AIAA SciTech Forum
5-9 January 2015, Kissimmee, Florida
53rd AIAA Aerospace Sciences Meeting

10.2514/6.2015-1945

Comparison of inexact- and Quasi-Newton Algorithms for Aerodynamic Shape Optimization

Alp Dener*, Gaetan K. W. Kenway†, Zhoujie Lyu‡,
Jason E. Hicken§ and Joaquim R. R. A. Martins¶

Large-scale aerodynamic and multidisciplinary design problems challenge conventional optimization algorithms, because these problems typically involve thousands of design variables and constraints. Alternative algorithms must be developed that produce solutions to large-scale design problems in a reasonable time. To this end, we investigate a scalable reduced-space Newton–Krylov optimization algorithm. This inexact-Newton algorithm uses a novel matrix-free Krylov solver that requires only KKT-matrix-vector products; these products are formed by solving two linear PDEs. We present preliminary results that benchmark the inexact-Newton algorithm against a conventional quasi-Newton algorithm on the Euler-based drag minimization of the Common-Research-Model wing benchmark. For this problem, the preliminary results indicate that the inexact-Newton algorithm scales well and outperforms the conventional algorithm for problems with more than 500 design variables.

I. Introduction

Adjoint-based aerodynamic shape optimization (ASO) was first introduced to the applied aerodynamics community by Jameson over 25 years ago [1]. Since that time, adjoint methods have matured significantly, and the steps necessary to "differentiate" a flow solver are now well understood. Even some commercial computational fluid dynamics (CFD) libraries now include adjoint capabilities[a].

As is well known, the computational cost of the adjoint-based gradient is virtually independent of the design-space dimension. This has encouraged researchers to explore problems with increasingly large numbers of design variables. For aircraft, which are governed by complex physics, this design flexibility has the potential to dramatically improve old concepts and reveal new ones.

Unfortunately, expanding the dimension of the design space is a doubled-edged sword, and the increased design flexibility comes at a price: the optimization problems themselves become harder to solve. While the adjoint-based gradient enables inexpensive evaluation of the first-order optimality conditions, it does not influence the scalability of the gradient-based optimization algorithm itself[b]. Thus, as ASO problems grow ever larger, the bottleneck becomes the scalability of the optimization algorithm. This is the primary motivation for the present study.

State-of-the-art algorithms for ASO problems use quasi-Newton approximations for the Hessian and explicit (i.e., stored) constraint Jacobians. We will refer to this implementation as the conventional approach. This class of algorithm has proven to be highly effective for modest-sized problems with a few constraints, but it exhibits the following short-comings for large-scale problems with greater than 10^3 design variables:

- Constraints that depend on the state variables require separate adjoints to populate the Jacobian. For problems with many state-based constraints, the cost of these adjoints becomes prohibitive. Constraint

*PhD Student, Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, AIAA Student Member
†Postdoc Research Fellow, Department of Aerospace Engineering, University of Michigan, AIAA Student Member
‡Postdoc Research Fellow, Department of Aerospace Engineering, University of Michigan, AIAA Student Member
§Assistant Professor, Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, AIAA Member
¶Associate Professor, Department of Aerospace Engineering, University of Michigan, AIAA Associate Fellow
[a]Specifically, ANSYS 14.5, http://www.ansys.com/Products/ANSYS+14.5+Release+Highlights/
[b]By scalability, we mean the cost as measured by the number of PDE solves, assuming an adjoint-based gradient.

aggregation methods provide one means of ameliorating the number of adjoint solutions required [2]; however, aggregation presents its own difficulties, namely inaccurate representation of the constraints and poorly conditioned optimization problems [3].

- Quasi-Newton methods become less effective as the design dimension grows. For large-scale problems, limited-memory quasi-Newton methods, such as L-BFGS [4], are often used, and these have a linear, rather than super-linear, asymptotic convergence rate. Another drawback of quasi-Newton methods is their reliance on steepest-descent directions initially and when restarting. This often leads to many iterations in the line-search or trust-region subproblems.

Some alternatives to conventional sequential quadratic optimization (SQO) have been explored by the ASO community. In particular, one-shot methods have demonstrated the potential for rapid convergence on large-scale problems [5–7]; see also the monograph [8]. One-shot ASO algorithms, also known as all-at-once or full-space algorithms, include the CFD residual as a constraint in the optimization statement. The first-order optimality conditions are then solved using either Newton–Krylov [9, 10] or multigrid [7, 8] algorithms.

A potential drawback with one-shot approaches is that the CFD and optimization algorithms must be tightly integrated, making it difficult to "swap" CFD solvers between applications. One-shot algorithms are also less attractive for unsteady ASO, where the state constraint can be several orders of magnitude larger than the steady problem.

Another alternative to conventional, and the one we highlight in this work, is the class of reduced-space Newton–Krylov (RSNK) algorithms [8, 11–14]. RSNK algorithms have excellent asymptotic convergence. When properly globalized, they also perform well initially, because they capture the curvature of the Lagrangian. Moreover, they can be implemented using matrix-free Krylov methods, so RSNK algorithms do not require the Hessian or constraint Jacobian to be computed and stored explicitly.

Despite their attractive properties, RSNK methods are not commonly used by the ASO community. One reason for this is that few optimization algorithms implement (fully) matrix-free RSNK algorithms. The adoption of RSNK methods may also be hindered by a perception that they are difficult to implement. One of our objectives in this work is to demonstrate that RSNK algorithms can be implemented using the same software routines required by conventional SQO libraries.

The remainder of the paper is organized as follows. Section II provides a high-level overview of the two optimization algorithms we are comparing: the benchmark SNOPT library and Kona, our in-house RSNK library. The efficient evaluation of matrix-vector products plays an important role in RSNK algorithms, and Section III is devoted to this topic. Section IV defines the ASO problem we use to compare the performance of the two algorithms. Results are presented in Section V, and conclusions and future work can be found in Section VI.

II. Optimization Algorithms

In this section we describe the two optimization algorithms used in this work. To facilitate these descriptions, we need to introduce some notation. Consider the following generic equality-constrained optimization problem:

$$\begin{aligned} \min_{x} \quad & f(x, u(x)) \\ \text{subject to} \quad & c(x, u(x)) = 0. \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ are the design variables and $u \in \mathbb{R}^s$ are the state variables. We assume throughout that the objective function, $f : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}$, and constraints, $c : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^n$, are C^2 continuous functions. The state variables are assumed to be an implicit function of the design variables via a discretized set of PDEs, denoted here by

$$R(x, u(x)) = 0. \tag{2}$$

Note that we are restricted to equality-constrained problems in this study, because the current RSNK algorithm cannot handle inequality constraints. Future work will consider more general inequality constraints.

To define the first-order optimality conditions for (1), we introduce the Lagrangian

$$L(x, \lambda) \equiv f(x) + \lambda^T c(x),$$

where $\lambda \in \mathbb{R}^m$ are the Lagrange multipliers associated with the equality constraints. In defining L, we have dropped the explicit dependence of f and c on the states, since $u = u(x)$. The first-order optimality conditions follow by differentiating L with respect to x and λ, yielding

$$\begin{aligned} g(x,\lambda) \equiv \nabla_x L = \nabla_x f + \lambda^T \nabla_x c = 0, \\ c(x) = 0, \end{aligned} \tag{3}$$

where $g(x,\lambda)$ is the gradient of the Lagrangian and ∇_x denotes the total-derivative operator with respect to x.

In the context of PDE-constrained optimization, the total-derivative of an arbitrary function $h(x,\lambda,u(x))$ can be evaluated efficiently by solving an adjoint problem. Using this approach, the total derivative of h with respect to x can be computed by evaluating

$$\nabla_x h = \frac{\partial h}{\partial x} + \psi^T \frac{\partial R}{\partial x}$$

where $\psi \in \mathbb{R}^s$ are adjoint variables that satisfy the linear PDE

$$\frac{\partial h}{\partial u} + \psi^T \frac{\partial R}{\partial u} = 0. \tag{4}$$

In the above equations, $\partial/\partial x$ and $\partial/\partial u$ denote the partial derivative operators with respect to x and u, respectively.

