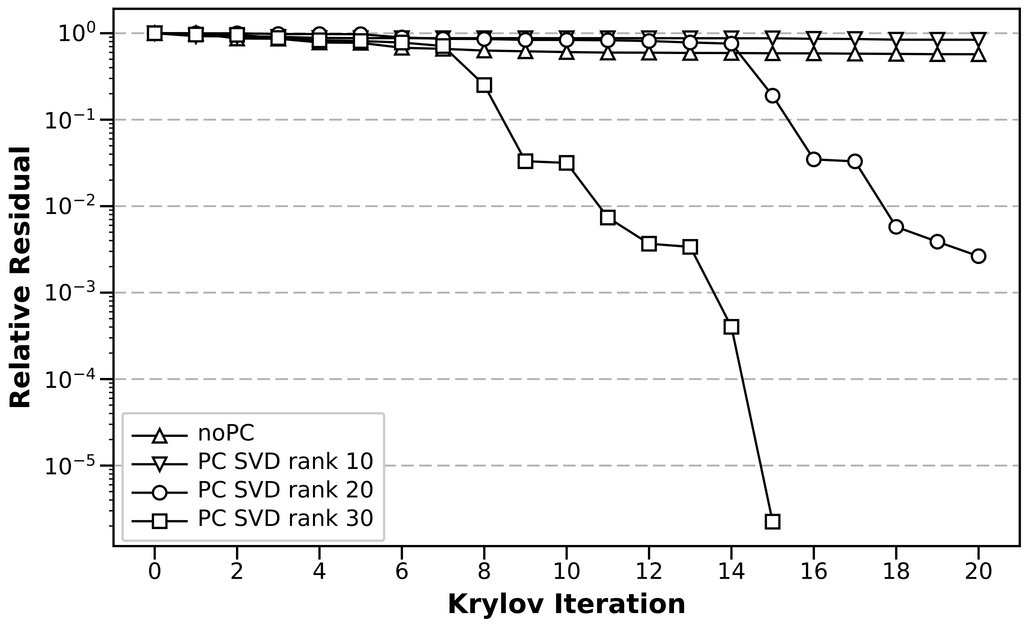


**Figure 4.2. Impact of $n_\Sigma$, the rank of the approximate SVD in Algorithm 2, on the Krylov iterative method**

### 4.3.2 Effectiveness of Preconditioner

To show the effectiveness of the proposed preconditioner, Figure 4.2 plots the convergence history of the Krylov method when the Lanczos-based preconditioner uses an increasing number of SVD ranks. The KKT system is extracted from the first Newton step of the final corrector phase at $\mu < \epsilon_\mu$ (i.e. the last homotopy iteration) for the small case. The KKT system is most ill-conditioned at the final corrector step, so this particular system provides the most stringent test for the preconditioner.

### 4.3.3 Optimal Thickness and Comparison with SNOPT

The structural optimization problem was solved using the proposed algorithm, with and without the preconditioner. Convergence histories are compared with those of SNOPT in Figures 4.5a, 4.5b, and 4.5c for the small, medium, and larges cases, respectively. The cost on the horiztonal axis measures the equivalent number of PDE solves, i.e., the total CPU time divided by the time needed for one forward solution. The vertical axis measures the change in absolute optimality, absolute feasiblity, and absolute complementarity.

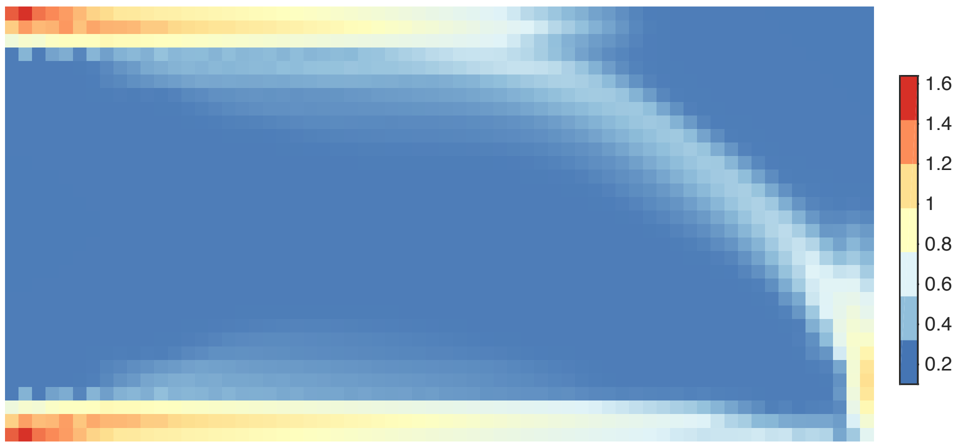
The final optimal thickness distribution on the large mesh obtained using the homotopy-globalized RSNK method with SVD preconditioner is plotted in Figure 4.3a; the distribution obtained without the preconditioner is shown in Figure 4.3b; finally, the thickness distribution obtained using SNOPT is shown in Figure 4.3c. The thickness distribution pattern in all three plots is qualitatively similar and in agreement with physical intuition for this problem.

However, the solution using the globalized RSNK method with the preconditioner has a sharper resolution than that without the preconditioner. In addition, The thickness distribution using SNOPT is ruffled along the trapezoid edges; this is consistent with the convergence plots of SNOPT in 4.5c below, which shows that SNOPT is having difficulty converging the large problem.

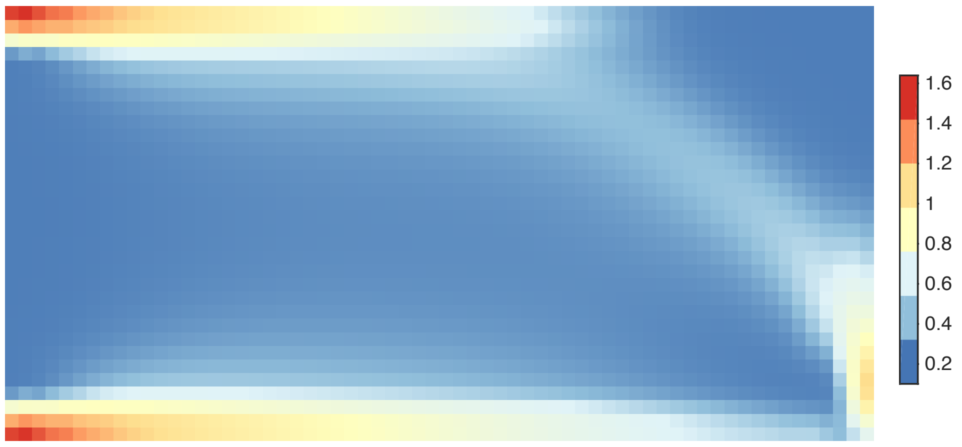
To quantitatively measure the differences in the thickness distributions of the three methods, Figure 4.4 selectively show the thickness distributions along horizontal slices at different vertical locations where SNOPT oscillates the most. The y direction goes up, so the lower values of y are in the bottom. Figure 4.4 also show that the thickness is constant over elements.

The convergence histories in Figure 4.5 highlight the need for preconditioning on this particular problem: without the approximate-SVD preconditioner, the optimality, complementarity and feasibility cannot be sufficiently reduced, despite the fact that the thickness distribution still looks reasonable. In contrast, when the approximate-SVD preconditioner is used the proposed algorithm can successfully reduce the optimality and feasibility to the desired tolerance.

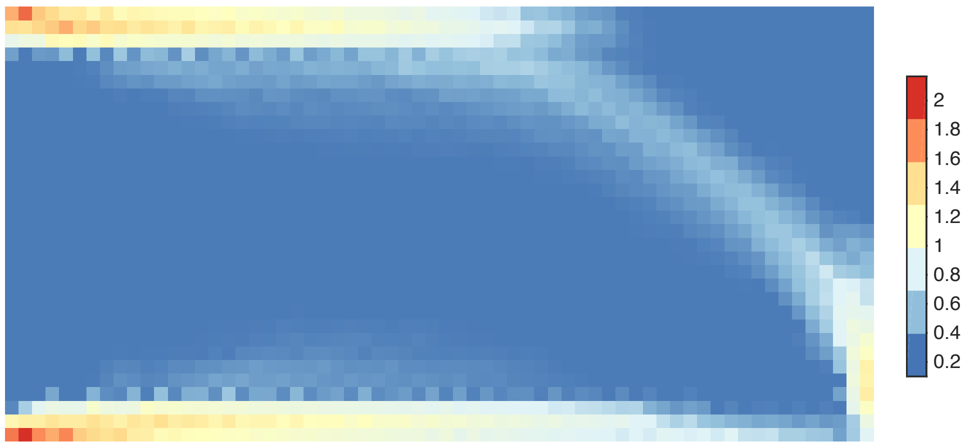
Figures 4.5a–4.5c also show that the performance of SNOPT degrades as the



(a) Globalized RSNK with Preconditioner

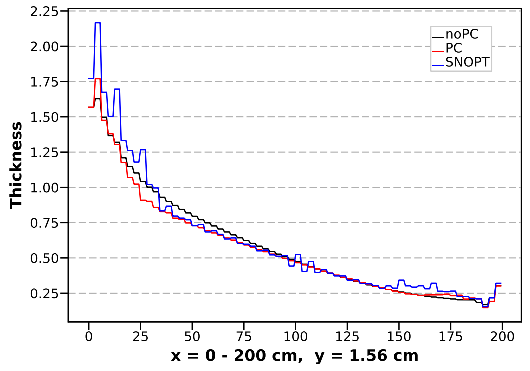


(b) Globalized RSNK without Preconditioner

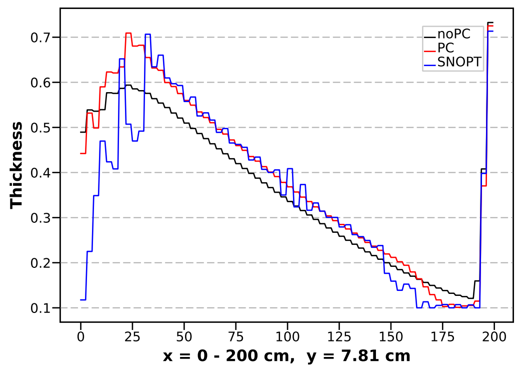


(c) SNOPT

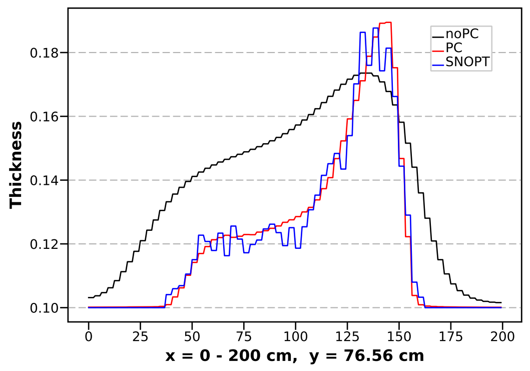
**Figure 4.3. Optimal thickness distribution using Algorithm 1 with Preconditioner, without Preconditioner, and SNOPT**



