51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
07 - 10 January 2013, Grapevine (Dallas/Ft. Worth Region), Texas

AIAA 2013-1043

Comparison of reduced- and full-space algorithms for PDE-constrained optimization

Jason E. Hicken*
Rensselaer Polytechnic Institute, Troy, New York, 12180

Juan J. Alonso†
Stanford University, Stanford, California 94305-3030

PDE-constrained optimization problems are often solved using reduced-space quasi-Newton algorithms. Quasi-Newton methods are effective for problems with relatively few degrees of freedom, but their performance degrades as the problem size grows. In this paper, we compare two inexact-Newton algorithms that avoid the algorithmic scaling issues of quasi-Newton methods. The two inexact-Newton algorithms are distinguished by reduced-space and full-space implementations. Numerical experiments demonstrate that the full-space (or one-shot) inexact-Newton algorithm is typically the most efficient approach; however, the reduced-space algorithm is an attractive compromise, because it requires less intrusion into existing solvers than the full-space approach while retaining excellent algorithmic scaling. We also highlight the importance of using inexact-Hessian-vector products in the reduced-space.

I. Introduction

Engineers often rely on large-scale multi-physics simulations to aid in the design of complex engineering systems. For example, simulations based on partial-differential equations (PDEs) are used to complement experiments and conduct parameter studies of candidate designs. Increasingly, the design is guided by PDE-constrained optimization; numerical optimization of the design based directly on high-fidelity simulations.

The adoption of large-scale simulations in industry has driven, and has been driven by, significant improvements in computing power and algorithms. For example, research into parallelization, multi-grid, and Newton-Krylov methods has permitted ever larger problems to be considered.

While considerable effort has been focused on the scalability of PDE-based simulations, much less attention has been given to the scalability of optimization algorithms. Even among efficient gradient-based algorithms, reduced-space quasi-Newton methods remain ubiquitous in the engineering community, despite the fact that these methods often scale linearly with the number of design variables. This scaling will render quasi-Newton methods (e.g. BFGS) impractical as we move toward higher dimensional design spaces.

Full-space methods offer one avenue to good algorithmic scaling during an optimization.[1–11] Also known as all-at-once, one-shot, or simultaneous analysis and design (SAND), these methods solve the equations governing the simulation(s), adjoint problem(s), and optimization as a single coupled system.

The aerospace community has investigated the full-space approach for at least two decades, but practitioners remain reluctant to adopt the method. There are several possible explanations for this reluctance.

- Few (if any) general purpose optimization codes exist that can accommodate large-scale parallel simulations, so developers must extend their PDE solver(s) to handle optimization.
- Many physics-based simulations use highly nonlinear models, such as turbulence closures and combustion models, that pose significant challenges for general-purpose optimization algorithms that cannot make use of problem-dependent globalization strategies.

*Assistant Professor, Department of Mechanical, Aerospace, and Nuclear Engineering, Member AIAA
†Associate Professor, Department of Aeronautics and Astronautics, Senior Member AIAA

- In some cases, e.g. unsteady simulations, the memory requirements of the full-space approach are prohibitive.

In this paper we present a reduced-space inexact-Newton-Krylov (INK) method that offers a potential compromise between full-space optimization methods and reduced-space quasi-Newton methods. By using a Newton-based solution strategy, the reduced-space INK method retains the excellent algorithmic scaling of full-space methods like LNKS.[6,7] On the other hand, by remaining in the reduced space, the proposed INK method can better leverage existing software.

Previous work suggests that the Hessian-vector products used in reduced-space INK methods must be computed with high precision to maintain orthogonality (or conjugacy) between the Krylov subspace vectors. We show that this accuracy requirement can be relaxed, by combining approximate (or inexact) Hessian-vector products with an optimal Krylov subspace method like Flexible Generalized Minimal RESidual (FGMRES).[12] These inexact Hessian-vector products are essential to the efficient performance of INK methods applied in the reduced space.

We have organized the paper as follows. Section II provides a review of the general PDE-constrained optimization problem. Subsequently, we describe the three algorithms that will be compared. The reduced-space algorithms, quasi-Newton and INK, are described in Section III. The full-space algorithm is summarized in Section IV. Results comparing the three optimization methods are presented in Section V, and concluding remarks can be found in Section VI.

II. PDE-constrained optimization

We begin by introducing notation. We will use $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{u} \in \mathbb{R}^n$ to denote the finite-dimensional control and state variables, respectively. In the context of PDE-constrained optimization, the state variables arise from the chosen discretization of the PDE; $\mathbf{u}$ may represent function values at nodes in a mesh or coefficients in a basis expansion. The control variables can be given a similar interpretation.

Let $\mathcal{R}(\mathbf{x},\mathbf{u}) = \mathbf{0}$ represent the discretized PDE, or PDEs, together with appropriate boundary conditions and initial conditions, if the latter are necessary. Furthermore, let the functional $\mathcal{J} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ denote the objective, e.g. drag or weight, that we are interested in minimizing. We consider the following discretized PDE-constrained optimization problem.

$$\begin{aligned} &\text{minimize} && \mathcal{J}(\mathbf{x},\mathbf{u}), && \mathbf{x} \in \mathbb{R}^m,\ \mathbf{u} \in \mathbb{R}^n, \\ &\text{subject to} && \mathcal{R}(\mathbf{x},\mathbf{u}) = \mathbf{0}. \end{aligned} \tag{1}$$

Next, we introduce the Lagrangian $\mathcal{L} : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ of (1), which is a discrete functional defined by

$$\mathcal{L}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}) \equiv \mathcal{J}(\mathbf{x},\mathbf{u}) + \boldsymbol{\psi}^T P \mathcal{R}(\mathbf{x},\mathbf{u}). \tag{2}$$

Here, $P \in R^{n\times n}$ is a symmetric positive-definite matrix that defines a discrete inner product appropriate to the chosen discretization. To clarify this inner product, suppose $\mathcal{R}$ contains only the discretized PDE (and no boundary/initial conditions). Then P is the $n \times n$ identity matrix for Galerkin finite element methods, since the weak form already incorporates a quadrature of the integral inner product. For summation-by-parts (SBP) finite-difference schemes P will be the weight matrix associated with the derivative operator.

If we assume that both $\mathcal{J}$ and $\mathcal{R}$ are continuously differentiable, then a local minimum of (1) will be a stationary point of the Lagrangian that satisfies the following first-order optimality conditions:[13]

$$\mathcal{L}_{\boldsymbol{\psi}} = 0 \quad \Rightarrow \quad P\mathcal{R}(\mathbf{x},\mathbf{u}) = 0, \tag{3a}$$

$$\mathcal{L}_{\mathbf{u}} = 0 \quad \Rightarrow \quad \mathcal{J}_{\mathbf{u}}(\mathbf{x},\mathbf{u}) + \boldsymbol{\psi}^T P \mathcal{R}_{\mathbf{u}}(\mathbf{x},\mathbf{u}) = 0, \tag{3b}$$

$$\mathcal{L}_{\mathbf{x}} = 0 \quad \Rightarrow \quad \mathcal{J}_{\mathbf{x}}(\mathbf{x},\mathbf{u}) + \boldsymbol{\psi}^T P \mathcal{R}_{\mathbf{x}}(\mathbf{x},\mathbf{u}) = 0. \tag{3c}$$

Subscripted variables indicate differentiation with respect to that variable, e.g. $\mathcal{J}_{\mathbf{u}} \equiv \partial\mathcal{J}/\partial\mathbf{u}$.

