VI International Conference on Adaptive Modeling and Simulation
ADMOS 2013
J. P. Moitinho de Almeida, P. Díez, C. Tiago and N. Parés (Eds)

INEXACT-HESSIAN-VECTOR PRODUCTS FOR EFFICIENT REDUCED-SPACE PDE-CONSTRAINED OPTIMIZATION

Jason E. Hicken

Department of Mechanical, Aerospace, and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York, 12180, United States
e-mail: hickej2@rpi.edu, www.optimaldesignlab.com

Key words: Adaptive Modeling, Simulation, PDE-constrained optimization, second-order adjoints, matrix-free, Hessian-vector products

Abstract. We investigate reduced-space Newton-Krylov (NK) algorithms for engineering parameter optimization problems constrained by partial-differential equations. We review reduced-space and full-space optimization algorithms, and we show that the efficiency of the reduced-space strategy can be improved significantly with inexact-Hessian-vector products computed using approximate second-order adjoints. Results demonstrate that the proposed reduced-space NK algorithm has excellent scaling that makes it suitable for large-scale optimization problems. Moreover, reduced-space NK combines the attractive attributes of both reduced-space quasi-Newton methods and full-space approaches — namely, modularity, robustness, and scalibilty.

1 Introduction

Partial differential equation (PDE) constrained optimization problems can be posed in the full-space or the reduced-space. In full-space formulations the PDE state variables — e.g. pressure and velocity for incompressible flows — are included as optimization variables, and the PDE becomes an explicit constraint in the optimization. In contrast, reduced-space formulations treat the state variables as implicit functions of the design variables: for a given set of design variables the PDE is solved for the states.

In practice, engineers often prefer reduced-space formulations. Reduced-space methods lend themselves to modularity, so implementation is typically easier than full-space methods. Unfortunately, conventional reduced-space optimization algorithms exhibit poor algorithmic scaling. For example, the computational cost of limited-memory quasi-Newton methods is often proportional to the number of design variables. This scaling limits the number of design variables that can be considered.

Motivated by the above observations, we consider reduced-space inexact-Newton-Krylov (INK) algorithms, which offer the potential for design-dimension-independent algorithmic scaling. One of the challenges with reduced-space INK methods for PDE-constrained optimization is the efficient computation of Hessian-vector products needed by the Krylov solver. In particular, it is widely believed that these products must computed with high accuracy to avoid convergence difficulties. This accuracy requirement can render reduced-space INK methods orders of magnitude more expensive than full-space methods [11]. In this paper, we argue that the Hessian-vector products can be computed inexactly, and numerical examples demonstrate that the resulting reduced-space INK algorithm offers an attractive alternative to its full-space counterpart.

2 PDE-constrained Optimization: Formulations and Algorithms

In this section we briefly review the generic PDE-constrained optimization problem and highlight commonly used formulations and solution strategies. For a comprehensive review of solution methods see, for example, [1].

2.1 Problem and Notation

We are interested in solving the following PDE-constrained optimization problem:

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

The objective functional is $\mathcal{J}$, which we will assume is C^2 continuous on its domain. The variables $\mathbf{x}$ and $\mathbf{u}$ 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. The PDE itself, together with appropriate boundary and initial conditions, is represented by the equation $\boldsymbol{\mathcal{R}}(\mathbf{x},\mathbf{u})=\mathbf{0}$.

A local solution of (1) must satisfy the first-order optimality conditions, which can be found by differentiating the Lagrangian. In order to define the Lagrangian of (1), we first introduce the symmetric positive definite matrix $P\in R^{n\times n}$ that defines a discrete inner product appropriate to the chosen discretization of the PDE. Then, the Lagrangian is given by

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

where $\boldsymbol{\psi}\in\mathbb{R}^n$ are the Lagrange multipliers, also called the adjoint or costate variables in the context of PDE-constrained optimization.

Thus, the first-order optimality conditions for a solution to (1) are [20]:

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

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

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

Subscripted variables indicate differentiation with respect to that variable, e.g. $\mathcal{J}_{\mathbf{u}} \equiv \partial\mathcal{J}/\partial\mathbf{u}$. The first-order conditions (3) are also called the Karush-Kuhn-Tucker, or KKT, conditions.