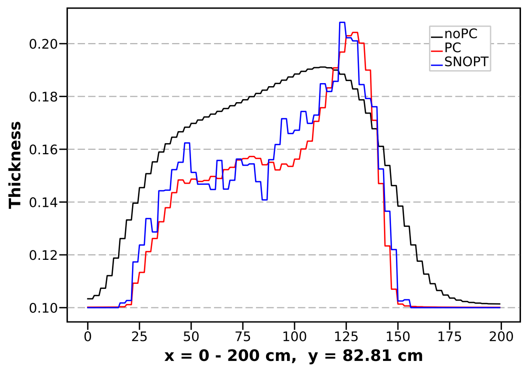
(a) small



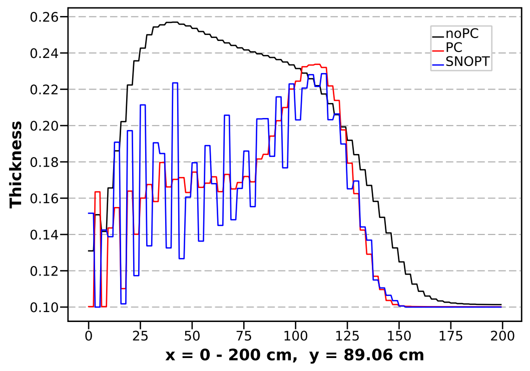
(b) small



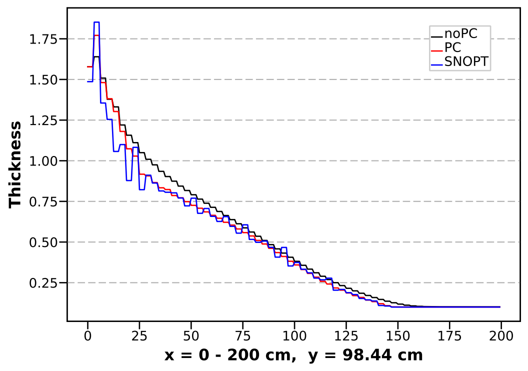
(c) medium



(d) medium



(e) large



(f) large

**Figure 4.4. Comparison of thickness distribution along horizontal slices**

problem size increases; on the small problem, optimality is reduced by only three orders of magnitude, while on the large problem the optimality fluctuates and struggles to decrease significantly. The feasibility achieved by SNOPT is somewhat better than its optimality, but it also degrades with problem size. The proposed predictor-corrector algorithm is able to converge the optimality and feasiblity five orders of magnitude on all three problems. We see that the optimization cost increases approximately linearly with $n$ on this problem, and the cost is driven primarily by the increasing number of singular values used in the preconditioner.

Last but not least, it is worth noting that, the small amount of regularization $\epsilon_\mu = 10^{-6}$ at the final corrector phase of the RSNK method should have helped with its convergence. In contrast, SNOPT has no regularization. It is thus estimated that linearly dependent constraint Jacobian might exist and regularization can help improve the performance of optimization algorithms. However, the interface of SNOPT makes it not easy to add external regularizations, and attempts have been made but without success.

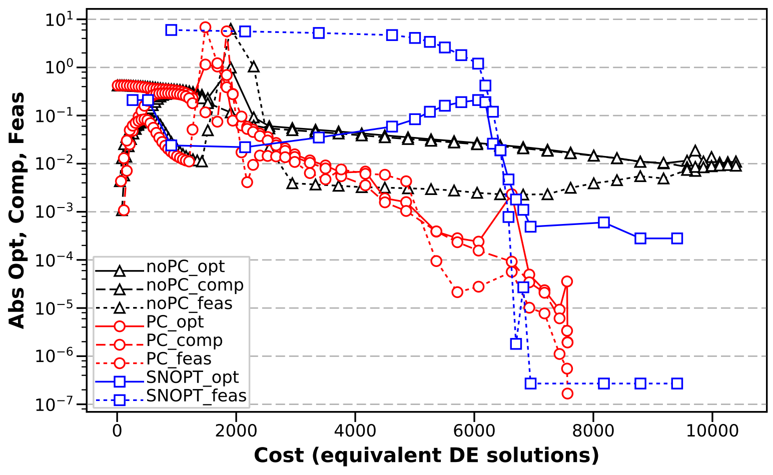
## 4.4 Aerodynamic Shape Optimization

The last application is an Euler-based aerodynamic shape optimization problem starting from the NASA Common Research Model (CRM) wing [86]. This problem is an invisicid version of a problem defined by the AIAA Aerodynamic Design Optimization Discussion Group (ADODG) [87], and involves drag minimization subject to lift, pitching moment, and geometric constraints.

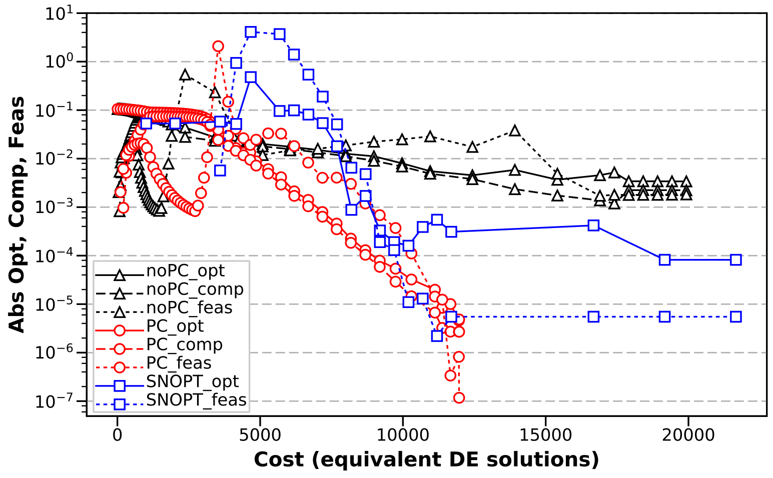
### 4.4.1 Problem Description

The baseline wing-only geometry is extracted from the CRM wing-body configuration with a blunt trailing edge. An IGES file is provided to the public as shown in Figure 4.6. The CRM is modeled after a contemporary transonic commercial airliner, with a size similar to that of Boeing 777. It has been designed with good aerodynamic performance, but with an aggressive pressure recovery on the outboard wing, which provides room for performance improvements.

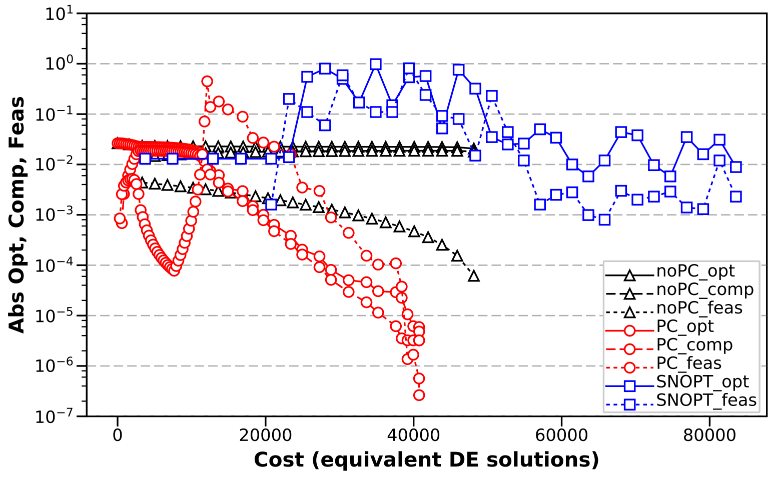
For completeness and repeatability, we provide the following geometric infor-



(a) small



(b) medium



(c) large

**Figure 4.5. Convergence histories for the three sizes of the structural optimization problem**

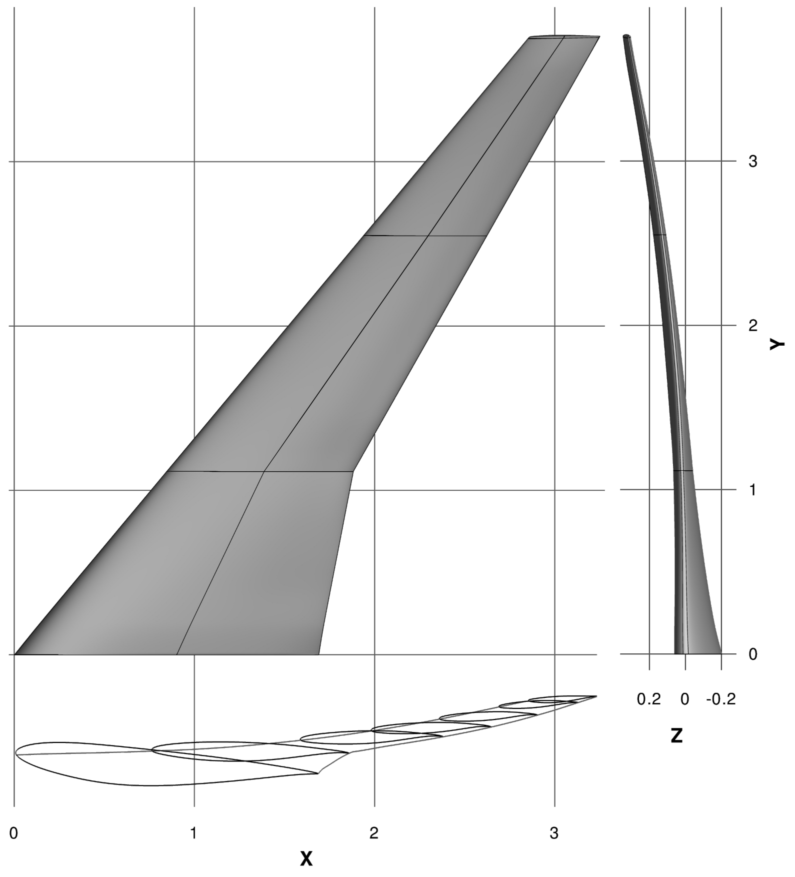


**Figure 4.6. CRM wing [21]**

mation. The origin of the coordinate system is at the leading edge of the wing root, and all coordinates are scaled by the mean aerodynamic chord of 275.8 inches. Pitching moments are measured about the point $(1.2077, 0, 0.007669)$ using the reference length. All aerodynamic coefficients are computed based on the projected area $S_{\text{ref}} = 3.407014$ squared reference units.

### 4.4.2 Mesh Generation, Parameterization and Flow Solver

The mesh CGNS file is generated by the University of Michigan's MDOLab's in-house hyperbolic mesh generator, and uses an O-grid topology method from the wing surface to a farfield a distance 25 times the wing span away from the origin. Three levels of mesh are available for Euler cases as shown in Table 4.3. The results in the table show that the drag coefficient appears to be grid converged on the L1

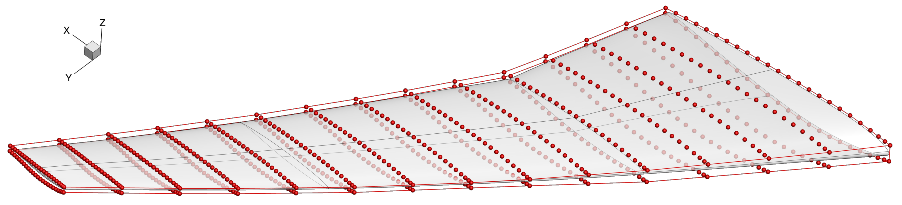


**Figure 4.7. CRM FFD [21]**

grid. Therefore, only the L1 grid is used in this study.