For finite-dimensional optimization problems, the first-order optimality conditions (3) are also called the Karush-Kuhn-Tucker, or KKT, conditions. The first condition, equation (3a), simply states that an optimal solution must satisfy the discretized PDE[a]. Equation (3b) is called the adjoint equation, and the Lagrange multipliers that solve this equation are called the adjoint or costate variables. They are traditionally denoted

[a]Since P is assumed to be symmetric-positive-definite, P can be eliminated from (3a) to recover the PDE constraint in (1).

by $\boldsymbol{\psi} \in \mathbb{R}^n$. Finally, condition (3c) indicates that the gradient of the objective function must vanish in the linearized feasible space.

Most PDE-constrained optimization algorithms seek solutions to (1) by solving, either directly or indirectly, the set (3). Algorithms for solving (3) can be classified into two broad families: full space and reduced space. Below, we review these two formulations and describe the specific reduced-space and full-space methods that we will compare.

III. Reduced-space Methods

III.A. Overview

Within the aerospace community, reduced-space methods have been the most popular approach to solving PDE-constrained optimization problems. These methods assume that the state and adjoint variables are implicit functions of the control variables. This assumption is reasonable, because the PDE should be well-posed with a unique solution $\mathbf{u}$ for a given (valid) $\mathbf{x}$; similar arguments hold for the adjoint equation and $\boldsymbol{\psi}$.

Using the implicit function theorem, the reduced-space approach recasts the optimization problem as

$$\text{minimize} \qquad \mathcal{J}\left(\mathbf{x}, \mathbf{u}(\mathbf{x})\right), \qquad \mathbf{x} \in \mathbb{R}^m. \tag{4}$$

One can show (see, for example, Ref. 14) that the first-order optimality conditions for (4) reduce to

$$\mathbf{g}(\mathbf{x}) \equiv \mathcal{J}_{\mathbf{x}}\left(\mathbf{x}, \mathbf{u}(\mathbf{x})\right) + \left(\boldsymbol{\psi}(\mathbf{x})\right)^T P \mathcal{R}_{\mathbf{x}}\left(\mathbf{x}, \mathbf{u}(\mathbf{x})\right) = 0, \tag{5}$$

where $\mathbf{g}(\mathbf{x})$ denotes the reduced gradient. As mentioned above, $\mathbf{u}(\mathbf{x})$ is an implicit function of $\mathbf{x}$ via the solution to the primal discretized PDE (3a). The adjoint variable $\boldsymbol{\psi}(\mathbf{x})$ is an implicit function of $\mathbf{x}$ via the solution to the primal PDE and adjoint equation (3b).

Since reduced-space methods satisfy equations (3a) and (3b) at each iteration, only the gradient condition (5) must be handled by the optimization algorithm. One approach to solving $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ is to apply Newton's method, i.e. solve

$$H_k \Delta \mathbf{x}_k = -\mathbf{g}(\mathbf{x}_k), \tag{6}$$

where $\Delta \mathbf{x}_k \equiv \mathbf{x}_{k+1} - \mathbf{x}_k$ and $H_k \equiv \mathbf{g}_{\mathbf{x}}(\mathbf{x}_k)$ is the Hessian of the objective function in the reduced space.

It is not immediately obvious how we might efficiently compute the reduced Hessian H_k, or, more to the point, how we might solve (6). Quasi-Newton methods avoid computing the exact Hessian by constructing an approximate (or approximate inverse) Hessian. We describe a particular quasi-Newton method in next section that will serve as our benchmark for the other algorithms considered.

III.B. Reduced-space quasi-Newton Algorithm

Quasi-Newton algorithms replace the exact Hessian, or its inverse, in (6) with an approximation. The Broyden-Fletcher-Goldberg-Shanno (BFGS) update is considered the most effective quasi-Newton scheme,[13] so we use BFGS for our benchmark optimization algorithm. In particular, we use a limited-memory variant of BFGS[15] (L-BFGS) that is suitable for large-scale optimization problems.

Quasi-Newton schemes, like Newton's method itself, require globalization to ensure convergence for initial iterates that are far from the solution. We adopt two strategies to globalize the L-BFGS method. First, we apply a line search that satisfies the strong-Wolfe conditions; the line search is described by Fletcher[16] and uses quadratic interpolation in the "zoom" phase. The second globalization is a homotopy using parameter continuation.[17] This globalization strategy is not as common as line searches and deserves some additional discussion.

The homotopy method adds a parameterized term to the original objective functional with the aim of making a modified problem that is easier to solve. Once this initial modified problem is solved, the parameterized term is updated producing a new modified problem that is solved using the solution from the first modified problem as the initial iterate. The process then repeats. Consequently, the method leads to a sequence of modified optimization problems with the solution of one becoming the initial iterate for the next. The parameterized term becomes smaller with each iteration until the original problem and solution are recovered.

The parameter continuation adopted here replaces the original optimization problem with the following.

$$\text{minimize} \qquad (1-\nu)\mathcal{J}(\mathbf{x}) + \frac{\nu\beta}{2}(\mathbf{x}-\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0), \qquad \mathbf{x}\in\mathbb{R}^m, \tag{7}$$

where $\nu \in [0,1]$ is the homotopy parameter and $\beta > 0$ is a fixed scaling. We have dropped the state-variable dependency for simplicity. The modified objective functional is simply a scaling of the original objective plus a quadratic. For $\nu = 1$ the solution is clearly $\mathbf{x}_0$, the initial design or control. Intuitively, if $\nu \approx 1$ we expect $\mathbf{x}_0$ to be close to the solution of the modified problem and, therefore, provide a good choice as the initial iterate for Newton's method or a quasi-Newton method. Once this modified problem is solved, ν is decreased and the next modified problem is solved.

Ideally, the scaling β is chosen such that the Hessian of $\frac{\beta}{2}(\mathbf{x}-\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0)$ has entries with similar magnitude to the Hessian of the objective, $\mathcal{J}$. Since the Hessian is difficult or expensive to estimate, in the present work we choose $\beta = \|\mathbf{g}(\mathbf{x}_0)\|_2$ to ensure that the gradients of the two terms are similar in magnitude at the initial design.