2.2 Full-space Approach

The KKT conditions are a set of nonlinear algebraic equations, so a natural solution strategy is Newton's method with an appropriate globalization. As usual, the potential for rapid convergence makes Newton's method attractive. The full-space approach that we adopt here is based on the Lagrange-Newton-Krylov-Schur (LNKS) method of Biros and Ghattas [2, 3]. One difference between their method and the present scheme is the parameter continuation used for globalization; more details on this globalization can be found in [13].

The Newton update equation corresponding to (3) is

$$\begin{bmatrix} 0 & \mathcal{L}_{\boldsymbol{\psi},\mathbf{u}} & \mathcal{L}_{\boldsymbol{\psi},\mathbf{x}} \\ \mathcal{L}_{\mathbf{u},\boldsymbol{\psi}} & \mathcal{L}_{\mathbf{u},\mathbf{u}} & \mathcal{L}_{\mathbf{u},\mathbf{x}} \\ \mathcal{L}_{\mathbf{x},\boldsymbol{\psi}} & \mathcal{L}_{\mathbf{x},\mathbf{u}} & \mathcal{L}_{\mathbf{x},\mathbf{x}} \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{\psi}_k \\ \Delta\mathbf{u}_k \\ \Delta\mathbf{x}_k \end{pmatrix} = - \begin{bmatrix} \mathcal{L}_{\boldsymbol{\psi}} \\ \mathcal{L}_{\mathbf{u}} \\ \mathcal{L}_{\mathbf{x}} \end{bmatrix}, \tag{4}$$

where the subscript k denotes the current Newton iteration. Solving (4) with a direct method is usually impractical for large-scale PDE-constrained optimization problems; therefore, the approach adopted in LNKS is to use an inexact-Newton-Krylov approach. An advantage of using an inexact-Newton method [7] is that the KTT system can be solved approximately and inexpensively during the early Newton iterations

A Krylov-based approach avoids the need to form the KKT matrix explicitly, since Krylov methods use matrix-vector products. However, to be effective, Krylov-iterative solvers must be preconditioned. We adopt the preconditioner $\tilde{P}_2$ from [2], which was found to be efficient in terms of CPU time. This preconditioner approximates the full-space Hessian by dropping second-order derivatives, with the exception of $\mathcal{L}_{\mathbf{x},\mathbf{x}}$, which is replaced with a L-BFGS quasi-Newton approximation. In addition, the Jacobian $\boldsymbol{\mathcal{R}}_{\mathbf{u}}$ is replaced with a suitable preconditioner A.

2.3 Reduced-space Approaches

Full-space methods for PDE-constrained optimization are efficient [11, 2, 12], often requiring only a few multiples, typically O(10), of the PDE solution cost. Nevertheless, full-space methods have significant disadvantages: lack of appropriate optimization

libraries; inability to leverage specialized globalization strategies, and; potentially prohibitive memory requirements. These disadvantages motivate reduced-space methods, which we review in this section.

For a valid set of control variables, the state equations will be invertible. Thus, we can invoke the implicit function theorem and define the state variables in terms of the control variables: $\mathbf{u} = \mathbf{u}(\mathbf{x})$. Consequently, the optimization problem (1) reduces to

$$\text{minimize} \quad \mathcal{J}(\mathbf{x}, \mathbf{u}(\mathbf{x})), \quad \mathbf{x} \in \mathbb{R}^m. \tag{5}$$

The first-order optimality conditions for a solution of (5) are the same as (1); after all, they solve the same problem. The difference is, in the reduced formulation, the primal and adjoint PDEs (equations (3a) and (3b)) must be solved at each optimization iteration.

In other words: in reduced-space formulations the optimization algorithm is responsible for satisfying (3c), while the user must satisfy (3a) and (3b). The advantage of this approach is that efficient software libraries are usually available to solve the primal and adjoint PDEs; typically, these libraries are parallel and have specialized globalizations tuned to their discipline. The disadvantage is the added cost of accurately solving the state and adjoint equations early in the optimization process.