**Table 4.3. Three grid levels and baseline aerodynamic coefficients**

| **Mesh level** | **Mesh size** | $\mathbf{C_D}$ | $\mathbf{C_L}$ | $\mathbf{C_M}$ | $\alpha$ |
|---|---:|---|---|---|---|
| L0 | 840,192 | 0.00995 | 0.49879 | -0.20182 | 2.5° |
| L1 | 105,024 | 0.01085 | 0.48925 | -0.19502 | 2.5° |
| L2 | 13,000 | 0.01417 | 0.46638 | -0.18142 | 2.5° |

The wing geometry is parameterized using a free-form deformation (FFD) volume method [88]. In FFD, the wing geometry is embedded inside a B-spline volume, and the geometric changes of the wing surface are performed by changing the B-spline control points. Figure 4.7 shows the FFD volume and the control points. For the following study, the displacement of the control points in the vertical $z$ direction are used as the design variables.

The flow solver used in this study is SUmb [89], a finite-volume, cell-centered multi-block structured flow solver that can model a variety of problems, including the compressible Euler, laminar Navier-Stokes and Reynolds-Averaged Navier-Stokes (RANS) equations. In this study, the flow was modeled with the compressible Euler equations. The DRP cluster on Rensselaer Polytechnic Institute's CCI was used, consisting of 64 nodes; each node has 128GB of system memory and two eight-core 2.6 GHz Intel Xeon E5-2650 processors.

### 4.4.3 Optimization Problem Statement

During the optimization, the wing's sectional shape and twist are altered by changing the vertical $z$ coordinates of the FFD control points. The control points are distributed in $n_x$ chord wise locations and across $n_y$ spanwise sections; see Figure 4.7. The leading and trailing edges of the root section are fixed. For the other sections, the trailing edge is fixed while the leading edge is free. This permits arbitrary wing twists to be implicitly represented by the remaining degrees of freedom. The wing planform is fixed because the $x$ and $y$ coordinates of the FFD control points are fixed. The angle of attack is one of the design variables and is permitted to change.

The optimization problem can be stated formally as

$$\begin{array}{rll}
\min\limits_{x} & C_D & \text{Drag coefficient objective} \\
\text{subject to} & C_L \geq 0.5 & \text{Lift coefficient constraint} \\
& C_{M_y} \geq -0.17 & \text{Pitching moment constraint} \\
& t \geq 0.25 t_{\text{base}} & \text{Minimum thickness constraints} \\
& V \geq V_{\text{base}} & \text{Minimum volume constraint} \\
& \Delta z_{TE,\text{upper}} = -\Delta z_{TE,\text{lower}} & \text{Fixed trailing edge constraints} \\
& \Delta z_{LE,\text{upper, root}} = -\Delta z_{LE,\text{lower, root}} & \text{Fixed leading edge of the wing root}
\end{array}$$

The thicknesss constraints are imposed at 25 chordwise locations from 1% to 99% along the chord and 30 spanwise locations covering the full span. The design variables include the angle of attack plus the FFD shape design variables. The distributions of the FFD control points considered in this work are listed in Table 4.4. Each spanwise station can control a distinct airfoil shape, and the chordwise points control the shape of each airfoil, with half on the top and half on the bottom.

The lift coefficient, $C_L$, drag coefficient, $C_D$, and pitching moment coefficient, $C_{M_y}$, are aerodynamic coefficients calculated from the state variables. The state variables are implicit functions of the angle of attack and the wing surface shape through the compressible Euler equations. The flight condition considered is at Mach number 0.65.

**Table 4.4. Number of geometric design variables from three different sizes of FFD boxes**

| | **192** | **480** | **768** |
|---|---|---|---|
| Chordwise | 12 | 20 | 24 |
| Spanwise | 8 | 12 | 16 |
| Vertical | 2 | 2 | 2 |

### 4.4.4 Results

This section presents the convergence results for both Kona and SNOPT. For Kona, the convergence plots contain the iteration history of the infinity norm of the complementarity products, the feasibility, and the optimality. For SNOPT, the history of the merit function, the feasibility and optimality are plotted.

#### 4.4.4.1 Kona Results

Figures 4.9, 4.10, and 4.11 show the convergence histories for the L1 grid, with the number of design variables equal to 192, 480, and 768, respectively. Subplot (a) in these figures show the complementarity, feasibility and optimality histories, while subplot (b) shows the histories of $C_D$, $C_L$, and $C_{My}$. Note that the optimality and feasibility are only reduced by approximately $10^{-2}$, which is similiar to the tolerance achieved by SNOPT as shown in the next subsection. In addition, Figure 7 in [21] also exhibits a similar amount of reduction in feasibility and optimality, albeit for the RANS equations.

Note that in Kona, the $C_L$, and $C_{My}$ inequality constraints do not become feasible very quickly, and they remain infeasible for the first half of the iteration. That could be explained by the stronger influence of the homotopy term on the optimization steps. Toward the start of the second half of the convergence history, the inequality constraints are satisfied.

Table 4.5 lists the initial and optimal aerodynamic coefficients using Kona method.

**Table 4.5. The initial and optimal aerodynamic coefficients using Kona**

| | Size | $C_D \times 10^4$ | $C_L$ | $C_{My}$ |
|---|---|---|---|---|
| Initial | | 108.5017 | 0.489247 | -0.195020 |
| | 192 | 96.1716 | 0.499805 | -0.170138 |
| Optimal | 480 | 97.7451 | 0.499907 | -0.170002 |
| | 768 | 99.0509 | 0.499902 | -0.169954 |

**Table 4.6. The initial and optimal aerodynamic coefficients using SNOPT**

| | Size | $C_D \times 10^4$ | $C_L$ | $C_{My}$ |
|---|---|---|---|---|
| Initial | | 108.5025 | 0.489247 | -0.195021 |
| | 192 | 97.9155 | 0.499997 | -0.169998 |
| Optimal | 480 | 98.0529 | 0.499999 | -0.169999 |
| | 768 | 99.0552 | 0.499999 | -0.170000 |

#### 4.4.4.2 SNOPT Results

For comparison, Figures 4.12, 4.13, and 4.14 show the SNOPT convergence histories for the L1 grid with the number of design variables equal to 192, 480 to 768, respectively.

Table 4.6 lists the initial and optimal aerodynamic coefficients using SNOPT method.

To intuitively see the change of the design variables during the optimization process, Figure 4.8 shows the initial, Kona optimized, and SNOPT optimized wing airfoil profiles at six different span-wise locations,

#### 4.4.4.3 Discussion

The results above demonstrate that the homotopy RSNK method can solve a difficult, large-scale aerodynamic shape optimization problem. However, its computational performance is disappointing on this problem relative to SNOPT. Kona's poor performance is largely due to the computational expense of forming the SVD-based preconditioner. This can be explained by the fact that one Lanczos iteration in the preconditioner involves one Jacobian vector product and one vector Jacobian product for both the equality and inequality constraints. This makes the SVD ap-

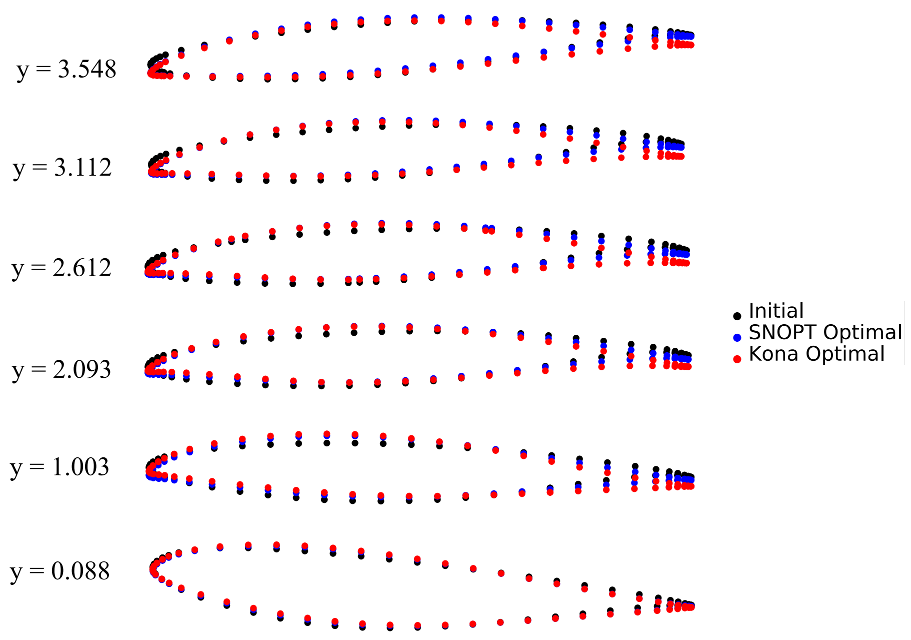
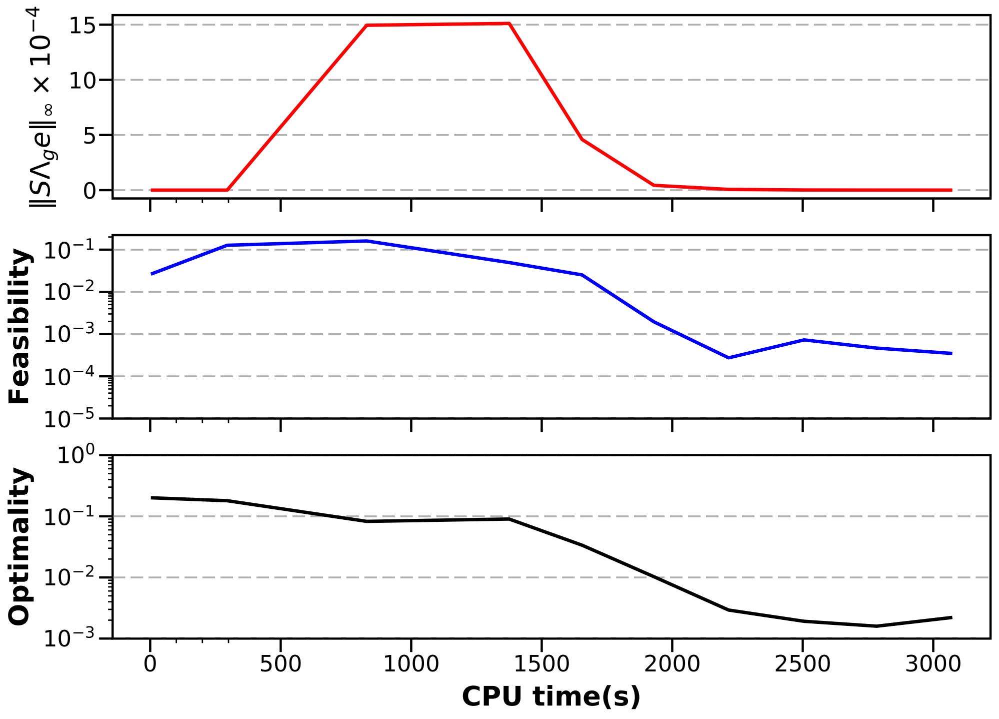
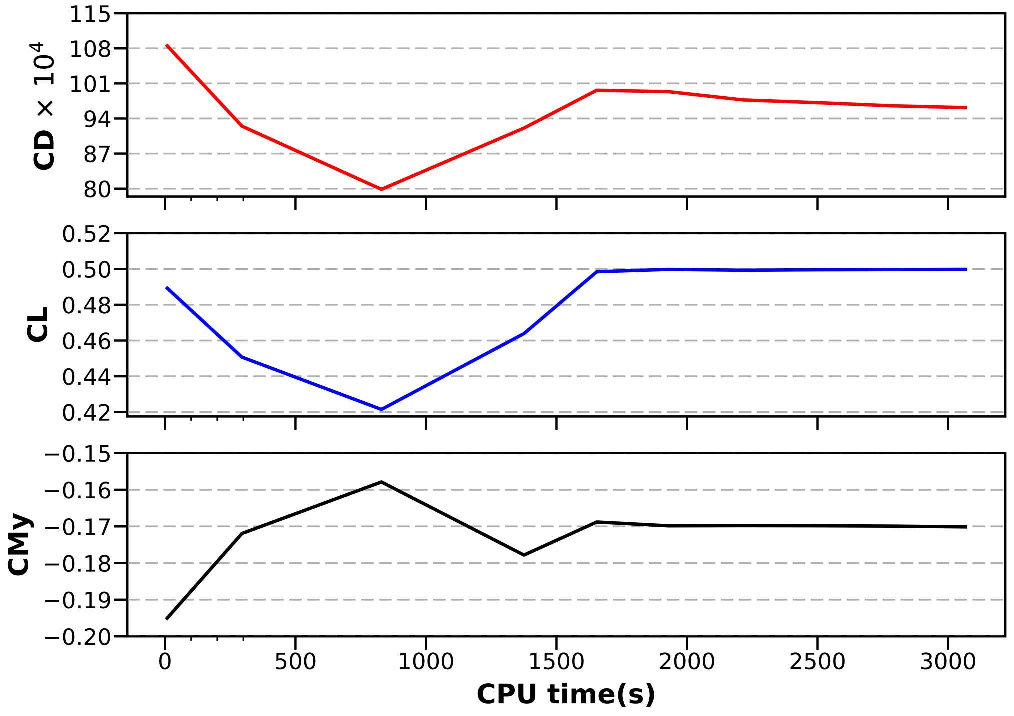