At this point, we highlight an important observation. The total-derivative of the Lagrangian can be computed in two mathematically equivalent ways: 1) by solving a single adjoint problem to compute $\nabla_x L$, or; 2) by solving $m+1$ adjoints associated with $\nabla_x f$ and the m equality constraints $\nabla_x c$. The first approach is used in RSNK, while the second approach is adopted in conventional optimization algorithms. Although the first approach is the most efficient for computing the gradient of the Lagrangian, the second approach can be more efficient overall if the cost of computing the constraint Jacobian, $\nabla_x c$, is compensated by the valuable information it provides, e.g. the basis for the null space.

The two optimization algorithms considered in this work use the sequential quadratic optimization (SQO) paradigm, which we briefly review now. At iteration k, the trial design update $p \in \mathbb{R}^n$ is based on the solution of the following quadratic optimization problem:

$$\begin{aligned} \min_p \quad & f_k + g_k^T p + \frac{1}{2} p^T \mathsf{W}_k p \\ \text{subject to} \quad & \mathsf{A}_k p + c_k = 0, \end{aligned} \tag{5}$$

where $\mathsf{A} \equiv \nabla_x c$ is the (total) Jacobian of the constraints, and $\mathsf{W} \equiv \nabla^2_{xx} L$ is the (total) Hessian of the Lagrangian, or an approximation to it. The subscript k indicates that a particular function or derivative is evaluated at the current primal-dual solution (x_k, λ_k). Note that W may not be convex in the null space of A, in which case (5) is modified; however, this modification is algorithm dependent.

II.A. Benchmark Algorithm: SNOPT

SNOPT (sparse nonlinear optimizer) is an implementation of the SQO method suitable for general nonlinear constrained problems [15]. SNOPT is capable of solving large-scale nonlinear optimization problems with thousands of constraints and design variables. The search directions are determined by minimizing a quadratic model of the Lagrangian function subject to linearized constraints, i.e. the SQO paradigm. To globalize the algorithm, a line search seeks to reduce an augmented Lagrangian merit function to ensure convergence to a local optimum from any starting point. To form the quadratic subproblem, SNOPT approximates the Hessian of the Lagrangian using semi-definite quasi-Newton approach. For all optimizations performed with SNOPT, the full memory BFGS quasi-Newton method is used.

We have used SNOPT extensively in the past to solve aerospace engineering design problems, such as aerodynamic shape optimization [16–18], aerostructural design optimization of aircraft configurations [19–21], and satellite design [22]. SNOPT is also a suitable benchmark, because it uses a conventional SQO algorithm, requiring a separate adjoint for each row of the Jacobian and employing a quasi-Newton method.

Note that, although SNOPT is capable of solving problems with general inequality constraints using an active set method, we apply it to an equality-constrained problem in this study. This is necessary in order to have a fair comparison with the RSNK algorithm, which currently accommodates only equality constraints.

II.B. Reduced-space Newton–Krylov Algorithm

We begin our description of the Reduced-Space Newton–Krylov (RSNK) algorithm by assuming that W is positive-definite in the null space of A. In this convex case, solving the QO subproblem (5) reduces to solving the following saddle-point problem for the primal-dual update (p, d):

$$\begin{bmatrix} \mathsf{W}_k & \mathsf{A}_k^T \\ \mathsf{A}_k & 0 \end{bmatrix} \begin{bmatrix} p \\ d \end{bmatrix} = - \begin{bmatrix} g_k \\ c_k \end{bmatrix},$$

or, more concisely,

$$\mathsf{K}s = b, \tag{6}$$

where K denotes the primal-dual, or KKT, matrix.

At iteration k, the RSNK algorithm finds an inexact solution to (6) using FLECS [23], a Krylov-based linear solver for nonconvex saddle-point problems. A significant advantage of Krylov-based solvers like FLECS is that they can be implemented matrix-free, since they require only matrix-vector and preconditioning operations. An efficient implementation of the KKT-matrix-vector products is described in detail in Section III. Preconditioning, while important for Krylov-subspace methods in general, was not used for the ASO problem considered in this work.

The QO subproblem (5) must be modified when the Lagrangian Hessian, W, is not positive definite in the null space of the constraint Jacobian. The FLECS linear solver handles nonconvex subproblems by including a trust-region constraint, which prevents unbounded steps. Finally, the QO subproblems are embedded in a trust-region framework [24], and a basic filter [25] is used to globalize the method.

While the RSNK algorithm is admittedly more complex than conventional SQO algorithms, most of this complexity can be hidden from the end user. For example, in our implementation of RSNK in the library Kona, the user need only provide the operations listed in Table 1. Most of these operations exist as, or can be adapted from, components already implemented in conventional ASO algorithms.

III. Computing KKT-matrix-vector Products

This section explains how KKT-matrix-vector products are evaluated in the optimization library Kona. Our goal is to provide an intuitive explanation of how the products are formed by drawing a connection between the finite-difference-based approach and the adjoint-based approach. We emphasize that this explanation is for pedagogical purposes only, since the details concerning the KKT-matrix-vector products are hidden from the user; recall, the user need only implement the operations listed in Table 1.

III.A. Forming Products Using a Finite-difference

We begin by describing the finite-difference (FD) approximation of the KKT-matrix-vector product. Let x_k and λ_k denote the current estimates of the primal and dual solutions, respectively, and consider an arbitrary primal-dual vector $v = (v_p^T, v_d^T)^T \in \mathbb{R}^{n+m}$. Then a product between K and v can be evaluated using the first-order forward difference

$$\begin{bmatrix} \mathsf{W}_k & \mathsf{A}_k^T \\ \mathsf{A}_k & 0 \end{bmatrix} \begin{bmatrix} v_p \\ v_d \end{bmatrix} \approx \frac{1}{\epsilon} \begin{bmatrix} g(x_k + \epsilon v_p, \lambda_k + \epsilon v_d) - g(x_k, \lambda_k) \\ c(x_k + \epsilon v_p) - c(x_k) \end{bmatrix}, \tag{7}$$

where $\epsilon > 0$ is the step-size. As usual, the FD approximation is plagued by the step-size dilemma: what is the optimal ϵ that balances truncation and round-off errors?

Forward-difference approximations of matrix-vector products are often used in Jacobian-free Newton–Krylov (JFNK) methods for PDEs; see, for example, the review by Knoll and Keyes [26]. Unlike PDE residuals, which typically depend explicitly on the state variables, the first-order optimality conditions depend implicitly on the primal-dual variables, x and λ, via the state and adjoint, u and ψ. Consequently, in order to evaluate the perturbed gradient and constraints that appear in (7), we must resolve the (nonlinear) state PDE and adjoint PDE at $(x_k + \epsilon v_p, \lambda_k + \epsilon v_d)$.

The need to solve two perturbed PDEs is not a disadvantage of the FD approximation per se: solving two additional PDEs for the KKT-matrix-vector product is unavoidable. However, problems do arise when the FD approximation is implemented in a naïve manner. For example, the FD step-size dilemma is especially

Table 1. Operations required by the Kona library for RSNK optimization.