We now turn to the problem of solving the first-order condition (3c) for $\mathbf{x}$. As in the full-space, we begin with Newton's method. Linearizing about the current design, $\mathbf{x}_k$, we find the Newton-update equation

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

where $H_k \equiv (\partial \mathbf{g}/\partial \mathbf{x})_k$ is the reduced Hessian evaluated at $\mathbf{x}_k$, and $\mathbf{g}_k$ is the reduced gradient evaluated at $\mathbf{x}_k$. One of the challenges for reduced-space formulations is capturing the second-order information contained in H_k. This is because the reduced Hessian is the total derivative of the gradient with respect to $\mathbf{x}$, so variations in $\mathbf{u}$ and $\boldsymbol{\psi}$ must be accounted for

Quasi-Newton methods are a popular and successful class of algorithms that approximate the Hessian, thereby circumventing the need to compute the total derivative of $\mathbf{g}$. For quasi-Newton methods the Newton-update equation is replaced with

$$B_k \Delta \mathbf{x}_k = -\mathbf{g}_k, \tag{7}$$

where B_k is a quasi-Newton approximation to H_k. In this work, we consider the limited-memory BFGS quasi-Newton method [17] globalized with a strong-Wolfe-type line search algorithm [8]. For the Armijo sufficient-decrease condition we use the parameter $c_1 = 10^{-6}$, and for the curvature condition we use $c_2 = 0.999$; see [20] for the definition of these parameters.

Quasi-Newton methods are simple and effective for many problems; however, they are not necessarily suited to large-scale design spaces. For large problems, we must often resort to limited-memory quasi-Newton methods; while BFGS has a superlinear asymptotic

convergence rate [20], its limited-memory variant has only a linear rate of convergence [17]. In addition, during the early stages of optimization the quasi-Newton approximation may not capture curvature accurately, and this can lead to many subiterations in the line search or many trust-region radius updates. For these reasons, the computational cost of quasi-Newton methods typically grows with problem size; linear scaling is not unusual.

We want to retain the modularity of the reduced-space approach, but with algorithmic scaling that does not grow with problem size. The solution pursued here is to apply an inexact-Newton-Krylov strategy in the reduced-space [14, 4]. The inexact-Newton approach replaces the solution of (6) with the condition that $\Delta\mathbf{x}_k$ must satisfy

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

where η_k is the so-called forcing parameter and $\|\cdot\|$ denotes the L2 norm. To achieve superlinear convergence, we use $\eta_k = 0.1\min\left[1,\ (\|\mathbf{g}_k\|/\|\mathbf{g}_0\|)^{\frac{1}{2}}\right]$ [7]. To avoid oversolving the linear problem when the iterates are near the desired nonlinear tolerance, we set $\eta_k \leftarrow \max\left(\eta_k, \tau\|\mathbf{g}_0\|/\|\mathbf{g}_k\|\right)$, where τ is the nonlinear tolerance.

Similar to the full-space approach, using a Krylov method avoids the need to form the Hessian explicitly; only Hessian-vector products are required. The computation of the Hessian-vector products plays an important role in the efficiency of reduced-space INK and is discussed in detail in the following section.

To globalize the reduced-space INK algorithm, we use the Steihaug-Toint variant of the conjugate-gradient method [24, 25] in a standard trust-region framework [6]. In the context of optimization, limited-memory BFGS has been shown to be an effective preconditioner for Krylov iterative methods [19], and this is the preconditioner adopted here.

3 Inexact Hessian-vector Products & Second-order Adjoints

The Hessian-vector products needed in reduced-space INK methods can be computed using second-order adjoints [26]. By defining a new functional, $\mathbf{g}^T\mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^m$ is an arbitrary vector, and including both the state and (first-order) adjoint equations as constraints, one can show that the Hessian-vector product is given by [2, 14, 13]

$$H\mathbf{w} = \mathbf{g}_\mathbf{x}^T\mathbf{w} + \boldsymbol{\lambda}^T P\boldsymbol{\mathcal{R}}_\mathbf{x}(\mathbf{x},\mathbf{u}) + \mathbf{v}^T\boldsymbol{\mathcal{S}}_\mathbf{x}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}), \tag{9}$$

where $\boldsymbol{\mathcal{S}}$ denotes the first-order adjoint residual

$$\boldsymbol{\mathcal{S}}(\mathbf{x},\mathbf{u},\boldsymbol{\psi}) \equiv \boldsymbol{\mathcal{R}}_\mathbf{u}^T P\boldsymbol{\psi} + \mathcal{J}_\mathbf{u}^T. \tag{3b}$$

Note that the partial derivatives with respect to $\mathbf{x}$ in (9) treat $\mathbf{u}$, $\boldsymbol{\psi}$, $\boldsymbol{\lambda}$, and $\mathbf{v}$ as constants (i.e. these are not total derivatives).