One of the advantages of the homotopy (7) over other parameter-continuation strategies is that the BFGS approximation can be reused when the parameter ν changes. To see this, we recall the L-BFGS implementation listed in Algorithm 1, which is based on a two-loop recursion.[13] In the algorithm, $\hat{H}_k$ denotes the quasi-Newton approximation to the Hessian and $\hat{H}_k^0$ is the initial approximate Hessian (taken to be βI here). In the classical BFGS algorithm, $\mathbf{s}_k \equiv \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{z}_k = \mathbf{y}_k \equiv \mathbf{g}_{k+1} - \mathbf{g}_k$. When using the parameter continuation described above, $\mathbf{z}_k$ must be changed to be the difference between successive gradients of the *modified objective*. At step k, the gradient of the modified objective is $(1-\nu)\mathbf{g}_k + \nu\beta(\mathbf{x}_k - \mathbf{x}_0)$; consequently, for a given ν Algorithm 1 uses

$$\begin{aligned}\mathbf{z}_k &\equiv (1-\nu)(\mathbf{g}_{k+1} - \mathbf{g}_k) + \nu\beta(\mathbf{x}_{k+1} - \mathbf{x}_k)\\ &= (1-\nu)\mathbf{y}_k + \nu\beta\mathbf{s}_k.\end{aligned}$$

In practice we store $\mathbf{s}_i$ and $\mathbf{y}_i$ for $i = k-1, k-2, \ldots, k-m$, as in the classical L-BFGS implementation, and recompute $\mathbf{z}_k$ on-the-fly. This allows us to reuse gradient information from previous homotopy iterations that used different ν parameter values.

As is commonly done, the BFGS update is skipped if the curvature condition is violated; that is, if $\mathbf{s}_k^T\mathbf{y}_k < 0$. Notice that the curvature condition is computed using $\mathbf{y}_k$ rather than $\mathbf{z}_k$, because as $\nu \to 0$ it is $\mathbf{y}_k$ that we ultimately need to produce a positive-definite Hessian approximation. An alternative strategy, not pursued here, would be to keep $\mathbf{y}_k$ until $\mathbf{s}_k^T\mathbf{z}_k < 0$, and then discard this update. This could potentially accelerate convergence in the early homotopy iterations.

Algorithm 1: Limited-memory BFGS with two-loop recursion

Data: $\mathbf{u}$
Result: $\mathbf{v} = \hat{H}_k^{-1}\mathbf{u}$

$\mathbf{v} = \mathbf{u}$
for $i = k-1, k-2, \ldots, k-m$ **do**
  $\rho_i \leftarrow \frac{1}{\mathbf{z}_k^T\mathbf{s}_k}$
  $\alpha_i \leftarrow \rho_i\mathbf{s}_i^T\mathbf{v}$
  $\mathbf{v} \leftarrow \mathbf{v} - \alpha_i\mathbf{z}_i$
end
$\mathbf{v} \leftarrow \left(\hat{H}_k^0\right)^{-1}\mathbf{v}$
for $i = k-m, k-m+1, \ldots, k-1$ **do**
  $\gamma \leftarrow \rho_i\mathbf{z}_i^T\mathbf{v}$
  $\mathbf{v} \leftarrow \mathbf{v} + (\alpha_i - \gamma)\mathbf{s}_i$
end

III.C. Reduced-space inexact-Newton-Krylov Algorithm

As mentioned in the introduction, quasi-Newton methods can exhibit poor algorithmic scaling, and this motivates methods that solve (6) more accurately. Let us reexamine this exact Newton update:

$$H_k \Delta \mathbf{x}_k = -\mathbf{g}(\mathbf{x}_k). \tag{6}$$

If these update equations can be solved with sufficient accuracy, we may need significantly fewer iterations than a quasi-Newton method. Ideally, we want a method that scales independently of the number of design variables.

We can solve the linear system (6) with either a direct or iterative method. A direct method would require that we form, store, and factor the reduced Hessian at each iteration. Forming H_k explicitly requires the solution of m linear systems of size $n \times n$; therefore, we expect direct-solution methods to have algorithmic complexities similar to BFGS when applied to a quadratic objective.

Thus, we are left to consider iterative methods for (6). In particular, Krylov iterative methods are attractive, because they require products of H_k with arbitrary vectors $\mathbf{w} \in \mathbb{R}^m$ rather than the reduced Hessian itself. Below, we will show that such Hessian-vector products can be computed efficiently.

While Krylov subspace methods avoid the need to form H_k, they may require a significant number of iterations to converge. For example, assuming H_k is a symmetric-positive-definite matrix, the conjugate gradient (CG) method requires m iterations to converge in exact arithmetic. This would again lead to an algorithmic complexity that is similar to BFGS.

Fortunately, we do not need to solve (6) exactly. In the early Newton iterations, the linear model is typically not an accurate representation of the gradient $\mathbf{g}(\mathbf{x})$. Therefore, it is unnecessary to solve for the root of the linear model with high precision. This is the observation exploited by inexact-Newton methods,[18] which are sometimes called truncated-Newton in the optimization literature.[19] Rather than solve (6) exactly, an inexact-Newton reduced-space algorithm accepts approximate steps, $p_k \in \mathbb{R}^m$, that satisfy the following inequality.

$$\|H_k \mathbf{p}_k + \mathbf{g}(\mathbf{x}_k)\| \leq \eta_k \|\mathbf{g}(\mathbf{x}_k)\|, \tag{8}$$

where $\eta_k \in [0, 1)$ is the forcing parameter that controls the degree to which p_k satisfies the Newton linearization. The subscript on η_k indicates that the forcing parameter may change from one Newton-iteration to the next[b]. Most Krylov subspace solvers can easily accommodate the inexact-Newton condition (8).

The number of iterations needed by the Krylov solver can also be reduced through preconditioning; indeed, Krylov subspace methods must be preconditioned to be effective on most practical problems. For the present work, we maintain an L-BFGS quasi-Newton approximation to the Hessian and use this to precondition the Krylov-subspace method.[6,20]

To find a step p_k that satisfies (8), we employ the Flexible Generalized Minimal RESidual (FGMRES) method.[12] The reduced Hessian is symmetric, which suggests using a Krylov iterative method with a short-term recurrence, e.g. MINRES;[21] however, there are a couple important advantages to using FGMRES.

1. FGMRES permits the preconditioner to change from one Krylov iteration to the next; therefore, we can dynamically update the L-BFGS approximation using the Hessian-vector product produced during each Krylov iteration. In contrast, non-flexible Krylov methods can include the quasi-Newton updates only at the beginning of the next Newton iteration.[20]

2. More importantly, FGMRES explicitly orthogonalizes its basis vectors, which we have found is necessary when the Hessian-vector products are computed approximately; see Section III.C.1.

A disadvantage of using FGMRES is that 2 vectors of length m must be stored for each of its iterations. For the boundary-control problems typically encountered in aerodynamic shape optimization we have $m \ll n$ and the additional storage is insignificant. For distributed control problems the additional storage may become an issue.

Algorithm 2 describes a generic reduced-space optimization using the homotopy-based globalization method. The only significant difference between the quasi-Newton version and the inexact-Newton-Krylov (INK) version is the computation of the step $\mathbf{p}_k$. At the end of each globalization loop, we use a simple update for the continuation parameter; see line 18. The way that ν is reduced can impact the efficiency and robustness of the algorithm, and we do not claim that this particular update formula is appropriate for all cases.