Operation	Remarks
Vector operations: for $a, b \in \mathbb{R}$ evaluate $z = ax + by, \quad x, y, z \in \mathbb{R}^n$, $\lambda = a\mu + b\nu, \quad \lambda, \mu, \nu \in \mathbb{R}^m$, $w = au + bv, \quad u, v, w \in \mathbb{R}^s$.	These basic linear algebra tasks are performed outside the optimizer by the user to permit generic parallelization.
Inner products: for $x, y \in \mathbb{R}^n$, $\mu, \nu \in \mathbb{R}^m$, and $u, v \in \mathbb{R}^s$ evaluate the inner products $\langle x, y \rangle_n, \quad \langle \mu, \nu \rangle_m, \quad \langle u, v \rangle_s$.	The notation $\langle , \rangle$ indicates suitable inner product on the indicated spaces. These operations are also needed to enable parallelization.
Function evaluations: for $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^s$ evaluate and store the functions $f(x,u), \quad \nu = c(x,u), \quad w = R(x,u)$.	These are used when evaluating convergence criteria.
Objective derivatives: evaluate the objective function partial derivatives at the design-state pair (x, u): $y = \frac{\partial f}{\partial x}, \quad v = \frac{\partial f}{\partial u}$.	The objective gradients must be evaluated at arbitrary design-state pairs; these are partial derivatives, not total derivatives.
Matrix-vector products: evaluate Jacobian-vector products, linearized about (x, u), $v = \frac{\partial R}{\partial x} y, \quad y = \left(\frac{\partial R}{\partial x}\right)^T v, \quad y \in \mathbb{R}^n, v \in \mathbb{R}^s$ $v = \frac{\partial R}{\partial u} w, \quad w = \left(\frac{\partial R}{\partial u}\right)^T v, \quad v, w \in \mathbb{R}^s$ $\mu = \frac{\partial c}{\partial x} y, \quad y = \left(\frac{\partial c}{\partial x}\right)^T \mu, \quad y \in \mathbb{R}^n, \mu \in \mathbb{R}^m$ $\mu = \frac{\partial c}{\partial u} w, \quad w = \left(\frac{\partial c}{\partial u}\right)^T \mu, \quad w \in \mathbb{R}^s, \mu \in \mathbb{R}^m$.	These products can be evaluated using algorithmic differentiation (AD) or, in the case of the forward products, the complex-step method.
PDE solves: solve the following systems for u, w, and ϕ: $R(x,u) = 0, \quad \left(\frac{\partial R}{\partial u}\right) w = u, \quad \left(\frac{\partial R}{\partial u}\right)^T \phi = v$.	The linear systems are necessary to compute the (total) gradient of the Lagrangian, as well as the KKT-matrix-vector products.

difficult to resolve when the primal and dual variables have disparate scaling. Moreover, a black-box FD implementation of (7) may result in the computationally expensive operation of re-assembling the PDE Jacobian, its preconditioner, or both. These types of overhead expenses must be avoided to achieve efficient KKT products.

III.B. Forming Products Using Second-order Adjoints

This brings us to the adjoint-based approach to forming the KKT-matrix-vector product, which is closely related to the FD product. To see this relationship, consider solving the nonlinear PDE at the perturbed design $x_k + \epsilon v_p$. Let $u_k + \epsilon w$ denote the solution corresponding to the perturbed PDE, where $w \in \mathbb{R}^s$ and ϵ is the same scalar used in the design perturbation. For infinitesimal ϵ, we can linearize the PDE about (x_k, u_k):

$$R(x_k + \epsilon v_p, u_k + \epsilon w) = R(x_k, u_k) + \epsilon \frac{\partial R}{\partial x} v_p + \epsilon \frac{\partial R}{\partial u} w = 0.$$

Recognizing that $R(x_k, u_k) = 0$, we can rearrange the above equation to find a linear PDE for w:

$$\frac{\partial R}{\partial u} w = -\frac{\partial R}{\partial x} v_p, \tag{8}$$

where we have canceled the common ϵ factor. Equation (8) is solved in lieu of the perturbed nonlinear PDE to find the perturbation to the state u_k. An advantage of this approach is that the PDE Jacobian, the matrix on the left side of (8), is evaluated at (x_k, u_k). Therefore, the PDE Jacobian, and any corresponding preconditioner, does not need to be reformed.

An analogous process can be followed to find the perturbation to the adjoint. Let the adjoint PDE for the total derivative of the Lagrangian be denoted by

$$S(x,\lambda,u,\psi) = \left(\frac{\partial L}{\partial u}\right)^T + \left(\frac{\partial R}{\partial u}\right)^T \psi.$$

In addition, let $\psi_k + \epsilon\phi$ be the solution to the adjoint PDE evaluated at the perturbed variables $x_k + \epsilon v_p$, $\lambda_k + \epsilon v_d$, and $u_k + \epsilon w$. Then, linearizing the adjoint PDE about $(x_k, \lambda_k, u_k, \psi_k)$, we find

$$S(x_k + \epsilon v_p, \lambda_k + \epsilon v_d, u_k + \epsilon w, \psi_k + \epsilon\phi) = S(x_k,\lambda_k,u_k,\psi_k) + \epsilon\frac{\partial S}{\partial x}v_p + \epsilon\frac{\partial S}{\partial \lambda}v_d + \epsilon\frac{\partial S}{\partial u}w + \epsilon\frac{\partial S}{\partial \psi}\phi = 0$$

As with the nonlinear PDE, the adjoint PDE vanishes at the current solution, $S(x_k,\lambda_k,u_k,\psi_k) = 0$. Using this fact, and simplifying some of the products in the linearization, we arrive at the following PDE for ϕ:

$$\left(\frac{\partial R}{\partial u}\right)^T \phi = -\left(\frac{\partial c}{\partial u}\right)^T v_d - \begin{bmatrix}\frac{\partial S}{\partial x} & \frac{\partial S}{\partial u}\end{bmatrix}\begin{bmatrix} v_d \\ w\end{bmatrix} \tag{9}$$

Solving this PDE for ϕ provides the perturbation to the adjoint of the Lagrangian corresponding to the desired KKT product.

The second term on the right side of (9) involves second-partial derivatives with respect to x and u. In Kona, this term is approximated using a forward difference by evaluating the adjoint PDE at $(x_k + \epsilon v_p, \lambda_k, u_k + \epsilon w, \psi_k)$ and taking the difference with $S(x_k,\lambda_k,u_k,\psi_k)$. Note that, while the adjoint PDE residual is zero analytically, its numerical value is typically nonzero when iterative methods are used to solve for ψ. Thus, neglecting $S(x_k,\lambda_k,u_k,\psi_k)$ in the FD approximation can lead to significant errors in the KKT product.

The penultimate step in computing the KKT-matrix-vector product is to evaluate the first-order optimality at the perturbed design, dual, state, and adjoint. Subsequently, the forward-difference approximation between the perturbed and unperturbed optimality conditions yields the desired product. Thus, the differences in (7) become

$$\begin{aligned}\frac{1}{\epsilon}\left[g(x_k + \epsilon v_p, \lambda_k + \epsilon v_d) - g(x_k,\lambda_k)\right] \\ = \frac{1}{\epsilon}\left[\frac{\partial L}{\partial x}(x_k + \epsilon v_p, \lambda_k + \epsilon v_d, u_k + \epsilon w) + (\psi_k + \epsilon\phi)^T \frac{\partial R}{\partial x}(x_k + \epsilon v_p, \lambda_k + \epsilon v_d, u_k + \epsilon w)\right. \\ \left. - \frac{\partial L}{\partial x}(x_k,\lambda_k,u_k) + \psi_k^T \frac{\partial R}{\partial x}(x_k,\lambda_k,u_k)\right],\end{aligned} \tag{10}$$

and

$$\frac{1}{\epsilon}\left[c(x_k + \epsilon v_p) - c(x_k)\right] \approx \frac{\partial c}{\partial x}v_p + \frac{\partial c}{\partial u}w. \tag{11}$$

In Equation (10), the differences appearing on the right-hand side are explicit, in the sense that no additional PDEs must be solved. This is because the effect of the design and dual perturbations on the state and adjoint are already accounted for by w and ϕ. The variables w and ϕ are sometimes called second-order adjoints in the literature [27]. Further details on using second-order adjoints to form Hessian-vector products can found in Refs. [8, 13, 14, 28, 29].

In summary, the process of computing $\mathsf{K}v$ proceeds as follows. First, the second-order adjoint w is found by solving (8). Subsequently, w is used to form the right-hand-side of (9), which is then solved to find the second-order adjoint ϕ. Finally, these adjoints are used to find the matrix-vector product by evaluating (10) and (11).

III.B.1. Some Remarks on Implementation

We conclude this section with remarks on the practical implementation of the KKT-matrix-vector products.