The variables $\mathbf{v} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}^n$ are the second-order adjoint variables of the functional $\mathbf{g}^T\mathbf{w}$ corresponding to the primal and adjoint equations, respectively. The second-

order adjoints satisfy the equations (see, for example, [13])

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

$$\boldsymbol{\mathcal{R}}_{\mathbf{u}}^T P\boldsymbol{\lambda} = -\mathbf{g}_{\mathbf{u}}^T\mathbf{w} - \boldsymbol{\mathcal{S}}_{\mathbf{u}}^T\mathbf{v}. \tag{11}$$

The assembly and solution of the second-order adjoint equations (10) and (11) deserve some remarks.

- The system matrix of (10) is the Jacobian of the primal equations, and the system matrix of (11) is the transposed Jacobian. Most adjoint-based optimization algorithms are capable of solving for systems involving $\boldsymbol{\mathcal{R}}_{\mathbf{u}}^T$, and adapting these algorithms to solve systems involving $\boldsymbol{\mathcal{R}}_{\mathbf{u}}$ is straightforward.

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

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

  This term can be computed in the same way as the second term in the reduced gradient (3c).

- The right-hand side vector of the adjoint system for $\boldsymbol{\lambda}$ involves second derivatives. These terms amount to direction derivatives and can easily be computed using finite-difference approximations, the complex-step method, or algorithmic differentiation. See the appendix of [13] for details.

The Hessian-vector product involves the solution of the two second-order adjoint equations (10) and (11). These equations are typically solved using iterative methods, which suggests that we might reduce computational cost by sacrificing accuracy. In other words, can we compute inexact Hessian-vector products rather than exact products?

The use of inexact Hessian-vector products is entirely appropriate in the context of an inexact-Newton method: why compute accurate products when we only want an approximate solution anyway? Indeed, the analysis of Simoncini and Szyld [23] indicates that the accuracy of matrix-vector products can be relaxed provided the Krylov basis remains orthogonal, which is the case for methods like GMRES [22] and FGMRES [21]. On the other hand, inexact products can pose convergence problems when the Krylov basis is not explicitly orthogonalized, e.g. in the CG method [10, 23], although no such problems were observed in the present study.

Let $\widetilde{H_k\mathbf{w}}$ be the inexact Hessian-vector product (9) computed using the iteratively-solved second-order adjoints $\tilde{\mathbf{v}}$ and $\tilde{\boldsymbol{\lambda}}$. Then the error in the Hessian-vector product satisfies

$$\|H_k\mathbf{w} - \widetilde{H_k\mathbf{w}}\| \leq \delta_\lambda \|\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{x}}\| + \delta_v \|P^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-T}(\boldsymbol{\mathcal{S}}_{\mathbf{x}} + \boldsymbol{\mathcal{S}}_{\mathbf{u}}\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{x}})\|,$$

where δ_λ and δ_v are bounds on the second-order adjoint residuals for $\tilde{\boldsymbol{\lambda}}$ and $\tilde{\mathbf{v}}$, respectively, and are defined by

$$\|P\boldsymbol{\mathcal{R}}_{\mathbf{u}}\tilde{\mathbf{v}} + \mathbf{g}_{\boldsymbol{\psi}}^T\mathbf{w}\| \leq \delta_v = \frac{1}{2}\eta_k\|\mathbf{g}_k\|, \tag{12}$$

$$\|\boldsymbol{\mathcal{R}}_{\mathbf{u}}^T P\tilde{\boldsymbol{\lambda}} + \mathbf{g}_{\mathbf{u}}^T\mathbf{w} + \boldsymbol{\mathcal{S}}_{\mathbf{u}}^T\tilde{\mathbf{v}}\| \leq \delta_\lambda = \frac{1}{2}\eta_k\|\mathbf{g}_k\|. \tag{13}$$

Bounds for $\|\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{x}}\|$ and $\|P^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-T}(\boldsymbol{\mathcal{S}}_{\mathbf{x}} - \boldsymbol{\mathcal{S}}_{\mathbf{u}}\boldsymbol{\mathcal{R}}_{\mathbf{u}}^{-1}\boldsymbol{\mathcal{R}}_{\mathbf{x}})\|$ are also required and are more difficult to estimate. For this preliminary work, trial-and-error was used to determine that these terms are O(1) for the problem considered below; future work will investigate a priori methods of bounding these terms.