[b] In fact, η_k must tend to zero sufficiently fast to recover quadratic convergence.

Algorithm 2: Generic reduced-space algorithm with homotopy-based globalization

Data: $\mathbf{x}_0$
Result: $\mathbf{x}$ satisfying $\mathbf{g}(\mathbf{x}) = \mathbf{0}$

$\mathbf{x} \leftarrow \mathbf{x}_0$
for $j = 0, 1, 2, \ldots, N$ **do** (globalization loop)
  for $k = 0, 1, 2, \ldots, N_{global}$ **do** (Newton loop)
    solve $\mathcal{R}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ for $\mathbf{u}$ (solve the primal problem)
    solve $\mathcal{S}(\mathbf{x}, \mathbf{u}, \boldsymbol{\psi}) = \mathbf{0}$ for $\boldsymbol{\psi}$ (solve the adjoint problem)
    compute reduced gradient $\mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\psi})$
    compute modified gradient: $\mathbf{g}_\nu = (1 - \nu)\mathbf{g} + \nu\beta(\mathbf{x} - \mathbf{x}_0)$
    if $\|\mathbf{g}_\nu\|_2 \leq \tau_{global}$ **then** (check convergence of modified problem)
      exit Newton loop
    end
    solve $\hat{H}_k \mathbf{p}_k = -\mathbf{g}_\nu$ (quasi-Newton) or inexactly solve $H_k \mathbf{p}_k = -\mathbf{g}_\nu$ (INK)
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{p}_k$ (perform line search if using quasi-Newton)
    store $\mathbf{s}_i$ and $\mathbf{y}_i$ (quasi-Newton update)
  end
  if $\|\mathbf{g}\|_2 \leq \tau$ **then** (check for convergence)
    exit globalization loop
  end
  $\nu \leftarrow \max(\nu - 0.1, 0)$
end

III.C.1. Improving efficiency: inexact Hessian-vector products

It is well known that reduced-space Hessian-vector products can be computed by solving two second-order adjoint systems of size $n \times n$; see, for example, Appendix A and Refs. 6, 8, 22. What is less appreciated, is that these second-order adjoint systems do not need to be solved accurately, which significantly improves the efficiency of the reduced-space approach. Moreover, the Hessian-vector products can be determined without the need to compute second-order derivatives.

To keep the presentation compact, the derivations of the reduced-space Hessian-vector product and its corresponding adjoint equations are relegated to Appendix A. Here, we list only the final result of those derivations. Let $\mathbf{w} \in \mathbb{R}^m$ be an arbitrary vector in the design space. To compute the Hessian-vector product $H\mathbf{w} \equiv \mathbf{z}$, the following adjoint systems are solved, in order, for the variables $\mathbf{v}, \boldsymbol{\lambda} \in \mathbb{R}^n$:

$$P\mathcal{R}_{\mathbf{u}}\mathbf{v} = -\mathbf{g}_{\boldsymbol{\psi}}^T \mathbf{w}, \tag{9}$$

$$\mathcal{R}_{\mathbf{u}}^T P\boldsymbol{\lambda} = -\mathbf{g}_{\mathbf{u}}^T \mathbf{w} - \mathcal{S}_{\mathbf{u}}^T \mathbf{v}, \tag{10}$$

where $\mathbf{g} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\psi})$ is the reduced gradient of $\mathcal{J}$ (see (5)), and $\mathcal{S}$ is the $\boldsymbol{\psi}$-adjoint residual, which is simply the transpose of (3b):

$$\mathcal{S}(\mathbf{x}, \mathbf{u}, \boldsymbol{\psi}) \equiv \mathcal{R}_{\mathbf{u}}^T P\boldsymbol{\psi} + \mathcal{J}_{\mathbf{u}}^T. \tag{11}$$

The assembly and solution of the second-order adjoint equations (9) and (10) deserve some remarks.

- The system matrix of (9) is the Jacobian[c] of the primal equations, and the system matrix of (10) is the transposed Jacobian. Most adjoint-based optimization algorithms are capable of solving for systems involving $\mathcal{R}_{\mathbf{u}}^T$. Solving linear systems involving $\mathcal{R}_{\mathbf{u}}$ is less common in this context, but the infrastructure can easily be adapted.

- The right-hand side of the first adjoint system simplifies:

$$-\mathbf{g}_{\boldsymbol{\psi}}^T \mathbf{w} = -\mathcal{R}_{\mathbf{x}}^T P\mathbf{w}.$$

  This term can be computed in the same way as the second term in the reduced gradient (5).

[c]The matrix P is typically diagonal and amounts to a row scaling of the Jacobian

- The right-hand side vector of the adjoint system for $\boldsymbol{\lambda}$ involves second derivatives; however, the potential complications of computing second derivatives can be avoided by using a finite-difference approximation. Again, see Appendix A for the details.

Once $\mathbf{v}$ and $\boldsymbol{\lambda}$ have been computed, the Hessian-vector product can be evaluated using the following expression:

$$H\mathbf{w} = \mathbf{g}_{\mathbf{x}}^T\mathbf{w} + \boldsymbol{\lambda}^T P\mathcal{R}_{\mathbf{x}}(\mathbf{x},\mathbf{u}) + \mathbf{v}^T P\mathcal{S}_{\mathbf{x}}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}). \tag{12}$$

As above, any second derivatives that appear can be approximated by using a finite-difference approximation.

The accuracy of the second-order adjoints $\mathbf{v}$ and $\boldsymbol{\lambda}$ determines the accuracy of the Hessian-vector product and, in large part, the computational expense of the entire INK algorithm. Thus, the tolerance we use to solve the equations (9) and (10) plays an important role in the efficiency of the algorithm and deserves further discussion.

If we use a short-term recurrence Krylov iterative method (e.g. CG, MINRES), then the Hessian-vector products must be sufficiently accurate to maintain conjugacy/orthogonality between the Krylov subspace vectors. If conjugacy or orthogonality are lost due to inaccurate second-order adjoints, the linear system may fail to converge.[23] This accuracy requirement imposed on the second-order adjoints leads to expensive reduced-space INK algorithms and is a frequently cited disadvantage of these methods.[6,8]

While the Hessian-vector product must be accurate for short-term recurrence methods, the products can be computed inexactly if the Krylov iterative method maintains orthogonality of the solution subspace explicitly. This claim follows from theory of inexact-Krylov-subspace methods developed by Simoncini and Szyld.[24] Indeed, their results indicate that, for methods like GMRES, the Hessian-vector products can become *less accurate* as the Krylov iterative method proceeds. As we shall see, the use of inexact-Hessian products is critical to the efficiency of reduced-space INK methods.

IV. Full-space method

IV.A. Overview

In full-space methods, the nonlinear coupled system of equations (3) are solved simultaneously for the variables $(\mathbf{x}^T\mathbf{u}^T\boldsymbol{\psi}^T)$. For this reason, full-space methods are often referred to as one-shot, all-at-once, or simultaneous-analysis-and-design methods.