- From the user's perspective, a notable difference between an RSNK algorithm and a conventional algorithm is the need to solve the linearized and adjoint PDEs with arbitrary right-hand sides. Fortunately,

the adjoint solves required by conventional ASO algorithms can easily be adapted for the systems involving the transposed Jacobian; only the right-hand-side must change. In addition, the systems that involve the PDE Jacobian itself are analogous to the systems solved in JFNK methods; again, only the right-hand-side must change.

- The ϵ perturbation used in the adjoint-based product must be the same throughout the computation. For example, as mentioned earlier, a forward difference is used to compute the second term on the right-hand side of (9). This forward difference should use the same ϵ used later in the difference (10). This is consistent with the connection between this product and the full FD product.

- Matrix-free products with the PDE Jacobian are important for efficiency and accuracy, and should be provided analytically or using AD (we use the latter here). To appreciate this need, consider the right-hand side of (9), which requires that we evaluate the adjoint residual S. This adjoint residual is needed each time a KKT product is computed; therefore, if the evaluation of S re-assembles the Jacobian of the PDE explicitly, the overhead can adversely impact the efficiency of the RSNK algorithm.

IV. Optimization Problem Description

To evaluate the RSNK algorithm against the conventional SQO approach, we chose a variation of the common-research-model (CRM) wing benchmark developed by the AIAA Aerodynamic Design Optimization Discussion Group (ADODG). The problem consists of the lift- and moment-constrained drag minimization of the CRM wing. The geometry and specifications are given in the ADODG document[c]. This problem has been previously investigated by several researchers [30–33], and optimized geometries and meshes are publicly available [30].

The original ADODG CRM-wing optimization problem statement was modified for the present work. These modifications are summarized and justified below.

- Some of the matrix-vector products listed in Table 1 were not yet available as matrix-free products for the RANS equations. As mentioned earlier, without matrix-free products, the RSNK algorithm will not be competitive. Consequently, the present study was limited to the Euler equations.

- The optimization problem based on the Euler equations proved to be poorly conditioned, i.e., the condition number of K was high. We identified two sources of this ill conditioning. The first source is that the inviscid optimization is ill posed: with control over both aerodynamic and geometric twist, the optimal elliptical loading can be obtained in an infinite number of ways. Evidence for this "flat" design space was previously observed in the case of the RANS equations by performing random-start optimization runs using SNOPT [30]; the optimal solutions had approximately the same optimum drag coefficient, but varied in their geometries. The second source of ill-conditioning is the presence of shocks at the transonic Mach number of $M = 0.85$.

  Preconditioning the FLECS solver may help alleviate the ill-conditioning observed in this problem; however, although matrix-free preconditioners are currently under development, they were not available for this work.

  In the short term, the ill-conditioning was ameliorated by 1) considering a subsonic flow with a Mach number of $M = 0.65$, and 2) adding a regularization term to the objective function. The regularization is described in greater detail below.

- Finally, as mentioned earlier, Kona does not currently implement inequality constraints. This is not a problem for the lift and moment constraints, because a previous analysis of this problem revealed that these constraints are active at the optimum [30]. Consequently, we can reformulate them as equality constraints and still obtain the same solution. The thickness constraints are not so straightforward, because not all of them are active, and selecting individual locations where they are active proved cumbersome to implement. Fortunately, our experiments revealed that the regularized problem behaves similarly to the thickness-constrained problem, allowing us to omit the thickness constraints entirely. We will demonstrate this further in Section IV.B.

[c] https://info.aiaa.org/tac/ASG/APATC/AeroDesignOpt-DG/default.aspx

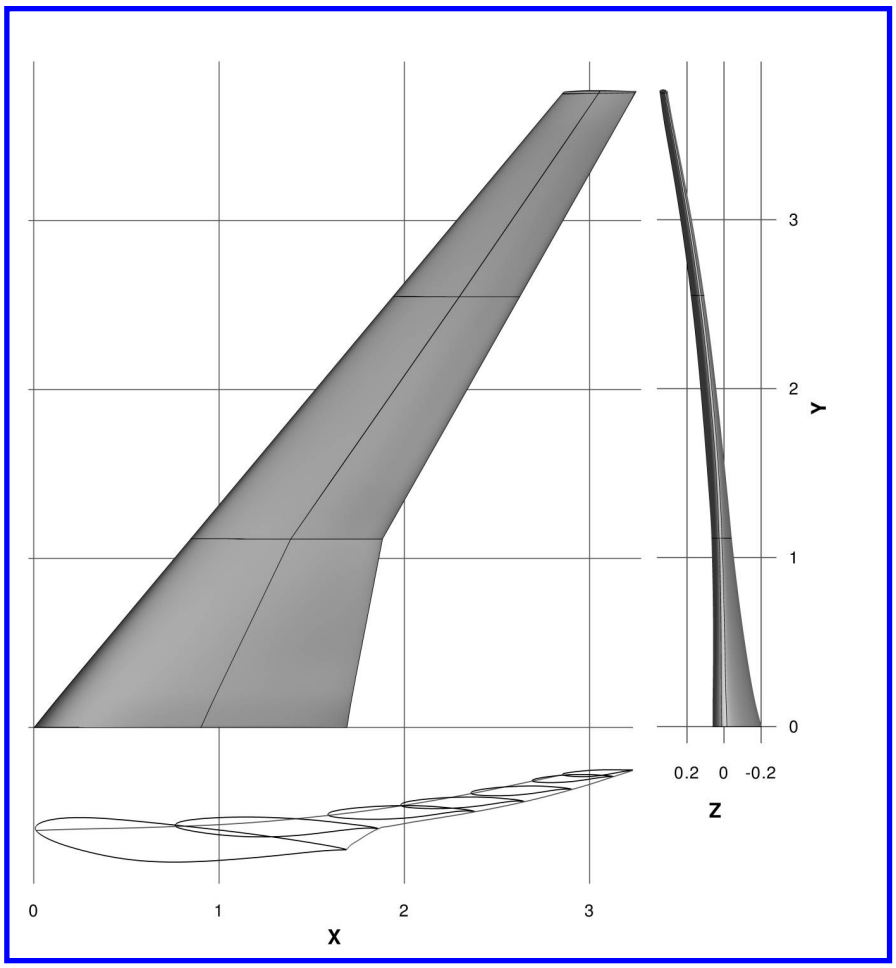

Figure 1. ADODG CRM wing geometry [30].

In summary, the optimization problem considered in this work is

$$\begin{aligned} \min_{x} \quad & f(x, u(x)) = C_D(x, u(x)) + \mu x^T x \\ \text{subject to} \quad & c(x, u(x)) = \begin{bmatrix} C_L(x, u(x)) - 0.5 \\ C_{My}(x, u(x)) + 0.17 \\ V(x) - V_0 \end{bmatrix} = 0, \end{aligned}$$

where the Mach number and angle of attack are fixed to $M = 0.65$ and $\alpha = 2.2°$ respectively. As before, x denotes the vector of design variables and $u(x)$ are the state variables expressed as implicit functions of the design variables. The coefficients of lift and pitching moment, C_L and C_{My}, are defined as equality constraints and fixed to the ADODG benchmark values of 0.5 and -0.17, respectively. The volume of the wing, $V(x)$, is constrained to its initial volume, V_0.

The objective function is the sum of the drag coefficient, C_D, and the regularization term, which is the squared-L2 norm of the design vector multiplied by μ. The parameter μ controls the magnitude of the regularization term, and it must be tuned to ensure that the regularization does not overwhelm the coefficient of drag. The μ parameter was determined through trial and error such that the value of $\mu x^T x$ was about three orders of magnitude smaller than the coefficient of drag at convergence. The introduction of the regularization increases the convexity of the design space, which decreases the condition number of the Hessian W.