Based on the bounds δ_v and δ_λ, as well as the O(1) estimates for the remaining terms, we have

$$\|H_k\mathbf{w} - \widetilde{H_k\mathbf{w}}\| \lessapprox \eta_k\|\mathbf{g}_k\|.$$

Thus, the approximate solution of the second-order adjoints, based on (12) and (13), is such that the accuracy of the inexact Hessian-vector products is comparable to the accuracy of the linear solve.

4 Results

In this section we investigate the performance of the reduced-space INK algorithm and compare its performance with that of the reduced-space quasi-Newton (QN) method and the full-space LNKS algorithm.

Our model problem for the investigations will be the inverse design of an inviscid nozzle flow. The PDE constraint for this problem is the quasi-one-dimensional Euler equations, given by

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

where the flux and source are

$$\boldsymbol{\mathcal{F}} = \big(\rho u A,\ \ (\rho u^2 + p)A,\ \ u(e+p)A\big)^T, \qquad \text{and} \qquad \boldsymbol{\mathcal{G}} = \big(0,\ \ p\tfrac{dA}{dx},\ \ 0\big)^T,$$

respectively. The state variables are density (ρ), momentum per unit volume (ρu), and energy per unit volume (e). Pressure, which also appears in the Euler equations, is determined using the ideal-gas equation of state: $p = (\gamma - 1)(e - \frac{1}{2}\rho u^2)$. Finally, $A = A(x)$ denotes the spatially varying nozzle area, which, when discretized, will become our control variable. The boundary values are provided by the exact solution, which is determined using the Mach relations. The exact solution is based on a stagnation temperature of 300K and stagnation pressure of 100 kPa. The critical nozzle area is $A^* = 0.8$ and the gas constant is 287 J/(kg K). In the implementation, the equations and variables have been nondimensionalized using the density and sound speed at the inlet, $x = 0$.

The Euler equations (14) are discretized using a summation-by-parts finite-difference scheme [16]. In particular, the derivatives are approximated using a third-order accurate diagonal-norm operator, and the boundary conditions are imposed weakly using SAT penalty terms [9, 5]. To stabilize the discrete equations, we add scalar third-order accurate artificial dissipation [18].

For the reduced-space formulations, the discretized Euler equations are solved using a Newton-GMRES algorithm [15]. The GMRES [22] Krylov solver is preconditioned using an LU factorization of a first-order accurate discretization that is based on nearest-neighbours and first-order scalar dissipation. The linearized forward problem (10) is also solved using GMRES and the same preconditioner. The adjoint problems are solved using GMRES with the Jacobian-vector products and preconditioners replaced with their transposed versions.

The nozzle area $A(x)$ is discretized using cubic B-splines with open uniform knot vectors. The area is fixed at the ends of the nozzle such that $A(0) = 2$ and $A(1) = 1.5$. The control variables consist of the interior B-spline control points. To avoid confusion between the design variables and the x-coordinate, we will use $\mathbf{A}$ to denote the vector of (interior) B-spline control points. In all cases, the initial design $\mathbf{A}_0$ corresponds to the set of control points that produce the linearly varying area $A(x) = 2 - 0.5x$. The target nozzle area is a cubic function of x that passes through the fixed inlet and outlet areas and has a local minimum at $x = 0.5$ given by $A(0.5) = 1$.