**Figure 4.8. Initial, SNOPT optimized, and SNOPT optimized airfoil profiles at different span-wise locations**

proximation expensive, because it uses 20 Lanczos iterations at each optimization iteration point. In addition, the inequality constraints consist of 750 thickness constraints, 1 volume constraint, and 2 nonlinear aerodynamic constraints. Thus, the preconditioner is lumping the nonlinear and linear inequality constraints together when computing the Jacobian-vector and vector-Jacobian products, which makes the SVD approximation less effective at approximating the nonlinear part. Rather than using the SVD to approximate the linear constraints, it would be advantageous to use the explicit linear Jacobians.

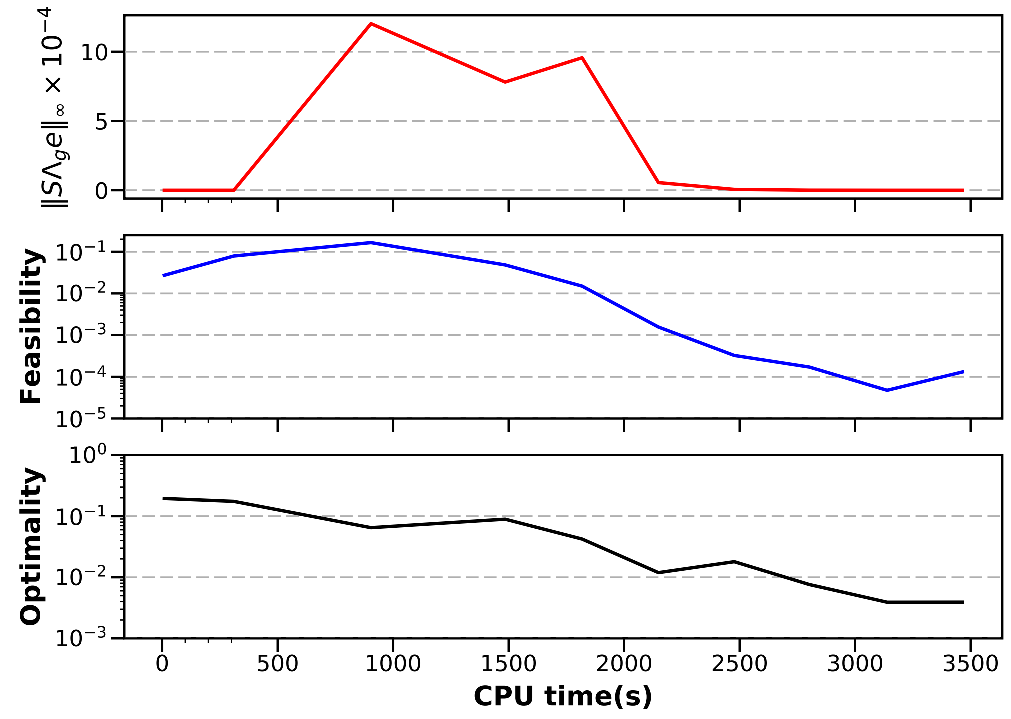


(a)

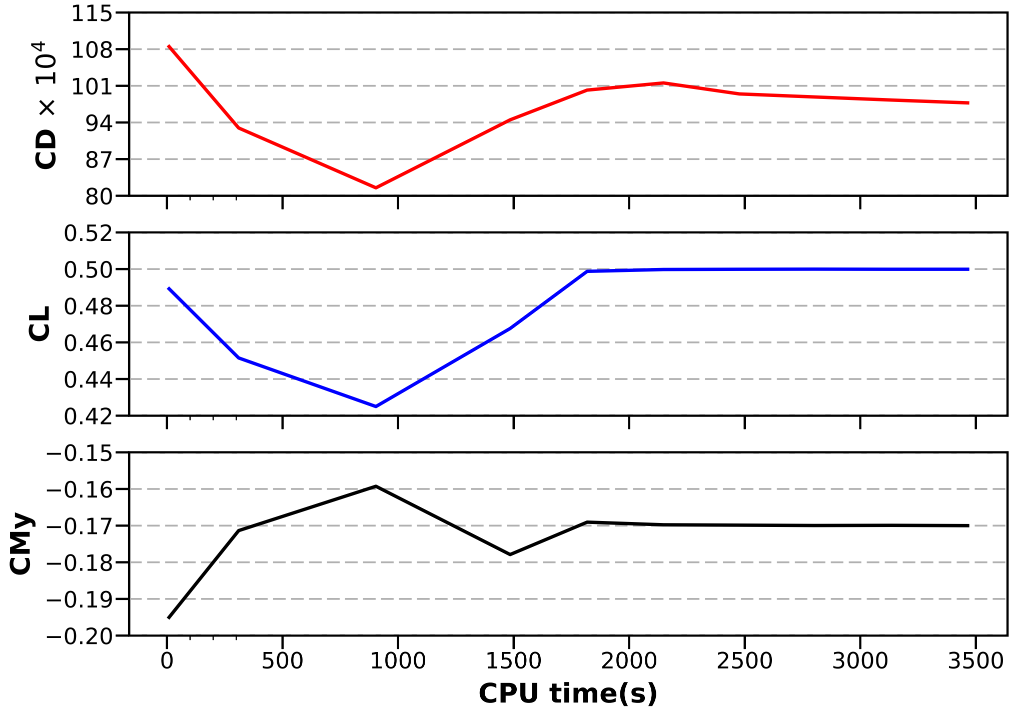


(b)

**Figure 4.9. Convergence histories for L1 grid, no. of design 192**

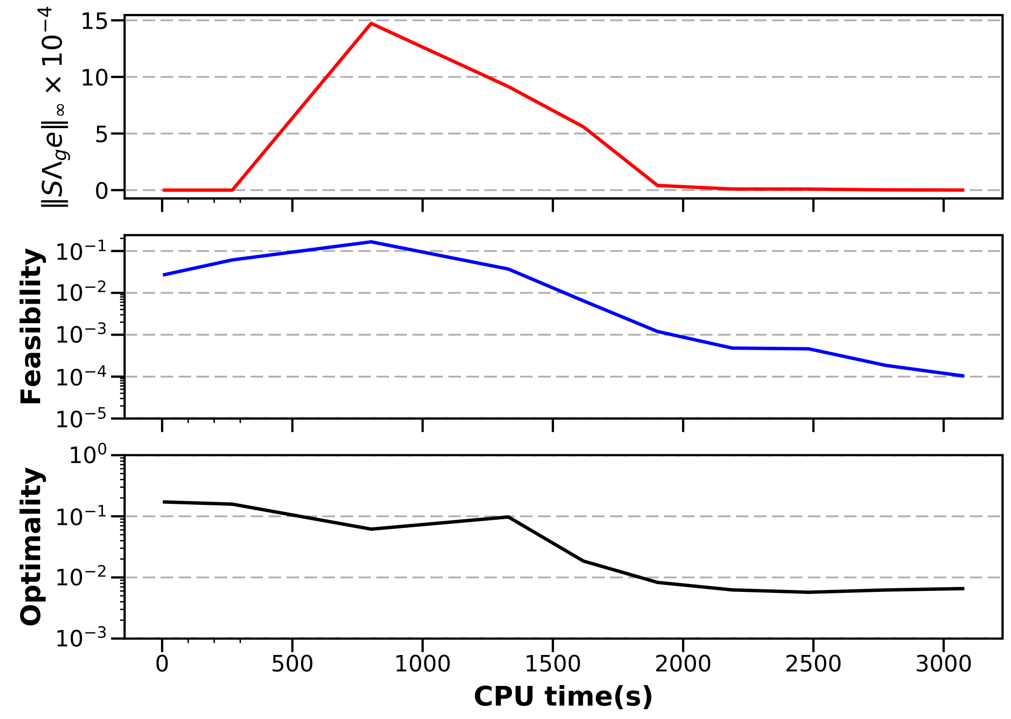


(a)