Most full-space algorithms are based on some form of Newton's method applied to (3). That is, at each iteration k of the algorithm, the following linearized system is solved, or solved approximately, for the updates $\Delta\boldsymbol{\psi}_{k+1} \equiv \boldsymbol{\psi}_{k+1} - \boldsymbol{\psi}_k$, $\Delta\mathbf{u}_{k+1} \equiv \mathbf{u}_{k+1} - \mathbf{u}_k$, and $\Delta\mathbf{x}_{k+1} \equiv \mathbf{x}_{k+1} - \mathbf{x}_k$.

$$\begin{bmatrix} \mathcal{L}_{\boldsymbol{\psi},\boldsymbol{\psi}} & \mathcal{L}_{\boldsymbol{\psi},\mathbf{u}} & \mathcal{L}_{\boldsymbol{\psi},\mathbf{x}} \\ \mathcal{L}_{\mathbf{u},\boldsymbol{\psi}} & \mathcal{L}_{\mathbf{u},\mathbf{u}} & \mathcal{L}_{\mathbf{u},\mathbf{x}} \\ \mathcal{L}_{\mathbf{x},\boldsymbol{\psi}} & \mathcal{L}_{\mathbf{x},\mathbf{u}} & \mathcal{L}_{\mathbf{x},\mathbf{x}} \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{\psi}_{k+1} \\ \Delta\mathbf{u}_{k+1} \\ \Delta\mathbf{x}_{k+1} \end{pmatrix} = - \begin{bmatrix} \mathcal{L}_{\boldsymbol{\psi}} \\ \mathcal{L}_{\mathbf{u}} \\ \mathcal{L}_{\mathbf{x}} \end{bmatrix}. \tag{13}$$

Newton's method remains attractive here, because, under suitable conditions, it exhibits quadratic convergence. However, as usual, Newton's method poses two challenges: the efficient solution of the linear system (13) and globalization. Below, we will describe a full-space algorithm that addresses these issues.

IV.B. Full-space Inexact-Newton Algorithm

In this work we use a full-space algorithm that is based on the Lagrange-Newton-Krylov-Schur (LNKS) method of Biros and Ghattas.[6,7] We briefly summarize our implementation below.

- We use the same homotopy-based globalization strategy adopted for the reduced-space algorithms, i.e. the term $\frac{\nu\beta}{2}(\mathbf{x}-\mathbf{x}_0)^T(\mathbf{x}-\mathbf{x}_0)$ is added to the objective function, and ν is gradually reduced to zero.
- A Krylov iterative method (e.g. FGMRES) is used to solve the Newton-update equation (13). The full-space Hessian-vector products are computed using a forward-difference approximation.
- We adopt the preconditioner $\tilde{P}_2$ from Ref. 6, which was found to be the most efficient in terms of CPU time. This preconditioner approximates the full-space Hessian by dropping second-order terms, with

the exception of $\mathcal{L}_{\mathbf{x},\mathbf{x}}$, which is replaced with a L-BFGS quasi-Newton approximation. In addition, the Jacobian $\mathcal{R}_{\mathbf{u}}$ is replaced with a suitable preconditioner. Thus,

$$\tilde{P}_2 = \begin{bmatrix} 0 & A & P\mathcal{R}_{\mathbf{x}} \\ A^T & 0 & 0 \\ \mathcal{R}_{\mathbf{x}}^T P & 0 & \hat{H} \end{bmatrix},$$

where A and A^T denote preconditioners for the linearized primal and dual PDEs, respectively. Note that the global preconditioner has full rank; by assumption, A and A^T are invertible, as is $\hat{H}$.

V. Results

V.A. Steady quasi-one-dimensional nozzle flow

Our first comparison of the three optimization algorithms uses the inverse design of an inviscid nozzle. The flow is modelled using the quasi-one-dimensional Euler equations. For a spatially varying nozzle area, $A(x)$, the governing equations are

$$\frac{\partial \mathcal{F}}{\partial x} - \mathcal{G} = 0, \qquad \forall\, x \in [0,1], \tag{14}$$

where the flux and source are given by

$$\mathcal{F} = \begin{pmatrix} \rho u A \\ (\rho u^2 + p)A \\ u(e+p)A \end{pmatrix}, \qquad \text{and} \quad \mathcal{G} = \begin{pmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{pmatrix},$$

respectively. The unknown variables are the density, ρ, momentum per unit volume, ρu, energy per unit volume, e, and pressure, p. The underdetermined system is closed using the ideal gas law equation of state for the pressure: $p = (\gamma - 1)(e - \frac{1}{2}\rho u^2)$.

Boundary conditions at the inlet and outlet are provided by the exact solution, which is determined using the Mach relations. The stagnation temperature and pressure are 300K and 100 kPa, respectively. The specific gas constant is taken to be 287 J/(kg K) and the critical nozzle area is $A^* = 0.8$. The equations and variables are nondimensionalized using the density and sound speed at the inlet.

The governing equations (14) are discretized using a third-order accurate diagonal-norm summation-by-parts scheme,[25,26] with boundary conditions imposed weakly using penalty terms.[27,28] Scalar third-order accurate artificial dissipation is added to prevent oscillatory solutions.[29] The discretized Euler equations are solved using a Newton-GMRES algorithm.[30,31] GMRES[32] is preconditioned using an LU factorization of a first-order accurate discretization based on nearest-neighbours. The linear adjoint problems are also solved using GMRES with the appropriate LU or $\mathrm{U}^T\mathrm{L}^T$ factorized first-order preconditioner.

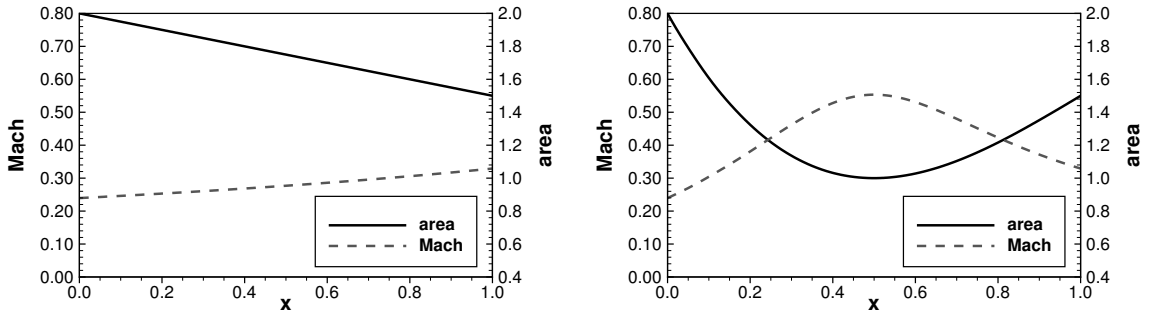
The nozzle area is parameterized using a cubic b-spline with an open uniform knot vector. The control points at the ends of the nozzle are fixed such that the area satisfies $A(0) = 2$ and $A(1) = 1.5$. The design variables are the remaining B-spline control points. The initial design $\mathbf{x}_0$ sets the control points to recover a nozzle with a linearly varying area between the fixed inlet and outlet. Figure 1(a) shows the initial nozzle area and Mach number distribution. The target nozzle area is a cubic function of x that passes through the inlet and outlet areas and has a local minimum at $x = 0.5$ given by $A(0.5) = 1$; see Figure 1(b).