The design variables control the wing's cross-sectional shape and relative position along the span. The wing is parameterized using a free-form deformation (FFD) volume approach [34]; Fig. 2 illustrates the FFD volume for 768 control points. The z-coordinates of these control points are used as the design variables in

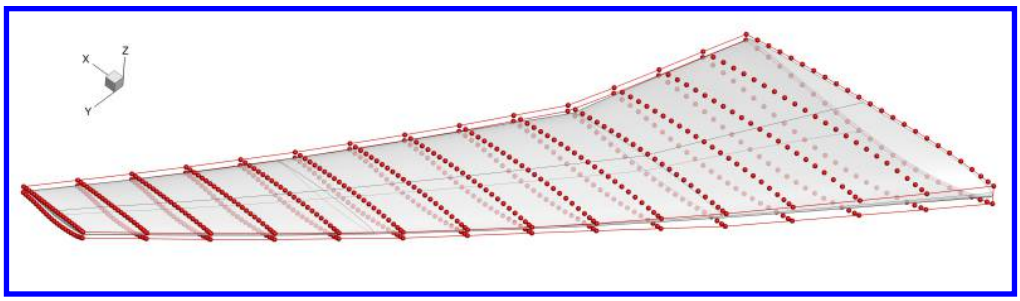

Figure 2. CRM wing and the FFD parameterization using 768 control points [30].

Table 2. Number of FFD box control points in each direction shown for the range of FFD box sizes used in this study.

	FFD Box Sizes					
Directions	**72**	**192**	**320**	**480**	**616**	**768**
Chordwise	6	12	16	20	22	24
Spanwise	6	8	10	12	14	16
Thickness	2	2	2	2	2	2

the following studies. Table 2 shows the directional resolution of the entire range of FFD box sizes used in this study.

IV.A. CFD Solver

The coefficients of lift, drag, and pitching moment are evaluated using SUmb [35], a finite-volume flow solver capable of solving the compressible Euler, laminar Navier–Stokes and RANS equations; for the latter, SUmb supports a variety of turbulence models. SUmb includes a discrete adjoint method that was implemented using AD [36,37], which has been previously used for RANS-based aerodynamic shape optimization [17,18].

This initial adjoint implementation in SUmb used assembled state and design variable Jacobians, computed using forward mode AD with Tapenade [38]. While the stored assembled operators are efficient for solving the direct and adjoint systems, it is cost-prohibitive to use this technique for state and design variable perturbations, since each each would require an expensive matrix reassembly. For Jacobian-vector products, the forward mode AD code was easily repurposed for this task. However, the transpose Jacobian-vector products required the implementation of reverse mode AD for the same routines. This technique enables efficient transpose Jacobian vector products for arbitrary design and state variable vectors in a matrix-free fashion. The efficient computation of these products is critical to the overall cost of the aerodynamic shape optimization using the RSNK algorithm. Future work will focus on implementing the matrix-free products on the differentiated RANS equations [39].

For our investigation, SUmb was configured to solve the Euler equations with a combined multigrid Runge–Kutta (RK) and Newton–Krylov (NK) approach. The coarse grid solution is converged to a tolerance of 10^{-4} using RK, and the fine grid solution is converged to a tolerance of 10^{-10}, where the solver switches to NK at a 10^{-2} PDE residual tolerance.

IV.B. Impact of the Regularization

Before we compare the two optimization algorithms, it is important to illustrate the effect of the regularization on the optimization problem. To this end, Fig. 3 shows the SNOPT convergence histories for the regularized problem, the corresponding non-regularized problem without any thickness constraints, and the problem using the thickness constraints from the the original ADODG benchmark. The non-regularized problem, devoid of thickness constraints, is free to explore impractical geometric features, such as extremely thin trailing edges and sharp leading edges, in order to achieve a lower coefficient of drag. The regularized problem, on the other hand, appears to converge to a similar optimum design as the thickness constrained case, which suggests that it is a good proxy for the ADODG benchmark, given the limitations of our current

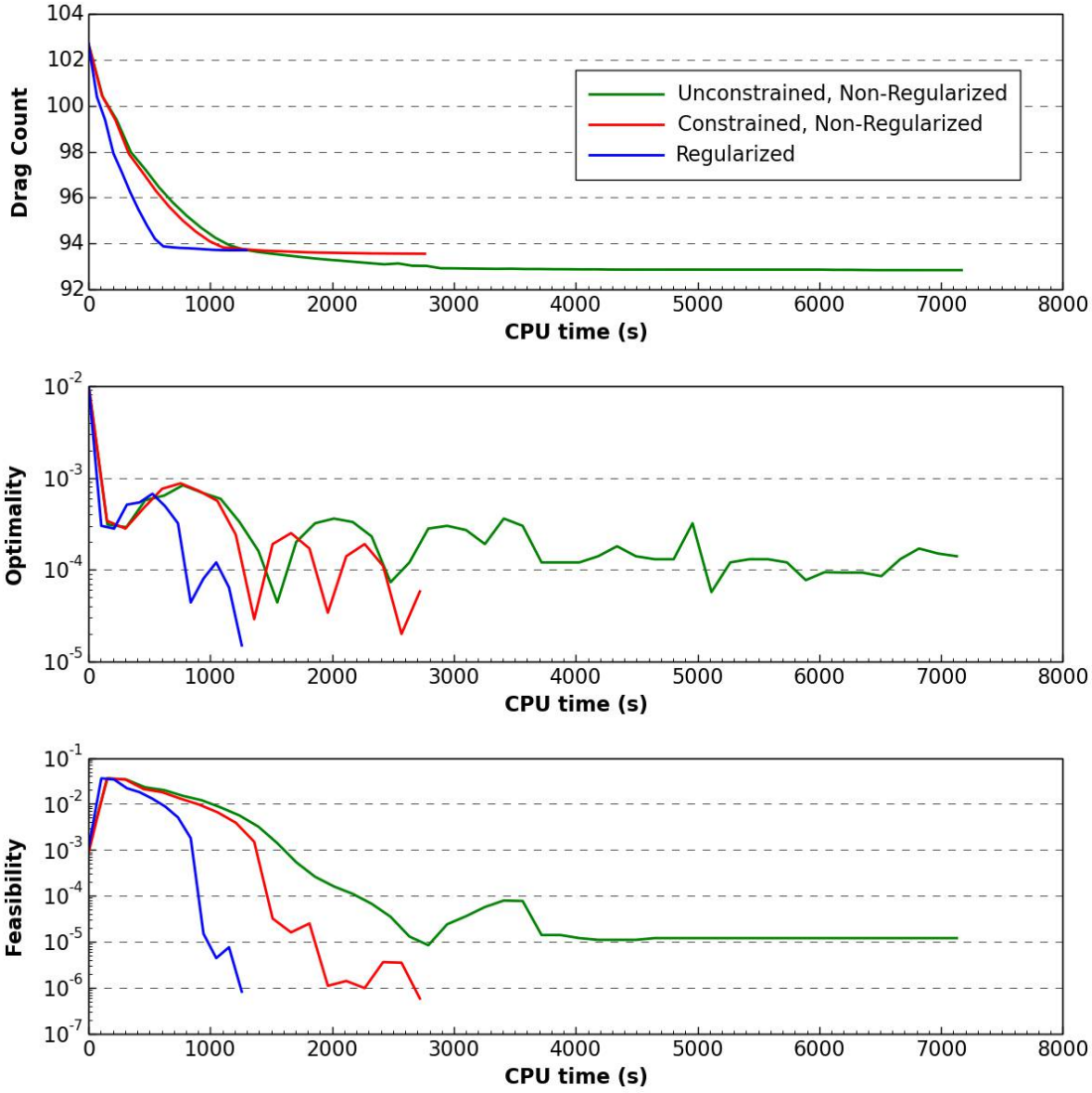

Figure 3. Comparison of SNOPT convergences on three variations of the problem: non-regularized without thickness constraints, non-regularized with 25% thickness constraints, and regularized.

RSNK implementation. Furthermore, the increased convexity provided by the regularization makes the problem easier for SNOPT as well as RSNK, and, therefore, it does not unfairly favor one algorithm over the other.

V. Results

In this section, we present the results of our investigation. All results in this section have been collected by solving an 840,192 cell Euler grid on a 2.5 GHz eight-core Intel Xeon E5-2650 cluster, using 64 cores across 4 compute nodes. Both SNOPT and RSNK were required to converge their optimality metrics by three orders of magnitude from the initial. For this specific problem, converging beyond three orders of magnitude did not yield any meaningful drag reduction.