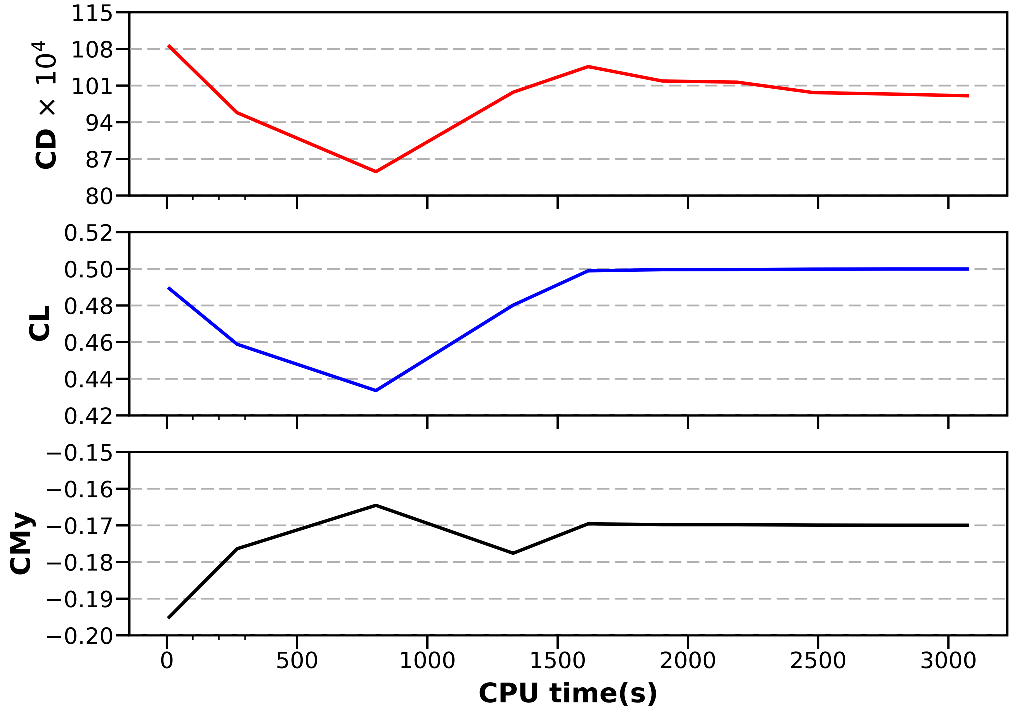


(b)

**Figure 4.10. Convergence histories for L1 grid, no. of design 480**

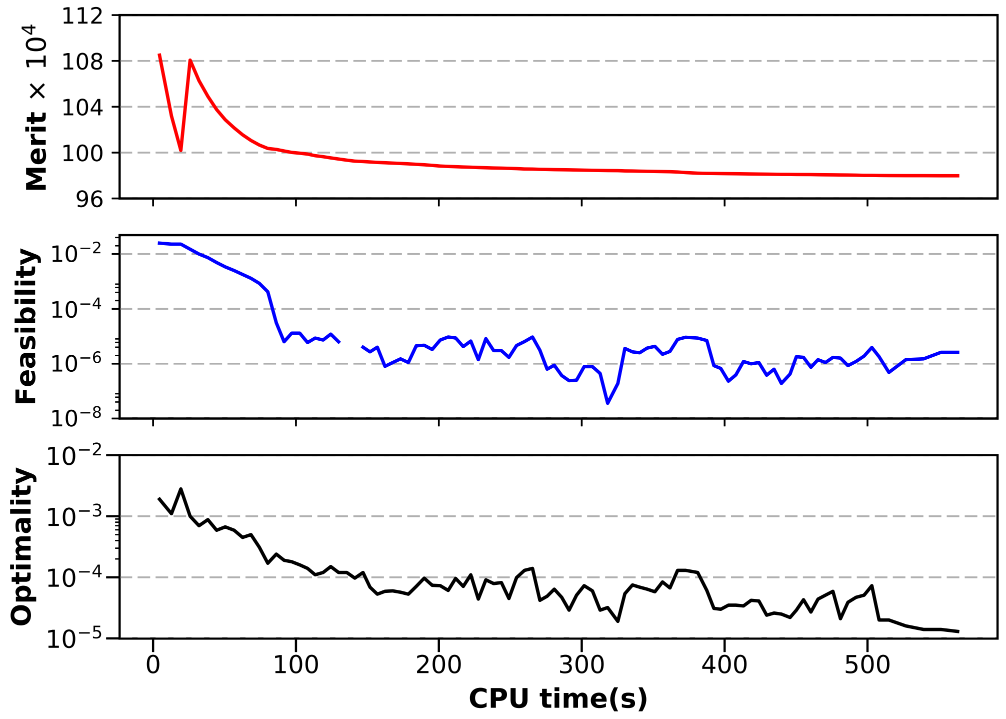


(a)

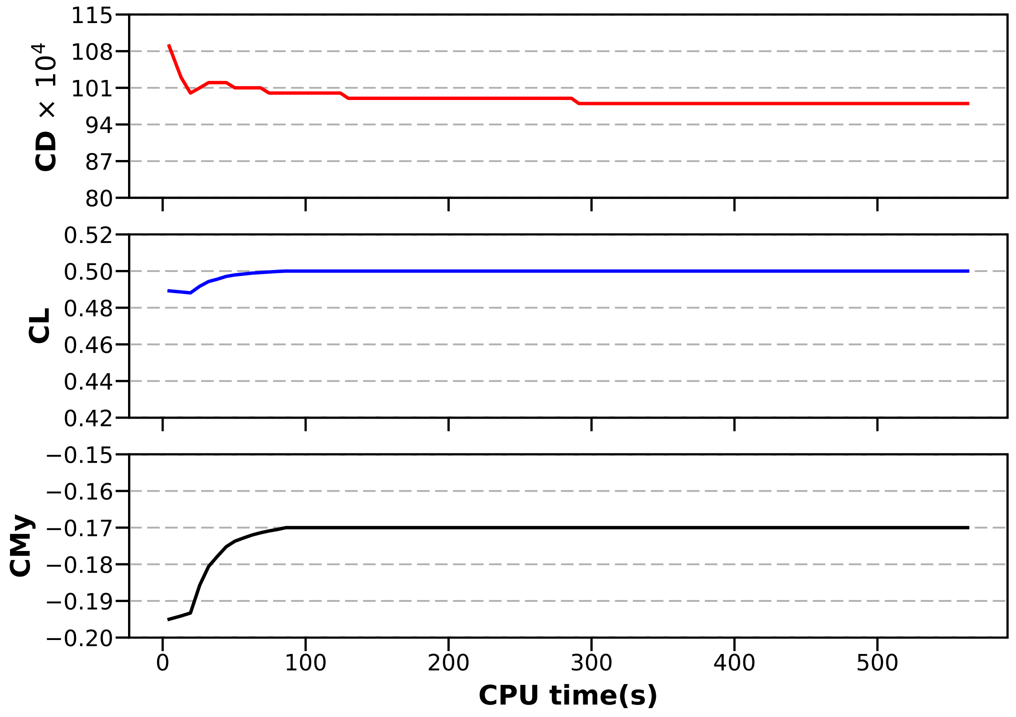


(b)

**Figure 4.11. Convergence histories for L1 grid, no. of design 768**

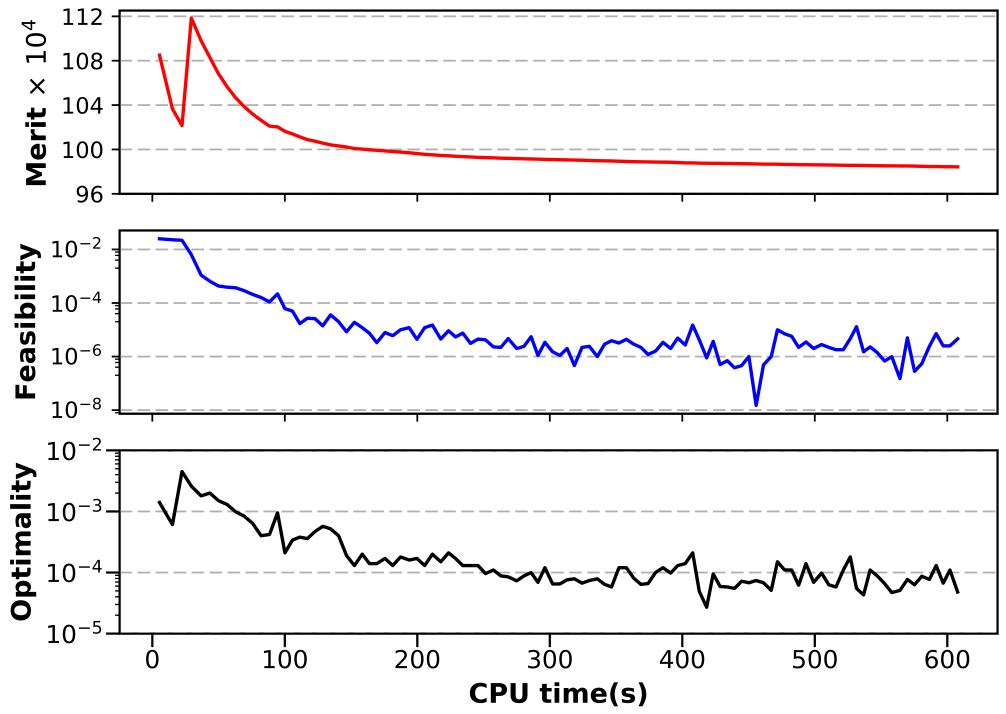


(a)

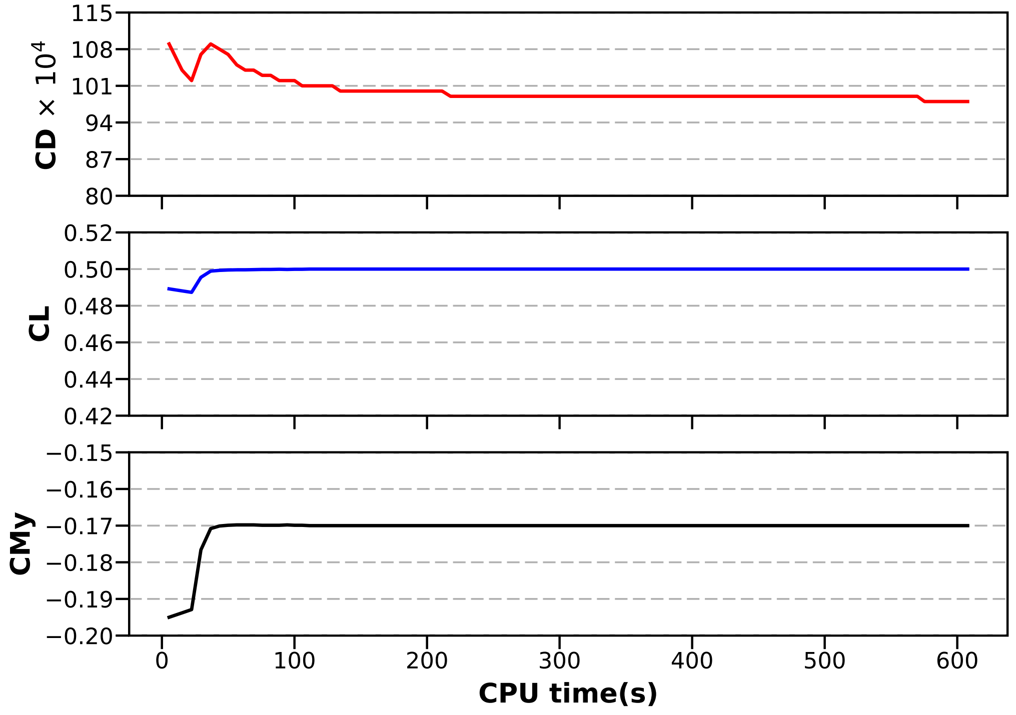


(b)