The objective function corresponds to an inverse problem for the pressure:

$$\mathcal{J} = \int_0^1 \frac{1}{2}(p - p_{\text{targ}})^2\, dx. \tag{15}$$

The objective function is discretized using a quadrature based on the SBP norm.[33] The functional is fourth-order accurate, because the discretization is third-order accurate and adjoint consistent.[34–36] The target pressure p_{targ} is found by solving the *discretized* equations with the nozzle area specified by the target area.

The optimal nozzle area produced by all three algorithms is indistinguishable from the target nozzle area, shown in Figure 1(b) together with the corresponding Mach number variation.



(a) initial (b) optimized

Figure 1. Nozzle area and corresponding Mach number for the initial design (left) and optimized design (right).

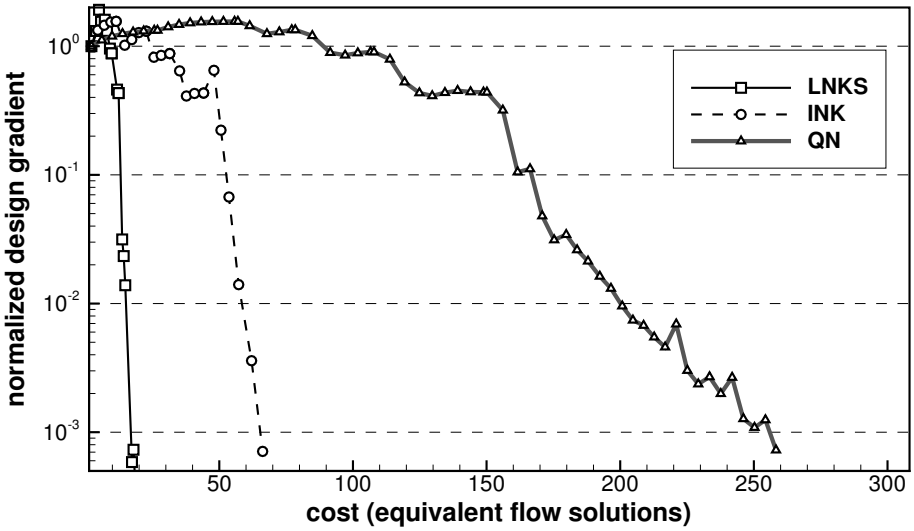


Figure 2. Convergence history of the normalized (design) gradient norm for the nozzle optimization problem with 60 design variables.

The optimization algorithms begin with $\nu = 0.5$. The value of ν is fixed until the reduced-gradient norm of the modified problem is less than 10% of its initial value (i.e. $\tau_{\text{global}} = 0.1$). The continuation parameter is then decreased by 0.1, and the process repeats until there is a three-order drop in the reduced-gradient norm of the original problem (i.e. $\tau = 10^3$). In the case of the full-space algorithm, the primal and dual residual norms must be reduced below 10^{-6} for convergence of the target problem. Both the full- and reduced-space INK algorithms use constant forcing parameters of $\eta_k = 0.1$ in the Krylov solvers. The second-order adjoint problems for the reduced-Hessian-vector products are solved to a tolerance of $0.09 < \eta_k$.

Figure 2 compares the convergence histories of the reduced-gradient norm for the three optimization algorithms. The results shown are for 60 design variables, and computation cost is measured in terms of equivalent flow solutions; more precisely, the cost is the number of preconditioner calls (both primal and dual) made by the optimization algorithm divided by the number of preconditioner calls made by a single flow solution.

The LNKS and INK algorithms exhibit rapid convergence typical of inexact-Newton schemes, and both are significantly faster than the quasi-Newton method. In particular, LNKS is more than an order of magnitude faster than the quasi-Newton method for this problem, and the INK algorithm is approximately 4 times faster.

We are interested in how the algorithms scale as we increase the number of design variables. Figure 3 plots the number of equivalent flow solutions required to converge the reduced-gradient norm 3 orders of magnitude, for design-space sizes between 20 and 60. Note that the quasi-Newton scheme failed for 50 design variables, so the results reported are for 49 design variables. For the design spaces considered, the

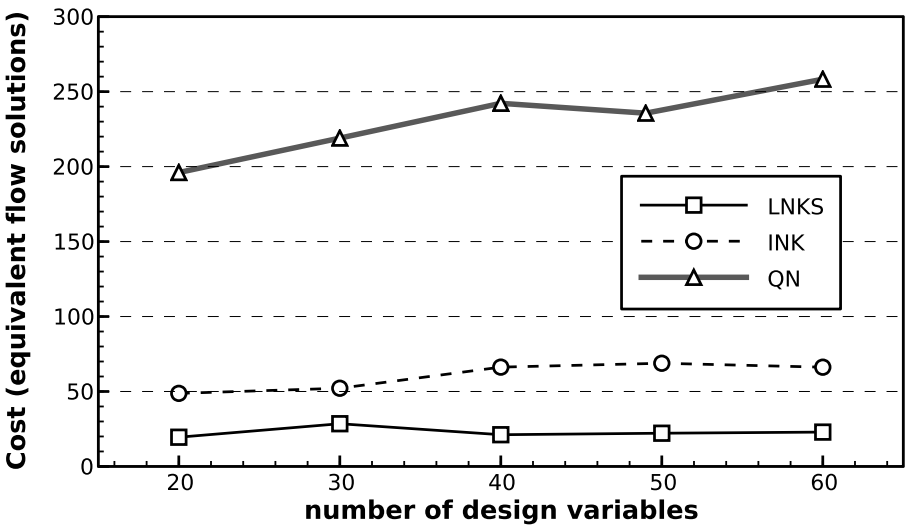


Figure 3. Cost of the nozzle optimization based on number of equivalent flow solutions.

LNKS and INK algorithms are relatively insensitive to the number of design variables, in contrast with the quasi-Newton method.

V.B. Unsteady one-dimensional acoustics problem

Next we consider an optimization constrained by an unsteady flow problem. Here, the design variables define a dynamic control that is optimized to minimize acoustic disturbances. We model the flow using the one-dimensional unsteady Euler equations:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathcal{F}}{\partial x} = \mathcal{G}, \qquad \forall\, x \in [0,1],\ t \in [0,T], \tag{16}$$

where $\mathcal{G}$ is a general source term and the conservative variables and the flux are given by

$$\mathbf{q} = \begin{pmatrix} \rho \\ \rho u \\ e \end{pmatrix}, \qquad \text{and} \quad \mathcal{F} = \begin{pmatrix} \rho u \\ (\rho u^2 + p) \\ u(e+p) \end{pmatrix}.$$

As before, the system is closed using the ideal gas law equation of state.