V.A. Regarding Superlinear Convergence with RSNK

We begin by investigating the convergence behavior of the RSNK algorithm. Because this algorithm is an inexact-Newton approach, it should be possible to obtain a superlinear convergence rate.

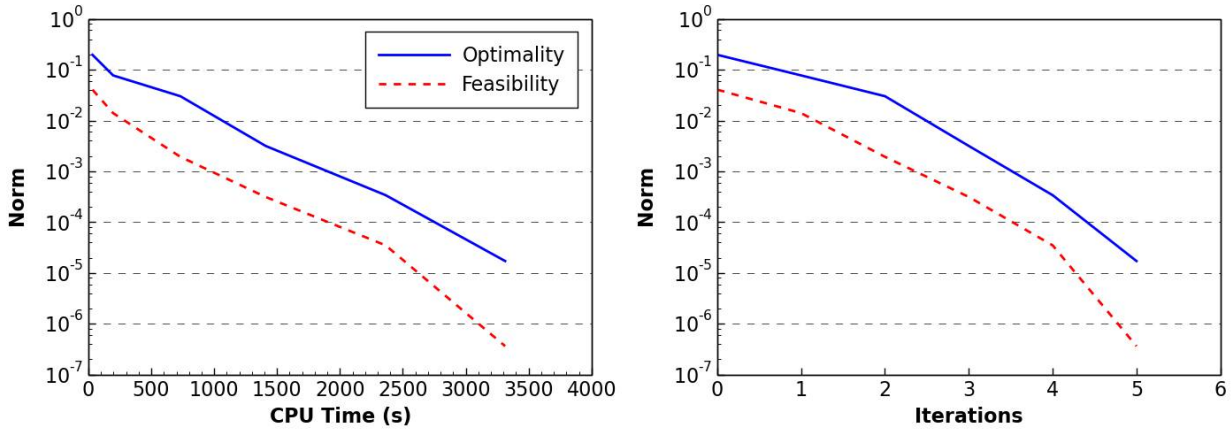

Figure 4. Convergence history of RSNK with 192 FFD coefficients, plotted against CPU time (left) and iterations (right).

Fig. 4 shows the convergence history of RSNK on the regularized problem with 192 design variables. The history has been plotted against two different cost metrics, CPU time in seconds and iterations. Plotted against iterations, RSNK does indeed achieve superlinear convergence of the optimality and feasibility on this problem. Superlinearity disappears when plotted against CPU time, because the convergence tolerance for the KKT system gets progressively smaller as the optimization approaches the solution; therefore, the FLECS solver requires more iterations to converge on this non-preconditioned problem. As a result, the elapsed time per outer iteration increases throughout the optimization process, causing the superlinear convergence to be lost when plotted against CPU time.

In practice, the RSNK algorithm aims to solve the KKT system inexactly, requiring convergence to a designated tolerance only within a small number of FLECS iterations. This inherently imposes a limit to how long each outer iteration can take. However, RSNK's superlinearity is nonetheless dependent on sufficiently reducing the KKT system residual. The superlinearity is lost when the solution to the KKT system becomes too inexact.

The ideal number of Krylov iterations for the FLECS solver is largely dependent on the cost of solving the second-order adjoint systems and the matrix-vector products used to assemble the KKT-matrix-vector product described in Section III. For this particular implementation and problem, our experiments indicate that establishing superlinear convergence requires 30 to 40 Krylov iterations. However, allowing this many iterations leads to a loss of competitiveness with respect to SNOPT in terms of CPU time. In the following sections, we will illustrate promising and competitive results achieved using 15 to 20 iterations, depending on the size of the design space, but it is worth noting that this limitation leads to a loss of superlinear convergence. This provides further evidence that an effective preconditioner for the KKT system is needed in order to achieve both superlinearity and improved computational cost.

V.B. Aerodynamic Shape Optimization of the CRM Wing

The coefficient of pressure distributions over the initial and optimized geometries are shown in Figures 5 and 6, respectively. The figures also include the C_D and constraint values for the geometries. The optimal designs shown here correspond to the 768 design-variable FFD volume, but the solution is representative of the entire range of FFD sizes used in this study. All solutions of this regularized problem with either algorithm converge to optima within 0.5 drag counts of each other.

Figure 7 shows the convergence histories for both algorithms over the full range of design spaces considered. Kona's optimality and feasibility metrics differ from SNOPT's, so these values have been normalized and the convergence histories show the reduction from the respective initial values. A clear difference between the two algorithms is that SNOPT permits significant infeasibility during the early iterations.

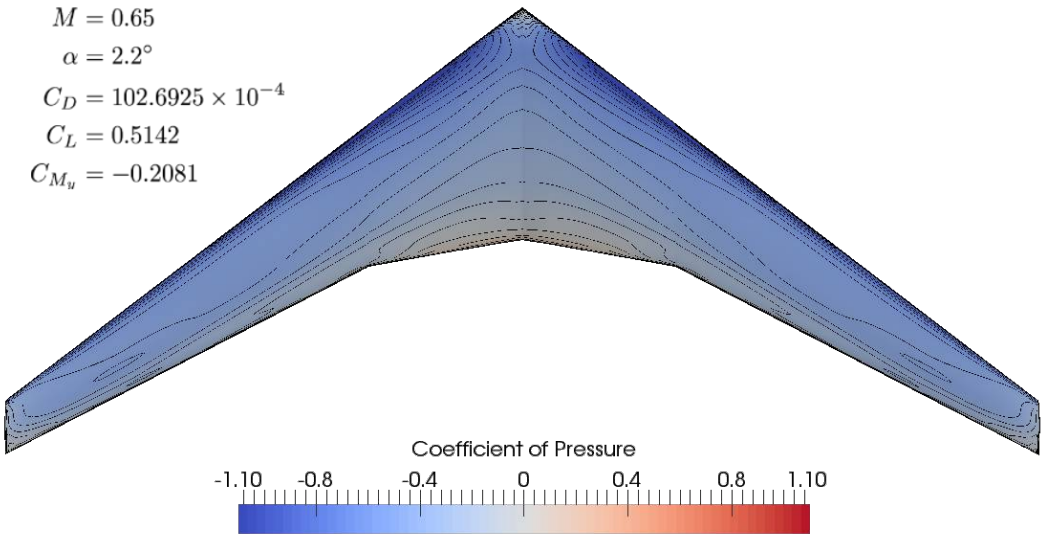

Figure 5. Euler-based C_p distribution on the initial CRM wing.

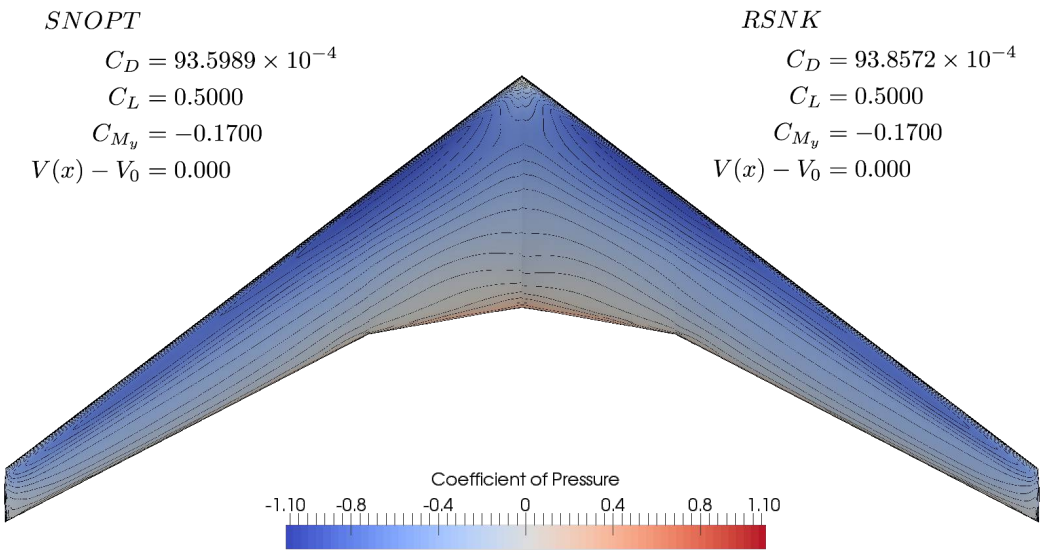

Figure 6. Aerodynamic solution of the optimal design for SNOPT (left half) and RSNK (right half). C_L, C_{My} and the volume constraint are approximately the same for both optimums to 8^{th} significant digit.