**Figure 4.12. Convergence histories for L1 grid, no. of design 192**

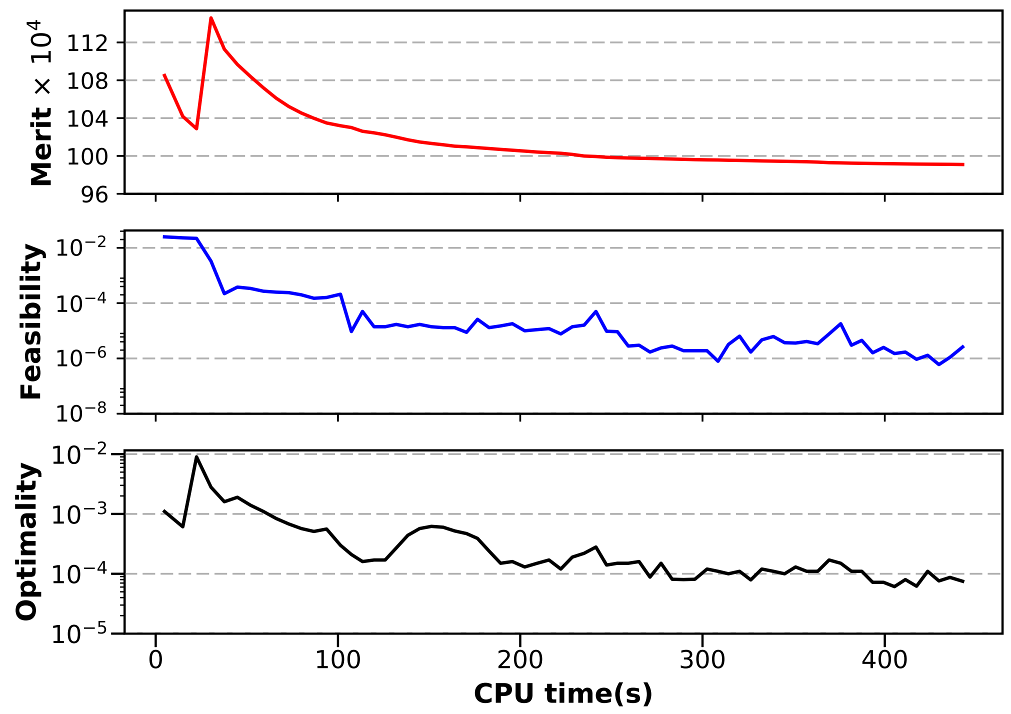


(a)

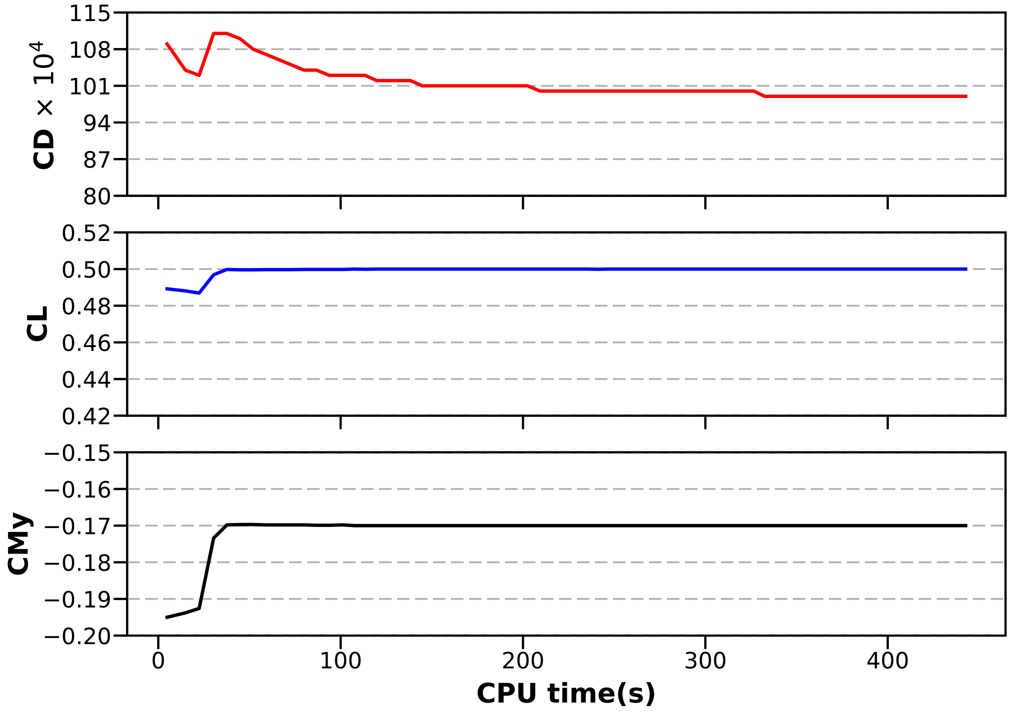


(b)

**Figure 4.13. Convergence histories for L1 grid, no. of design 480**



(a)



(b)

**Figure 4.14. Convergence histories for L1 grid, no. of design 768**

# CHAPTER 5
# CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Conclusions

When solving PDE-constrained design optimization problems, many practitioners favor a reduced-space formulation in which the state variables are implicit functions of the design variables. However, conventional optimization algorithms are not well suited to problems with many state-based constraints, such as structural stress constraints, because these algorithms require the contraint Jacobian explicilty; computing the Jacobian of state-based constraints requires solving an adjoint for each row.

In this work we have developed a matrix-free algorithm to handle reduced-space PDE-constrained optimization with state-based constraints. To cope with possibly indefinite or negative-definite Hessians, we follow the zero curve of a homotopy map using a predictor-corrector algorithm. The tangent predictor and Newton corrector linear systems are solved using a Krylov iterative method. To precondition the Krylov solver, we proposed a low-rank, approximate singular-value decomposition of the Schur complement of the KKT matrix with respect to the slacks and inequality multipliers. All the components of the algorithm — the predictor-corrector scheme, the Krylov method, and the preconditioner — require only matrix-vector products and, thus, avoid the need to compute the constraint Jacobian or Lagrangian Hessian explicitly.

The numerical experiments suggest that the matrix-free algorithm is effective. In particular, the results indicate that the algorithm can avoid stationary points that are not minimizers and that it scales well with the number of design variables. On a synthetic quadratic optimization problem the proposed algorithm was shown to be competitive with a state-of-the-art (matrix-explicit) active-set SQP algorithm. Furthermore, when applied to a difficult structural optimization problem the algorithm was able to satisfy the optimality and feasibility criteria whereas the matrix-explicit SQP algorithm failed to converge.

## 5.2 Recommendations

1. The numerical experiments suggest that, like PDEs, there are no preconditioners that work universally well for all optimization problems. This is especially true for "matrix-free" preconditioners. Therefore, it is recommended that specialized preconditioners be investigated for different types of reduced-space PDE-constrained optimization problems; for example, unique preconditioners for structural-stress constraints that scale with the problem size and for aerodynamic state-based constraints that are of fixed number.

2. During the course of this research, it was found that distinguishing between nonlinear and linear constraints is important for the construction of the matrix-free preconditioner. Unfortunately, given the current API in Kona, the current implementation of the preconditioner must apply the Lanczos SVD method to approximate the whole inequality block in (3.8), After separating the linear constraints from the nonlinear constraints, the SVD approximation will be only applied to the nonlinear part; the explicit linear Jacobians could be stored and used in the same way as conventional optimization methods. The explicit storage of the linear constraint Jacobian would mean that the algorithm is only "matrix-free" with respect to the nonlinear constraints, but the additional storage cost may be outweighed by the reduced computational cost.

3. While the homotopy-based globalization is effective, it often requires very small predictor steps and tight corrector tolerances to avoid local maximizers and saddle points. Unfortunately, following the zero curve this closely is computationally costly. It is recommended that other globalization methods be considered, including, perhaps, using no globalization but adding an a posteriori check on the curvature of the Hessian.

4. Once the algorithmic recommendations have been implemented, the aerodynamic shape optimization problem should be revisited using the RANS equations and a transonic flow. This will provide a more challenging and realistic test for the optimization algorithm.

5. In addition to comparisons with SNOPT, the proposed algorithms should be benchmarked against different types of optimization algorithms. In particular, Kona could be compared to the interior-point algorithm implemented in the IPOPT package.

6. The proposed homotopy RSNK algorithm has many parameters. While performance is insensitive to many of these parameters, there are some parameters that must be chosen carefully. To help the user, robust and automated procedures should be developed in order to set these parameters.

# REFERENCES

[1] "Climate change: Vital signs of the planet. (2018). Climate change evidence: How do we know?." [online] Available at: https://climate.nasa.gov/evidence/, Accessed Apr. 14, 2018.

[2] "Aviation and climate change : Aviation: Benefits beyond borders. (2018)." [online] Available at: https://aviationbenefits.org/environmental-efficiency/our-climate-plan/aviation-and-climate-change/, Accessed Apr. 14, 2018.

[3] J. E. Penner, *Aviation and the global atmosphere: a special report of IPCC working groups I and III in collaboration with the scientific assessment panel to the Montreal Protocol on substances that deplete the ozone layer.* Cambridge, UK: Cambridge University Press, 1999.

[4] D. Oxley and D. Goodger, "Air passenger forecasts global report," [online] Available at: https://www.iata.org/publications/Documents/global-report-sample2.pdf, Accessed Apr. 14, 2018.

[5] J. E. Green, "Civil aviation and the environment - the next frontier for the aerodynamicist," *The Aeronautical J.*, vol. 110, no. 1110, pp. 469–486, Aug. 2006.

[6] S. Skinner and H. Zare-Behtash, "State-of-the-art in aerodynamic shape optimisation methods," *Appl. Soft Computing*, vol. 62, no. 62, pp. 933–962, Sep. 2018.

[7] M. Darecki and C. Edelstenne, "Flightpath 2050 Europes Vision for Aviation," European Commission, Luxembourg, Rep. no. EUR-098-EN, 2011, [online] Available at: https://ec.europa.eu/transport/sites/transport/files/modes/air/doc/flightpath2050.pdf, Accessed Apr. 14, 2018.

[8] A. B. Lambe, G. J. Kennedy, and J. R. R. A. Martins, "Multidisciplinary design optimization of an aircraft wing via a matrix-free approach," presented at the 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf., Atlanta, GA, USA, 2014.

[9] Z. Lyu and J. R. Martins, "Aerodynamic design optimization studies of a blended-wing-body aircraft," *J. of Aircraft*, vol. 51, no. 5, pp. 1604–1617, Apr. 2014.

[10] M. Zhang, A. Rizzi, P. Meng, R. Nangia, R. Amiree, and O. Amoignon, "Aerodynamic design considerations and shape optimization of flying wings in tran-

sonic flight," presented at the 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conf., Indianapolis, IN, USA, 2012.

[11] K. Deckelnick, M. Hinze, and T. Jordan, "An optimal shape design problem for plates," *SIAM J. Numer. Anal.*, vol. 55, no. 1, pp. 109–130, Jun. 2017.