In summary, the optimization problem is

$$\begin{aligned} &\text{minimize} && \mathcal{J}(\mathbf{A}, \mathbf{u}) = \int_0^1 \frac{1}{2}(p(\mathbf{u}) - p_{\text{targ}})^2 \, dx, \qquad \mathbf{A} \in \mathbb{R}^m, \ \mathbf{u} \in \mathbb{R}^{3n}, \\ &\text{subject to} && \boldsymbol{\mathcal{R}}(\mathbf{A}, \mathbf{u}) = \mathbf{0}, \end{aligned} \tag{15}$$

where $\mathbf{u}$ denotes the vector of state variables (ρ, ρu, and e) at each of the n nodes, and $\boldsymbol{\mathcal{R}}(\mathbf{A}, \mathbf{u}) = \mathbf{0}$ denotes the discretized Euler equations. The target pressure p_{targ} is found by solving for $\mathbf{u}$ using the target nozzle area in the Euler equations.

4.1 Dynamic versus fixed tolerance for the inexact-Hessian-vector products

To demonstrate the impact of inexact Hessians on the reduced-space INK algorithm, we solve (15) with the second-order adjoint equations satisfying either 1) a fixed (relative) tolerance of 10^{-6} or 2) the dynamic tolerances (12) and (13).

Table 1 lists the computational cost for these two approaches over a range of design-variable dimensions. The cost is measured in terms of equivalent PDE solutions required to satisfy $\|\mathbf{g}(\mathbf{x}_k)\| \leq \tau \|\mathbf{g}(\mathbf{x}_0)\|$, where $\tau = 10^{-3}$. Specifically, the total number of PDE preconditioner calls (i.e. applications of $(LU)^{-1}$) used during the optimization is divided by the number of preconditioner calls to solve the PDE on the initial geometry.

The results show that, on average, the dynamic tolerance reduces the computational cost by 40.5% relative to the fixed tolerance. This illustrates the importance of using inexact Hessians in reduced-space INK algorithms.

Table 1: Number of equivalent PDE solutions required when using dynamic and static tolerances in second-order adjoint solves.

	number of design variables				
	20	30	40	50	60
fixed tolerance	102.7	103.5	107.5	113.8	108.3
dynamic tolerance	67.4	58.5	60.5	65.5	66.8
improvement (%)	34.3	43.5	43.8	42.5	38.4

4.2 Comparison of Optimization Methods

We now compare the reduced-space INK algorithm with the full-space LNKS algorithm and the reduced-space quasi-Newton algorithm. Figure 1 plots the computational cost of the three algorithms versus the design-variable dimension. Computational cost is measured using the number of equivalent PDE solves, as defined earlier. The optimizations are terminated when the relative reduced-gradient norm is below $\tau = 10^{-3}$. In the case of LNKS, the PDE and adjoint residual norms must be below 10^{-6} their initial values.

As expected, the quasi-Newton algorithm has a strong dependence on the number of design variables. In contrast, the two Newton-based algorithms have much weaker dependence on the dimension of $\mathbf{A}$. Moreover, even for this relatively small problem, there is a clear advantage to using inexact-Newton rather than quasi-Newton optimization; the computational cost is between 4 and 6 times lower using INK, and between 15 and 17 times lower using LNKS.

Comparing reduced-space INK with LNKS, we observe a factor of 3 to 4 reduction in cost using the full-space algorithm. What is not shown in the plot is the robustness issues associated with LNKS. Considerable parameter tuning was necessary to obtain convergence with LNKS, so there is trade-off between robustness and speed when moving from the reduced-space to the full-space. We argue that a factor of 3 to 4 is compensated for by the time needed to find suitable parameters in the full-space approach.

5 Summary and Discussion

We have shown that reduced-space inexact-Newton-Krylov (INK) algorithms offer an attractive compromise between reduced-space quasi-Newton and full-space Newton-type methods. For the nozzle-flow inverse design problem, reduced-space INK was 4 to 6 times faster than the quasi-Newton algorithm and was much less sensitive to the number of design variables. The LNKS full-space algorithm was found to be the most efficient scalable algorithm, but this efficiency comes at the price of robustness.