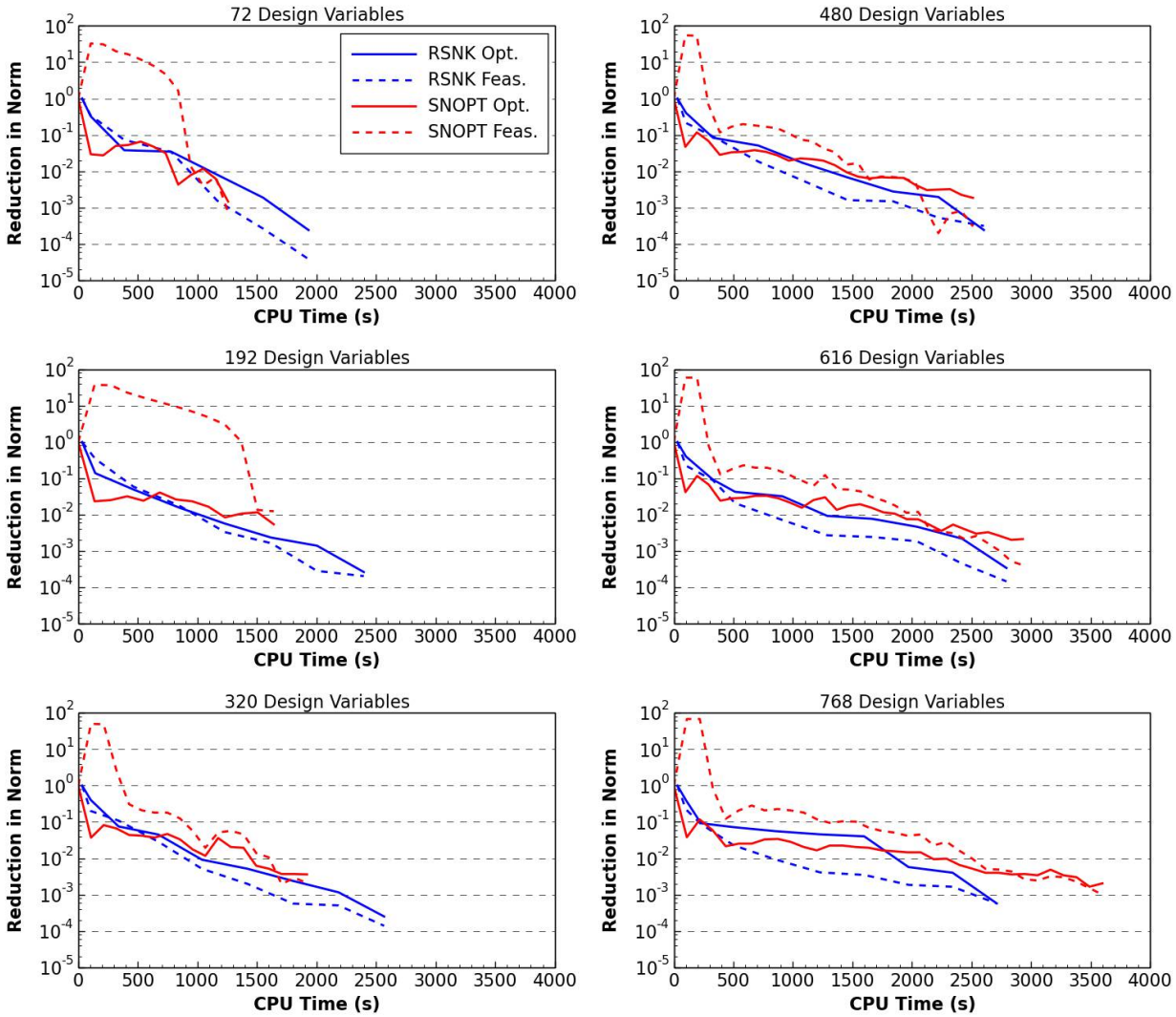

Figure 7. Optimality and feasibility convergence history of RSNK and SNOPT at different design space sizes.

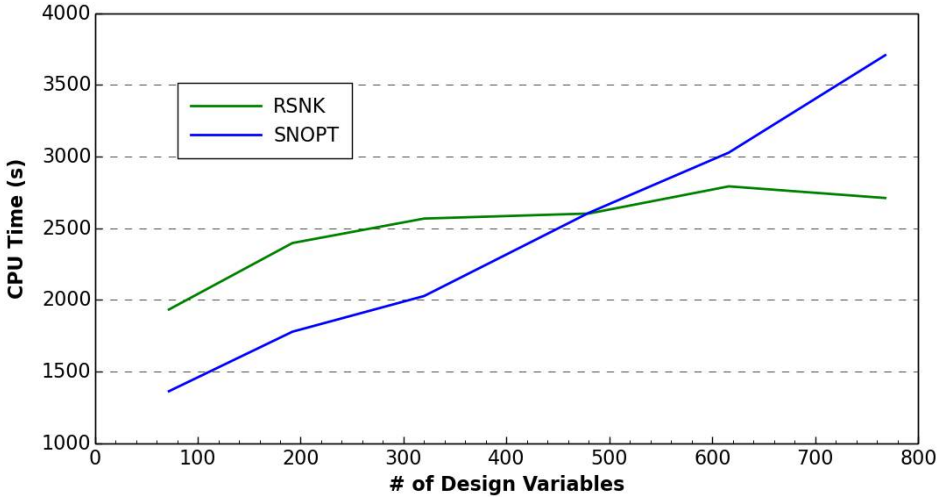

Figure 8. Computational cost scaling with respect to the size of the design space.

V.C. Computational Cost Scaling with Design Space

To determine how the optimization algorithms scale with the size of the design space, the convergence results from Figure 7 are compiled into a direct comparison of cost scaling in Figure 8.

The results demonstrate that RSNK has favorable scaling with respect to the dimensionality of the design space. Despite the loss of superlinearity, RSNK converges faster than SNOPT for 616 and 768 design variables, and remains competitive at 480. Should this trend continue to hold, RSNK promises to be an efficient algorithm for problems with thousands of design variables.

VI. Conclusions

We implemented and investigated a reduced-space Newton–Krylov (RSNK) algorithm in the solution of the aerodynamic shape optimization of the ADODG CRM wing. The aerodynamic shape optimization results were based on a subsonic Euler flow, and computational cost scaling was analyzed across 72, 192, 320, 480, 616 and 768 design variables. In addition, the RSNK algorithm library, Kona, was benchmarked against a Jacobian-explicit quasi-Newton optimization library, SNOPT.

Our results demonstrate that the RSNK algorithm is capable of superlinear convergence on a regularized problem. However, achieving this superlinear convergence requires more iterations of the KKT system solution than anticipated, preventing RSNK from being competitive with SNOPT in CPU time. This strongly suggests the need to precondition the primal-dual solver, such that the KKT system residual can be sufficiently reduced with fewer iterations, achieving superlinearity at a lower computational cost. Nonetheless, our results successfully demonstrate the potential for superlinear convergence on a high-fidelity aerodynamic shape optimization problem.

We have also shown that the RSNK algorithm is relatively insensitive to the size of the design space. Relative to SNOPT, we were able to achieve faster convergence to the optimum with 616 and 768 design variables, and remain competitive with 480 variables. The cost trend indicates that RSNK is well suited for problems with large dimensionality.