[12] K. A. James, G. J. Kennedy, and J. R. R. A. Martins, "Concurrent aerostructural topology optimization of a wing box," *Comput. Struct.*, vol. 134, pp. 1–17, Apr. 2014.

[13] L. Chen, C. Wu, and F. Sun, "Finite time thermodynamic optimization or entropy generation minimization of energy systems," *J. Non-Equilib. Thermodyn.*, vol. 24, no. 4, pp. 327–359, Dec. 1999.

[14] A. Bejan, L. Rocha, and S. Lorente, "Thermodynamic optimization of geometry: T-and Y-shaped constructs of fluid streams," *Int. J. of Thermal Sci.*, vol. 39, no. 9-11, pp. 949–960, Oct. 2000.

[15] A. Bejan and E. Mamut, *Thermodynamic optimization of complex energy systems.* Berlin, Germany: Springer Sci. & Bus. Media, 2012.

[16] M. D. O. Laboratory., [online] Available at: http://mdolab.engin.umich.edu/, Accessed Apr. 14, 2018.

[17] G. Kennedy, "Large-Scale Topology Optimization," [online] Available at: http://gkennedy.gatech.edu/research-project/large-scale-topo/, Accessed Apr. 14, 2018.

[18] S. J. Wright and J. Nocedal, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[19] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, Apr. 1999.

[20] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: an SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, pp. 99–131, Aug. 2006.

[21] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic shape optimization investigations of the common research model wing benchmark," *AIAA J.*, vol. 53, no. 4, pp. 968–985, Apr. 2015.

[22] G. K. W. Kenway and J. R. R. A. Martins, "Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration," *J. Aircraft*, vol. 51, no. 1, pp. 144–160, Jan. 2014.

[23] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins, "pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization," *Struct. Multidiscip. Optim.*, vol. 45, no. 1, p. 101–118, Jan. 2012.

[24] O. Pironneau, "On optimum design in fluid mechanics," *J. Fluid Mech.*, vol. 64, no. 1, pp. 97–110, Jun. 1974.

[25] A. Jameson, "Aerodynamic design via control theory," *SIAM J. Sci. Comput.*, vol. 3, no. 3, pp. 233–260, Jun. 1988.

[26] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders, "Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation," presented at the The 34rd AIAA Aerospace Sciences Meeting, Reno, NV, USA, 1996.

[27] A. Jameson, "Aerodynamic shape optimization using the adjoint method," in *VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese*, 2003, pp. 3–7.

[28] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide, "ADjoint: an approach for rapid development of discrete adjoint solvers," *AIAA J.*, vol. 46, no. 4, pp. 863–873, Apr. 2008.

[29] Z. Lyu, G. K. W. Kenway, C. Paige, and J. R. R. A. Martins, "Automatic differentiation adjoint of the reynolds-averaged navier-stokes equations with a turbulence model," presented at the 43rd AIAA Fluid Dynamics Conf. and Exhibit, San Diego, CA, USA, 2013.

[30] J. E. Hicken and D. W. Zingg, "Induced-drag minimization of nonplanar geometries based on the Euler equations," *AIAA J.*, vol. 48, no. 11, pp. 2564 – 2575, Nov. 2010.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar 2006.

[32] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-Scale Nonlinear Optimization.* Boston, MA, USA: Springer, 2006, pp. 35–59.

[33] Z. Lyu, Z. Xu, and J. R. R. A. Martins, "Benchmarking optimization algorithms for wing aerodynamic design optimization," presented at the 8th Int. Conf. on Computational Fluid Dynamics (ICCFD8), Chengdu, China, 2014.

[34] M. P. Rumpfkeil and D. J. Mavriplis, "Optimization-based multigrid applied to aerodynamic shape design," 2009, [online] Available at: https://pdfs.semanticscholar.org/b3bd/111b074ff343b062d43f7e8ee7ff59006dfd.pdf, Accessed Apr. 14, 2018.

[35] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1–3, pp. 503–528, Aug. 1989.

[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Berlin, Germany: Springer–Verlag, 2006.

[37] G. Biros and O. Ghattas, "Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: the Krylov-Schur solver," *SIAM J. Sci. Comput*, vol. 27, no. 2, pp. 687–713, Jul. 2005.

[38] G. Biros and O. 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," *SIAM J. Sci. Comput*, vol. 27, no. 2, pp. 714–739, Jul. 2005.

[39] E. Haber and U. M. Ascher, "Preconditioned all-at-once methods for large, sparse parameter estimation problems," *Inverse Problems*, vol. 17, no. 6, pp. 1847–1864, Dec. 2001.

[40] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton methods," *SIAM J. Numer. Anal.*, vol. 19, no. 2, pp. 400–408, Apr. 1982.

[41] S. G. Nash, "A survey of truncated-Newton methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 45–59, Dec. 2000.

[42] R. H. Byrd, F. E. Curtis, and J. Nocedal, "An inexact SQP method for equality constrained optimization," *SIAM J. Optimiz.*, vol. 19, no. 1, pp. 351–369, Jan. 2008.

[43] R. H. Byrd, F. E. Curtis, and J. Nocedal, "An inexact Newton method for nonconvex equality constrained optimization," *Math. Program.*, vol. 122, no. 2, pp. 273–299, 2010.

[44] M. Heinkenschloss and D. Ridzal, "A matrix-free trust-region SQP method for equality constrained optimization," *SIAM J. Optimiz.*, vol. 24, no. 3, pp. 1507–1541, Sep. 2014.

[45] J. E. Hicken, "Inexact Hessian-vector products in reduced-space differential-equation constrained optimization," *Optim. Eng.*, vol. 15, no. 3, pp. 575–608, Sep. 2014.

[46] A. Dener and J. E. Hicken, "Matrix-free algorithm for the optimization of multidisciplinary systems," *Struct. Multidiscip. Optim.*, vol. 56, no. 6, pp. 1429–1446, Jun. 2017.

[47] V. Akçelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, and B. van Bloemen Waanders, "Parallel algorithms for PDE-constrained optimization," in *Parallel Processing for Scientific Computing*, M. A. Heroux, P. Raghavan, and H. D. Simon,

Eds. Hingham, MA, USA: Society for Industrial and Applied Mathematics, 2006, ch. 16, pp. 291–322.

[48] M. Heinkenschloss and L. N. Vicente, "An interface optimization and application for the numerical solution of optimal control problems," *ACM Trans. Math. Softw.*, vol. 25, no. 2, pp. 157–190, Jun. 1999.

[49] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints* (Mathematical Modelling: Theory and Applications). Netherlands : Springer, 2010.

[50] A. Borzì and V. Schulz, *Computational Optimization of Systems Governed by Partial Differential Equations.* Philadelphia, PA, USA: Soc. for Ind. and Appl. Mathematics, Jan. 2011.

[51] E. L. Allgower and K. Georg, "Continuation and path following," *Acta Numerica*, vol. 2, pp. 1–64, Jan. 1993.

[52] L. T. Watson, "Globally convergent homotopy methods: A tutorial," 1986, [online] Available at: https://books.google.com/books?id=eKwpYyrADO4C, Accessed Apr. 14, 2018.

[53] J. E. Hicken and D. W. Zingg, "Globalization strategies for inexact-Newton solvers," presented at the 19th AIAA Computational Fluid Dynamics Conf., San Antonio, TX, USA, 2009.

[54] J. E. Hicken, H. Buckley, M. Osusky, and D. W. Zingg, "Dissipation-based continuation: a globalization for inexact-Newton solvers," presented at the 20th AIAA Computational Fluid Dynamics Conf., Honolulu, HI, USA, 2011.

[55] D. A. Brown and D. W. Zingg, "A monolithic homotopy continuation algorithm with application to computational fluid dynamics," *J. Comput. Physics*, vol. 321, pp. 55–75, May 2016.

[56] L. T. Watson and R. T. Haftka, "Modern homotopy methods in optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 74, no. 3, pp. 289–305, Sep. 1989.

[57] L. T. Watson, "Theory of globally convergent probability-one homotopies for nonlinear programming," *SIAM J. Optimiz.*, vol. 11, no. 3, pp. 761–780, Jul. 2006.

[58] Q. Huang, Z. Zhu, and X. Wang, "A predictor-corrector algorithm combined conjugate gradient with homotopy interior point for general nonlinear programming," *Appl. Math. Comput.*, vol. 219, no. 9, pp. 4379–4386, Jan. 2013.

[59] L. T. Watson, "Probability-one homotopies in computational science," *J. Comput. Appl. Math.*, vol. 140, no. 1, pp. 785–807, Mar. 2002.

[60] H. F. Walker, "An adaptation of Krylov subspace methods to path following problems," *SIAM J. Sci. Comput.*, vol. 21, no. 3, pp. 1191–1198, 1999.

[61] K. Georg, "A note on stepsize control for numerical curve following," in *Homotopy Methods and Global Convergence.* Boston, MA, USA: Springer, 1983, pp. 145–154.

[62] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 461–469, 1993.

[63] A. Dener and J. E. Hicken, "Matrix-free algorithm for the optimization of multidisciplinary systems," *Struct. Multidiscip. Optim.*, vol. 56, no. 6, pp. 1429–1446, Jun. 2017.

[64] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.

[65] Z. Wang, I. M. Navon, F. X. Dimet, and X. Zou, "The second order adjoint analysis: Theory and applications," *Meteorol. Atmos. Phys.*, vol. 50, no. 1-3, pp. 3–20, Mar. 1992.

[66] A. Dener, G. K. W. Kenway, Z. Lyu, J. E. Hicken, and J. R. R. A. Martins, "Comparison of inexact- and quasi-Newton algorithms for aerodynamic shape optimization," presented at the AIAA SciTech Conference, Kissimmee, FL, USA, 2015.

[67] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, vol. 14, no. 1, pp. 1–137, May 2005.

[68] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, Nov. 2002.

[69] M. Benzi and M. Tma, "A robust incomplete factorization preconditioner for positive definite matrices," *Numer. Linear Algebra*, vol. 10, no. 5-6, pp. 385–400, May 2003.

[70] J. Pestana and T. Rees, "Null-space preconditioners for saddle point systems," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 3, pp. 1103–1128, Aug. 2016.

[71] C. Lu, T. Delaney, and X. Jiao, "OPINS: An orthogonally projected implicit null-space method for singular and nonsingular saddle-point systems," Nov. 2015, [online] Available at: https://arxiv.org/abs/1511.06845, [Accessed Apr. 14, 2018].

[72] L. Bergamaschi, J. Gondzio, and G. Zilli, "Preconditioning indefinite systems in interior point methods for optimization," *Comput. Optim. Appl.*, vol. 28, no. 2, pp. 149–171, Jul 2004.

[73] R. Li, Y. Xi, and Y. Saad, “Schur complement based domain decomposition preconditioners with low-rank corrections,” May 2015, [online] Available at: https://arxiv.org/abs/1505.04340, Accessed Apr. 14, 2018.

[74] T. Rees, H. S. Dollar, and A. J. Wathen, “Optimal solvers for PDE-constrained optimization,” *SIAM J. Sci. Comput.*, vol. 32, no. 1, pp. 271–298, Jan. 2010.