The initial condition for the flow is a momentum perturbation on a uniform supersonic flow from left to right. In non-dimensional terms

$$\mathbf{q}(x,0) = \begin{pmatrix} \rho_0 \\ \rho_0 u_0 \\ e_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \widetilde{\rho u}(x) \\ 0 \end{pmatrix},$$

where

$$\widetilde{\rho u}(x) = -\frac{1}{100} e^{-\frac{(x-x_c)^2}{\sigma^2}}.$$

The center of the momentum perturbation is $x_c = 0.25$ and $\sigma = 0.05$. The (nondimensionalized) background flow is defined by $\rho_0 = 1$, $u_0 = 1$, and $e_0 = 1$. This initial condition creates a disturbance composed of both acoustic waves. As the solution evolves, the faster acoustic wave overtakes the slower one, resulting in two disturbances moving from left to right.

Table 1. **Algorithm comparison for the unsteady problem.**

	nonlinear it.	precond. calls	equiv. flow
quasi-Newton	34	428 410	264.1
INK	3	85 780	52.9
LNKS	6	152 142	93.8

The objective function is the following space-time integral:

$$\mathcal{J} = \int_0^T \int_0^1 \kappa(x) \frac{1}{2} (p - p_{\text{targ}})^2 \, dx \, dt,$$

where $\kappa(x)$ is a weighting kernel defined by

$$\kappa(x) = e^{-\frac{(x-x_\kappa)^2}{\sigma^2}},$$

with $x_\kappa = 0.85$. The target pressure is the unperturbed background pressure $p_{\text{targ}} = p_0 = 0.2$.

The Euler equations are discretized in space using a fourth-order accurate diagonal-norm SBP discretization and 201 uniformly spaced nodes. The second-order accurate midpoint rule is used to advance the solution in time. The nonlinear systems that arise at each iteration are solved using the same Newton-Krylov algorithm described in the preceding section. The time domain is divided into 200 intervals.

The control is a dynamic momentum source located at $x = 0.5$ and Gaussian in shape. The design variables are the control's magnitude at the midpoint of each time step; thus, as the time domain is refined the design space is also refined. The source vector used in the midpoint rule for the time slab $[T_k, T_{k+1}]$ is given by

$$\mathcal{G}_{k+\frac{1}{2}}^T = \begin{pmatrix} 0 & s_k e^{-\frac{(x-0.5)^2}{\sigma^2}} & 0 \end{pmatrix}, \tag{17}$$

where s_k is the value of the k^{th} design variable. Since there are 200 time steps, there are also 200 design variables.

The results for this optimization problem are summarized in Table 1. The algorithms use the same parameters as the previous example, except that no parameter continuation is necessary, so $\nu = 0$. The reduced-gradient norm has been decreased by three orders of magnitude. As before, the INK and LNKS methods are more efficient than the quasi-Newton approach. Somewhat surprisingly, reduced-space INK requires fewer equivalent flow solutions than LNKS on this problem.

VI. Conclusions

The algorithmic scaling of optimization algorithms must be considered if engineers are to tackle ever more complex designs using PDE-constrained optimization. Popular reduced-space quasi-Newton methods, while relatively easy to implement, scale poorly as the design-space dimension is increased. In contrast, full-space methods like LNKS offer excellent scaling, but implementation challenges limit their broad acceptance.

To find a suitable compromise, we have developed a novel inexact-Newton-Krylov (INK) method for PDE-constrained optimization that is applied in the reduced space. A novel element of the INK method is the use of inexact-Hessian-vector products, which is critical to its efficiency. Numerical experiments confirm that the proposed INK method has good algorithmic scaling as the number of design variables grows. Moreover, the reduced-space INK algorithm is significantly easier to implement than full-space one-shot methods.

A. Derivation of the reduced-space Hessian-vector product

Let $\mathbf{w} \in \mathbb{R}^m$ be a arbitrary vector in the design space and let $\mathbf{g} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \boldsymbol{\psi})$ be the reduced gradient of $\mathcal{J}$; see (5). We begin by recognizing that the inner product $\mathbf{g}^T\mathbf{w}$ is a functional whose total derivative is the desired Hessian-vector product $H\mathbf{w} \equiv \mathbf{z}$. This functional depends on the design variables, the state variables, and the (first-order) adjoint variables. The state variables and adjoint variables are implicit functions of the

design variables through their respective governing equations, so the appropriate Lagrangian for computing the total derivative of $\mathbf{g}^T\mathbf{w}$ is

$$\hat{\mathcal{L}}(\mathbf{x},\mathbf{u},\boldsymbol{\psi},\mathbf{v},\boldsymbol{\lambda}) \equiv \mathbf{g}^T\mathbf{w} + \boldsymbol{\lambda}^T P\mathcal{R}(\mathbf{x},\mathbf{u}) + \mathbf{v}^T\mathcal{S}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}).$$

We have introduced two additional adjoint variables: $\boldsymbol{\lambda}$ for the primal residual $\mathcal{R}$ and $\mathbf{v}$ for the residual of (3b), denoted by $\mathcal{S}$:

$$\mathcal{S} \equiv \mathcal{R}_{\mathbf{u}}^T P\boldsymbol{\psi} + \mathcal{J}_{\mathbf{u}}^T. \tag{11}$$

The desired Hessian-vector product is the derivative of $\hat{\mathcal{L}}$ with respect to $\mathbf{x}$:

$$H\mathbf{w} = \hat{\mathcal{L}}_{\mathbf{x}} = \mathbf{g}_{\mathbf{x}}^T\mathbf{w} + \boldsymbol{\lambda}^T P\mathcal{R}_{\mathbf{x}}(\mathbf{x},\mathbf{u}) + \mathbf{v}^T\mathcal{S}_{\mathbf{x}}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}). \tag{12}$$

Note that the partial derivatives with respect to $\mathbf{x}$ treat $\mathbf{u}$, $\boldsymbol{\psi}$, $\boldsymbol{\lambda}$, and $\mathbf{v}$ as constant. For example,

$$\begin{aligned}\mathbf{g}_{\mathbf{x}} &= \partial_{\mathbf{x}}\left[\mathcal{J}_{\mathbf{x}} + \boldsymbol{\psi}^T P\mathcal{R}_{\mathbf{x}}\right] \\ &= \mathcal{J}_{\mathbf{xx}} + \boldsymbol{\psi}^T P\mathcal{R}_{\mathbf{xx}}.\end{aligned}$$

While there are second derivatives in the expression for $H\mathbf{w}$, they always appear as products with other vectors. Thus, they can be approximated using finite-difference approximations. To illustrate this, consider the forward-difference approximation of $\mathbf{g}_{\mathbf{x}}^T\mathbf{w}$, which is given by

$$\mathbf{g}_{\mathbf{x}}^T\mathbf{w} = \frac{\mathbf{g}(\mathbf{x}+\epsilon\mathbf{w},\mathbf{u},\boldsymbol{\psi}) - \mathbf{g}(\mathbf{x},\mathbf{u},\boldsymbol{\psi})}{\epsilon} + \mathrm{O}(\epsilon),$$

where $\epsilon > 0$ is the perturbation parameter.