Over the course of this study, we developed the operations required by our RSNK optimization library, Kona, in part by using subroutines already available in the SUmb flow solver. While the required operations for RSNK were all available in SUmb, transpose Jacobian vector production operations were only available through stored jacobians computed using forward mode AD. The overhead associated with assembling the entire Jacobian for a single transpose vector product is not cost competitive. This bottleneck was removed by using reverse mode AD to compute transpose matrix vector products in a matrix-free fashion. This matrix-free approach proved crucial for RSNK. Additional work is currently underway for improving the efficiency of these matrix-free product computations.

In closing, RSNK shows promise as an efficient and effective optimization algorithm for high-fidelity PDE-constrained optimization problems involving large design spaces. In the immediate future, we hope to achieve robust superlinear convergence by preconditioning the KKT system, and subsequently replicate our results on a more challenging RANS-based aerodynamic shape optimization problem. In the longer term, we intend to explore the use of the RSNK algorithm for coupled multi-disciplinary systems, such as aerostructural optimization problems, which can scale to thousands of design variables and constraints [20, 40]. Targeting such a problem will help us determine how RSNK performs on problems with large numbers of state-dependent constraints, which we would not be able to explore adequately in this work with only two such constraints.

VII. Acknowledgements

The first author was supported by NSF through grant number 1332819 (program director: Chris Paredis), and the remaining authors were supported by NASA through grant number NNX14AC73A (technical monitor: Justin Gray). Computations were performed on the DRP Cluster at Rensselaer Polytechnic Institute's Center for Computational Innovations. The authors gratefully acknowledge this support.

References

[1]Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[2]Kennedy, G. J. and Hicken, J. E., "Improved constraint-aggregation methods," *Computer Methods in Applied Mechanics and Engineering*, Oct. 2014, (under review).

[3]Poon, N. M. K. and Martins, J. R. R. A., "An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis," *Structural and Multidisciplinary Optimization*, Vol. 34, No. 1, 2007, pp. 61–73.

[4]Liu, D. C. and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, Vol. 45, 1989, pp. 503–528.

[5]Hazra, S. B., "An efficient method for aerodynamic shape optimization," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, No. AIAA–2004–4628, Albany, New York, Aug. 2004.

[6]Hazra, S. B., "Direct treatment of state constraints in aerodynamic shape optimization using simultaneous pseudo-time-stepping," *AIAA Journal*, Vol. 45, No. 8, Aug. 2007, pp. 1988–1997.

[7]Hazra, S. B., "Multigrid one-shot method for state constrained aerodynamic shape optimization," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 3220–3248.

[8]Borzì, A. and Schulz, V., *Computational Optimization of Systems Governed by Partial Differential Equations*, Society for Industrial and Applied Mathematics, Jan. 2011.

[9]Biros, G. and Ghattas, O., "Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: the Krylov-Schur solver," *SIAM Journal on Scientific Computing*, Vol. 27, 2005, pp. 687–713.

[10]Biros, G. and Ghattas, O., "Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: the Lagrange-Newton solver and its application to optimal control of steady viscous flows," *SIAM Journal on Scientific Computing*, Vol. 27, 2005, pp. 714–739.

[11]Heinkenschloss, M. and Vicente, L. N., "An interface optimization and application for the numerical solution of optimal control problems," *ACM Transactions on Mathematical Software*, Vol. 25, No. 2, June 1999.

[12]Akçelik, V., Biros, G., and Ghattas, O., "Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation," *SC Conference*, 2002, pp. 41.

[13]Akçelik, V., Biros, G., Ghattas, O., Hill, J., Keyes, D., and van Bloemen Waanders, B., "Parallel Algorithms for PDE-Constrained Optimization," *Parallel Processing for Scientific Computing*, edited by M. A. Heroux, P. Raghavan, and H. D. Simon, chap. 16, Society for Industrial and Applied Mathematics, Jan. 2006, pp. 291–322.

[14]Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S., *Optimization with PDE constraints*, Springer, 2009.

[15]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: an SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.

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

[17]Lyu, Z., Kenway, G. K., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Studies on the Common Research Model Wing Benchmark," *AIAA Journal*, 2014, (In press).

[18]Lyu, Z. and Martins, J. R. R. A., "Aerodynamic Design Optimization Studies of a Blended-Wing-Body Aircraft," *Journal of Aircraft*, Vol. 51, No. 5, September 2014, pp. 1604–1617.

[19]Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Derivative Computations," *AIAA Journal*, Vol. 52, No. 5, May 2014, pp. 935–951.

[20]Kenway, G. K. W. and Martins, J. R. R. A., "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, January 2014, pp. 144–160.

[21]Kennedy, G. J. and Martins, J. R. R. A., "A parallel aerostructural optimization framework for aircraft design studies," *Structural and Multidisciplinary Optimization*, 2014, (In press).

[22]Hwang, J. T., Lee, D. Y., Cutler, J. W., and Martins, J. R. R. A., "Large-Scale Multidisciplinary Optimization of a Small Satellite's Design and Operation," *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, September 2014, pp. 1648–1663.

[23]Hicken, J. E. and Dener, A., "A Flexible Iterative Solver for Nonconvex, Equality-Constrained Quadratic Subproblems," Tech. rep., Scientific Computation Research Center, Oct. 2014, (under review).

[24]Conn, A. R., Gould, N. I. M., and Toint, P. L., *Trust Region Methods*, Society for Industrial and Applied Mathematics, Jan. 2000.

[25]Fletcher, R. and Leyffer, S., "Nonlinear programming without a penalty function," *Mathematical Programming*, Vol. 91, No. 2, Jan. 2002, pp. 239–269.

[26]Knoll, D. A. and Keyes, D. E., "Jacobian-free Newton-Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, Vol. 193, No. 2, 2004, pp. 357–397.

[27]Wang, Z., Navon, I. M., Dimet, F. X., and Zou, X., "The second order adjoint analysis: Theory and applications," *Meteorology and Atmospheric Physics*, Vol. 50, 1992, pp. 3–20.

[28]Hicken, J. E. and Alonso, J. J., "Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization," *51st AIAA Aerospace Sciences Meeting*, Jan. 2013, AIAA 2013–1043.

[29]Hicken, J., "Inexact Hessian-vector products in reduced-space differential-equation constrained optimization," *Optimization and Engineering*, 2014, pp. 1–34.

[30]Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, 2014, (Accepted).

[31]Vassberg, J. and Jameson, A., "Influence of Shape Parameterization on Aerodynamic Shape Optimization," Tech. rep., Von Karman Institute, Brussels, Belgium, April 2014.

[32]Telidetzki, K., Osusky, L., and Zingg, D. W., "Application of Jetstream to a Suite of Aerodynamic Shape Optimization Problems," *52nd Aerospace Sciences Meeting*, Feb 2014.

[33]Carrier, G., Destarac, D., Dumont, A., Meheut, M., Din, I. S. E., Peter, J., Khelil, S. B., Brezillon, J., and Pestana, M., "Gradient-Based Aerodynamic Optimization with the elsA Software," *52nd Aerospace Sciences Meeting*, Feb 2014.

[34]Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, September 2010, AIAA 2010-9231.

[35]van der Weide, E., Kalitzin, G., Schluter, J., and Alonso, J. J., "Unsteady Turbomachinery Computations Using Massively Parallel Platforms," *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2006, AIAA 2006-0421.

[36]Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and van der Weide, E., "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, April 2008, pp. 863–873.

[37]Lyu, Z., Kenway, G. K., Paige, C., and Martins, J. R. R. A., "Automatic Differentiation Adjoint of the Reynolds-Averaged Navier–Stokes Equations with a Turbulence Model," *21st AIAA Computational Fluid Dynamics Conference*, San Diego, CA, Jul 2013.

[38]Hascoët, L. and Pascual, V., "The Tapenade Automatic Differentiation tool: Principles, Model, and Specification," *ACM Transactions On Mathematical Software*, Vol. 39, No. 3, 2013.

[39]Lyu, Z., Kenway, G. K., Paige, C., and Martins, J., "Automatic Differentiation Adjoint of the Reynolds-Averaged Navier-Stokes Equations with a Turbulence Model," *21st AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, June 2013.

[40]Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Aerostructural optimization of the Common Research Model configuration," *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, GA, June 2014, AIAA 2014-3274.