[75] Q. Hu and J. Zou, “An iterative method with variable relaxation parameters for saddle-point problems,” *SIAM J. Matrix Anal. Appl.*, vol. 23, no. 2, pp. 317–338, Feb. 2001.

[76] Y. Saad, *Numerical Methods for Large Eigenvalue Problems.* Philadelphia, PA, USA: Soc. for Ind. Appl. Math., Jan. 1992.

[77] A. Dener, J. E. Hicken, P. Meng, G. J. Kennedy, J. Hwang, and J. Gray, “Kona: a parallel optimization library for engineering-design problems,” presented at the AIAA SciTech Conf., San Diego, CA, USA, 2016.

[78] J. E. Hicken and A. Dener, “A flexible iterative solver for nonconvex, equality-constrained quadratic subproblems,” *SIAM J. Sci. Comput.*, vol. 37, no. 4, pp. A1801–A1824, Jul. 2015.

[79] A. Dener, “A modular matrix-free approach to multidisciplinary design optimization,” Ph.D. dissertation, Dept. Mech. Aero. Nucl. Eng., Rensselaer Polytechnic Inst., Troy, NY, USA, Nov. 2017.

[80] P. E. Gill and E. Wong, “User’s guide for SNOPT version 7.6: Software for large-scale nonlinear programming,” Jan. 2017, [online] Available at: http://www.sbsi-sol-optimize.com/manuals/SNOPT%20Manual.pdf, Accessed Apr. 14, 2018.

[81] D. Orban, “CUTEr Test Problem Set,” [online] Available at: http://www.cuter.rl.ac.uk/index.html, Accessed Apr. 14, 2018.

[82] N. I. M. Gould, D. Orban, and P. L. Toint, “Cuter and sifdec: A constrained and unconstrained testing environment, revisited,” *ACM Trans. Math. Softw.*, vol. 29, no. 4, pp. 373–394, Dec. 2003.

[83] D. Orban, “Cuter website. (2018).” [online] Available at: http://www.cuter.rl.ac.uk/interfaces.html, Accessed Apr. 14, 2018.

[84] A. R. Conn, N. I. M. Gould, and P. L. Toint, “The SIF reference report,” in *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Berlin, Heidelberg, Germany: Springer, 1992, pp. 180–243, [online] Available at: https://doi.org/10.1007/978-3-662-12211-2_7, Accessed Apr. 14, 2018.

[85] “Cuter test problem list,” [online] Available at: http://www.cuter.rl.ac.uk/Problems/mastsif.shtml, Accessed Apr. 14, 2018.

[86] H. B. Lana Osusky and D. W. Zingg, "CRM Wing by Aerodynamic Design Optimization Discussion Group," [online] Available at: https://info.aiaa.org/tac/ASG/APATC/AeroDesignOpt-DG/Test%20Cases/ADODG%20Case%204%20CRM%20Wing.pdf, Accessed Apr. 14, 2018.

[87] AIAA, "Aerodynamic Design Optimization Discussion Group," [online] Available at: https://info.aiaa.org/tac/ASG/APATC/AeroDesignOpt-DG/default.aspx, Accessed Apr. 14, 2018.

[88] G. K. W. Kenway, G. J. Kennedy, and J. R. R. A. Martins, "A CAD-free approach to high-fidelity aerostructural optimization," presented at the Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conf., Fort Worth, TX, USA, 2010.

[89] "SUmb Manual from Mdolab ," [online] Available at: http://mdolab.engin.umich.edu/sites/default/files/SUmb_manual.pdf, Accessed Apr. 14, 2018.

[90] "Python Interface to CUTEr," [online] Available at: http://fides.fe.uni-lj.si/~arpadb/software-pycuter.html, Accessed Apr. 14, 2018.

[91] W. Hock and K. Schittkowski, "Test examples for nonlinear programming codes," *J. Optim. Theory and Appl.*, vol. 30, no. 1, pp. 127–129, Jan 1980.

# APPENDIX A
# SUPPLEMENTS ON THE CUTER PROBLEMS AND PARAMETERS TABLE

## A.1 Brief Overview of the CUTEr Test Problems

For each test problem, CUTEr provides access to the objective function and the constraint functions, as well as their derivatives.

The constraints include bound constraints on the variable and other types of constraints. The variable $x$ are subject to simple bounds:

$$\mathrm{bl}_i \leq x_i \leq \mathrm{bu}_i$$

where $\mathrm{bl}_i$ and $\mathrm{bu}_i$ are the lower and upper bound on $x_i$. When there is no lower or upper bound on $x_i$, then $\mathrm{bl}_i = -10^{20}$ or $\mathrm{bu}_i = 10^{20}$.

With the exception of the bound constraints, all remaining constraints are gathered in a single vector-valued function $c(x) \in \mathbb{R}^m$, which is then bounded as follows:

$$\mathrm{cl}_i \leq c_i(x) \leq \mathrm{cu}_i$$

where one of $\mathrm{cl}_i$ or $\mathrm{cu}_i$ is always around $10^{20}$. This means that the inequality constraint must take just one of the following two forms on a given problem:

$$c_i(x) \geq 0 \quad \text{or} \quad c_i(x) \leq 0.$$

For equality constraints $\mathrm{cl}_i$ and $\mathrm{cu}_i$ are both equal to 0. There is also a bool vector, Equatn $\in \mathbb{R}^m$, indicating whether a constraint is an equality constraint or not.

It is possible to instruct CUTEr to order equality constraints before inequality constraints, or linear constraints before nonlinear constraints, by reordering the components of $c(x)$. Likewise, components of $x$ can be reordered such that nonlinear variables appear before linear ones.

In addition, CUTEr can also output the Lagrangian function value, the gradient of the objective function and the Lagrangian, the Jacobian matrix and the Hessian matrix of the constraints.

## A.2 Kona-CUTEr Interface

By using a Python interface to CUTEr [90], one can access the test problems in Python environment and build the Kona-CUTEr interface. As described in [77], a Kona solver interface essentially asks for matrix-vector products with the PDE-Jacobian (if any), and matrix-vector and vector-matrix products with the Jacobians of the equality and inequality constraints. The explicit Jacobian matrices from CUTEr can be readily used to calculate their product with an arbitrary incoming vector for Kona. CUTEr problems are still valuable to verify Kona's accuracy, as well as its capability to overcome other numerical difficulties, including non-convex problems.

## A.3 Problem Classification

Each problem in CUTEr is classified following the Hock and Schittkowski scheme [91] by the string:

$$X\,X\,X\,r\ -\ X\,X\ -\ n\ -\ m$$

The first character defines the problem objective function type, with the following options:

- **U**: undefined,
- **C**: constant,
- **L**: linear,
- **Q**: quadratic,
- **S**: sum of squares,
- **O**: none of the above.

The second character defines the constraint function type, with the options:

- **U**: unconstrained,
- **X**: fixed variables,

- **B**: bounds on the variables,
- **N**: adjacency matrix of a linear network,
- **L**: linear,
- **Q**: quadratic,
- **O**: more general constraints.

The third character shows the smoothness of the problem, with the option of:

- **R**: the problem is regular, and its first and second derivatives exist and continuous everywhere,
- **I**: the problem is irregular.

The third character **r** holds an integer among 0, 1 and 2, indicating the highest derivatives provided analytically.

The first character after the hyphen indicates the origin of the problem, with the option of:

- **A**: the problem is academic, mainly used by researchers to test algorithms;
- **M**: the problem is a modeling exercise, with the solution not used in practical application;
- **R**: the solution of the problem has been used in a real application.

The next character has an option of:

- **Y**: the problem contains internal variables,
- **N**: the problem does not contain internal variables.

The last two characters have the following options:

- **n** - **m**: the number of variables and constraints (fixed variables and bound constraints excluded),
- **V** - **V**: an integer chosen by the user among the given list of fixed numbers.

## A.4 Parameter Table

**Table A.1. Parameters used in the test problems**

| **Parameters** | **Default** | **Range** | **Sphere** | **Nonconvex** | **Quadratic** | **CUTEr** | **Structural** | **ASO** |
|---|---|---|---|---|---|---|---|---|
| Predictor-Corrector Algorithm | | | | | | | | |
| $K_{\max}$ | 100 | $\geq$100 | 500 | 100 | 100 | 500 | 100 | 100 |
| $J_{\max}$ | 2 | $\geq$2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\epsilon_F$ | 1e-6 | [1e-8, 1e-3] | 1e-7 | 1e-7 | 1e-7 | 1e-7 | 1e-4 | 1e-3 |
| $\tau$ | 0.1 | [0.1,0.5] | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| $\epsilon_H$ | 0.1 | [0.1,0.5] | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| $\alpha_0$ | 0.05 | $\geq$0.01 | 0.05 | 0.05 | [40,60,80,100,120] | *[4] | 0.05 | 0.05 |
| $\delta_{\text{targ}}$ | 1.0 | [1.0,10] | 1 | 10 | 10 | * | 1.0 | 10 |
| $\phi^{\circ}_{\text{targ}}$ | 10.0 | [5.0,50] | 10 | 10 | 20 | * | 10 | 20 |
| $\zeta_{\max}$ | 50 | [10,50] | 50 | 50 | 50 | 50 | 50 | 50 |
| $\zeta_{\min}$ | 0.5 | 0.5 | 0.001 | 0.001 | 0.001 | 0.001 | 0.5 | 0.001 |
| $\Delta\mu_{\max}$ | -5e-4 | [-5e-4, -5e-1] | -5e-4 | -5e-4 | -5e-4 | -5e-4 | -5e-4 | -5e-4 |
| $\Delta\mu_{\min}$ | -0.9 | -0.9 | -0.9 | -0.9 | -0.9 | -0.9 | -0.9 | -0.9 |
| $s_0$ | $\mathbf{e}$[5] | $> 0$ | $g(x_0)$ | 5$\mathbf{e}$ | 10$\mathbf{e}$ | 10$\mathbf{e}$ | $g(x_0)$ | $g(x_0)$ |
| $\tau_s$ | 1e-6 | 1e-6 | 1e-6 | 1e-6 | 1e-6 | 1e-6 | 1e-6 | 1e-6 |
| Preconditioner | | | none | none | | | | |
| $n_{\mathbf{\Sigma}}$ | 5 | $\geq$2 | - | - | 2 | * | [20,80,320] | 20 |

[4]*: provided in Table 4.1
[5] $\mathbf{e} = [1, 1, ..., 1]$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\beta$ | 1.0 | >0 | - | - | - | - | 0.1 | - |
| $N_{\text{bfgs}}$ | 10 | [1, 20] | - | - | 10 | 10 | - | 20 |
| $\mu_e$ | -1 | [0, 1] | - | - | - | - | 1e-3 | 1e-3 |
| $\Sigma_e$ | 1 | [0, 1] | - | - | - | - | 1e-3 | - |
| Krylov Iterative Solver | | | | | | | | |
| $n_k$ | 20 | [10,30] | 20 | 20 | 20 | 20 | 20 | 20 |
| $\epsilon_{\text{krylov}}$ | 1e-2 | [0, 0.1] | 1e-2 | 1e-2 | 1e-2 | 1e-2 | 1e-4 | 1e-2 |