In general, engineers have focused on reduced-space quasi-Newton and full-space Newton methods. The results presented here indicate that reduced-space INK algorithms should be investigated more broadly, since they offer a scalable route to solving large-

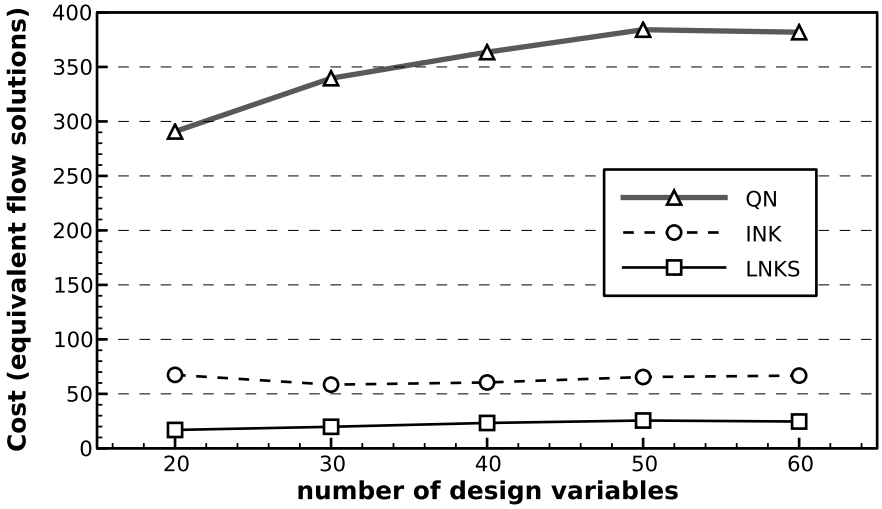

Figure 1: Cost of the nozzle-area optimization based on number of equivalent flow solutions.

scale optimization problems. Moreover, they require limited intrusion into existing PDE solvers and provide increased robustness relative to full-space algorithms.

Inexact-Hessian-vector products play an important role in the reduced-space INK algorithm presented here. The products are computed using second-order adjoints with dynamic tolerances. The use of dynamic tolerances was shown to reduce computational cost by approximately 40% relative to a fixed tolerance. We note that inexact-Hessian-vector products may pose challenges for traditional optimization algorithms that assume symmetry of the Hessian. Our current work is focused on developing optimization algorithms that are robust in the presence of inexactness in the Hessian.

REFERENCES

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

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

[3] ——, *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, 27 (2005), pp. 714–739.

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

[5] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, Journal of Computational Physics, 111 (1994), pp. 220–236.

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

[7] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[8] R. Fletcher, *Practical methods of optimization*, A Wiley-Interscience Publication, Wiley, second ed., 2000.

[9] D. Funaro and D. Gottlieb, *A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations*, Mathematics of Computation, 51 (1988), pp. 599–613.

[10] G. H. Golub and Q. Ye, *Inexact Preconditioned Conjugate Gradient Method with Inner-Outer Iteration*, SIAM Journal on Scientific Computing, 21 (1999), pp. 1305–1320.

[11] E. Haber and U. M. Ascher, *Preconditioned all-at-once methods for large, sparse parameter estimation problems*, Inverse Problems, 17 (2001), pp. 1847–1864.

[12] S. B. Hazra, *Direct treatment of state constraints in aerodynamic shape optimization using simultaneous pseudo-time-stepping*, AIAA Journal, 45 (2007), pp. 1988–1997.

[13] J. E. Hicken and J. J. Alonso, *Comparison of Reduced- and Full-space Algorithms for PDE-constrained Optimization*, in 51st AIAA Aerospace Sciences Meeting, no. AIAA–2013–1043, Grapevine, Texas, United States, Jan. 2013.

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

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

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

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

[18] K. Mattsson, M. Svärd, and J. Nordström, *Stable and accurate artificial dissipation*, Journal of Scientific Computing, 21 (2004), pp. 57–79.

[19] J. L. Morales and J. Nocedal, *Automatic Preconditioning by Limited Memory Quasi-Newton Updating*, SIAM Journal on Optimization, 10 (2000), pp. 1079–1096.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer–Verlag, Berlin, Germany, second ed., 2006.

[21] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific and Statistical Computing, 14 (1993), pp. 461–469.

[22] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[23] V. Simoncini and D. B. Szyld, *Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing*, SIAM Journal on Scientific Computing, 25 (2003), pp. 454–477.

[24] T. Steihaug, *The Conjugate Gradient Method and Trust Regions in Large Scale Optimization*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 626–637.

[25] P. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, 1981, pp. 57–88.

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