The expression for $H\mathbf{w}$ depends on $\mathbf{v}$ and $\boldsymbol{\lambda}$, so we need to solve for these adjoint variables. The linear systems that determine $\mathbf{v}$ and $\boldsymbol{\lambda}$ can be found by differentiating $\hat{\mathcal{L}}$ with respect to $\boldsymbol{\psi}$ and $\mathbf{u}$, respectively. After differentiating and rearranging, we arrive at

$$P\mathcal{R}_{\mathbf{u}}\mathbf{v} = -\mathbf{g}_{\boldsymbol{\psi}}^T\mathbf{w}, \tag{9}$$

$$\mathcal{R}_{\mathbf{u}}^T P\boldsymbol{\lambda} = -\mathbf{g}_{\mathbf{u}}^T\mathbf{w} - \mathcal{S}_{\mathbf{u}}^T\mathbf{v}. \tag{10}$$

As noted in the text, the right-hand-side of (9) reduces to $-\mathcal{R}_{\mathbf{x}}^T P\mathbf{w}$, which can be computed in the same way as the term $\boldsymbol{\psi}^T P\mathcal{R}_{\mathbf{x}}$ that appears in the gradient. The right-hand-side of (10) contains second-derivatives, but as above, these can be approximated using finite-difference approximations. For example,

$$\mathcal{S}_{\mathbf{u}}^T\mathbf{v} = \frac{\mathcal{S}(\mathbf{x},\mathbf{u}+\epsilon\mathbf{v},\boldsymbol{\psi}) - \mathcal{S}(\mathbf{x},\mathbf{u},\boldsymbol{\psi})}{\epsilon} + \mathrm{O}(\epsilon).$$

References

[1]Frank, P. D. and Shubin, G. R., "A comparison of optimization-based approaches for a model computational aerodynamics design problem," *Journal of Computational Physics*, Vol. 98, No. 1, Jan. 1992, pp. 74–89.

[2]Ta'asan, S., Kuruvila, G., and Salas, M. D., "Aerodynamic design and optimization in one shot," *The 30th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA–1992–25, Jan. 1992.

[3]Iollo, A., Salas, M. D., and Ta'asan, S., "Shape optimization governed by the Euler equations using an adjoint method," Tech. Rep. ICASE 93-78, NASA Langley Research Center, Hampton, Virginia, Nov. 1993.

[4]Feng, D. and Pulliam, T. H., "An all-at-once reduced Hessian SQP scheme for aerodynamic design optimization," Tech. Rep. RIACS-TR-95.19, NASA Ames Research Center, Moffett Field, California, Oct. 1995.

[5]Gatsis, J. and Zingg, D. W., "A fully-coupled Newton-Krylov algorithm for aerodynamic optimization," *16th AIAA Computational Fluid Dynamics Conference*, No. AIAA–2003–3956, Orlando, Florida, United States, June 2003.

[6]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.

[7]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.

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

[9]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.

[10]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.

[11]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.

[12]Saad, Y., "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal on Scientific and Statistical Computing*, Vol. 14, No. 2, 1993, pp. 461–469.

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

[14]Pierce, N. A. and Giles, M. B., "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, Vol. 65, No. 3, 2000, pp. 393–415.

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

[16]Fletcher, R., *Practical methods of optimization*, A Wiley-Interscience Publication, Wiley, 2nd ed., 2000.

[17]Allgower, E. L. and Georg, K., *Acta Numerica*, chap. Continuation and Path Following, Cambridge University Press, 1993, pp. 1–64.

[18]Dembo, R. S., Eisenstat, S. C., and Steihaug, T., "Inexact Newton methods," *SIAM Journal on Numerical Analysis*, Vol. 19, No. 2, 1982, pp. 400–408.

[19]Nash, S. G., "A survey of truncated-Newton methods," *Journal of Computational and Applied Mathematics*, Vol. 124, No. 1-2, 2000, pp. 45–59.

[20]Morales, J. L. and Nocedal, J., "Automatic Preconditioning by Limited Memory Quasi-Newton Updating," *SIAM Journal on Optimization*, Vol. 10, No. 4, 2000, pp. 1079–1096.

[21]Paige, C. C. and Saunders, M. A., "Solution of Sparse Indefinite Systems of Linear Equations," *SIAM Journal on Numerical Analysis*, Vol. 12, No. 4, 1975, pp. 617–629.

[22]Hinze, M. and Pinnau, R., "Second-order approach to optimal semiconductor design," *Journal of Optimization Theory and Applications*, Vol. 133, 2007, pp. 179–199.

[23]Golub, G. H. and Ye, Q., "Inexact Preconditioned Conjugate Gradient Method with Inner-Outer Iteration," *SIAM Journal on Scientific Computing*, Vol. 21, No. 4, 1999, pp. 1305–1320.

[24]Simoncini, V. and Szyld, D. B., "Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing," *SIAM Journal on Scientific Computing*, Vol. 25, No. 2, Jan. 2003, pp. 454–477.

[25]Kreiss, H. O. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations*, edited by C. de Boor, Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[26]Strand, B., "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47–67.

[27]Funaro, D. and Gottlieb, D., "A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations," *Mathematics of Computation*, Vol. 51, No. 184, Oct. 1988, pp. 599–613.

[28]Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236.

[29]Mattsson, K., Svärd, M., and Nordström, J., "Stable and accurate artificial dissipation," *Journal of Scientific Computing*, Vol. 21, No. 1, 2004, pp. 57–79.

[30]Keyes, D. E., "Aerodynamic applications of Newton-Krylov-Schwarz solvers," *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, Springer, New York, 1995, pp. 1–20.

[31]Nielsen, E. J., Walters, R. W., Anderson, W. K., and Keyes, D. E., "Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code," *12th AIAA Computational Fluid Dynamics Conference*, San Diego, CA, 1995.

[32]Saad, Y. and Schultz, M. H., "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, July 1986, pp. 856–869.

[33]Hicken, J. E. and Zingg, D. W., "Summation-by-parts operators and high-order quadrature," *Journal of Computational and Applied Mathematics*, Vol. 237, No. 1, Jan. 2013, pp. 111–125.

[34]Lu, J. C., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[35]Hartmann, R., "Adjoint Consistency Analysis of Discontinuous Galerkin Discretizations," *SIAM Journal on Numerical Analysis*, Vol. 45, No. 6, 2007, pp. 2671–2696.

[36]Hicken, J. E. and Zingg, D. W., "The role of dual consistency in functional accuracy: error estimation and superconvergence," *20th AIAA Computational Fluid Dynamics Conference*, No. AIAA-2011-3855, Honolulu, Hawaii, United States, June 2011.