PARALLEL NEWTON–KRYLOV METHODS FOR PDE–CONSTRAINED OPTIMIZATION*

GEORGE BIROS[†] AND OMAR GHATTAS[†]

Abstract. Large scale optimization of systems governed by partial differential equations (PDEs) is a frontier problem in scientific computation. The state-of-the-art for solving such problems is reduced-space quasi-Newton sequential quadratic programming (SQP) methods. These take full advantage of existing PDE solver technology and parallelize well. However, their algorithmic scalability is questionable; for certain problem classes they can be very slow to converge. In this paper we propose a full-space Newton-Krylov SQP method that uses the reduced-space quasi-Newton method as a preconditioner. The new method is fully parallelizable; exploits the structure of and available parallel algorithms for the PDE forward problem; and is quadratically convergent close to a local minimum. We restrict our attention to boundary value problems and we solve a model optimal flow control problem, with both Stokes and Navier-Stokes equations as constraints. Algorithmic comparisons, scalability results, and parallel performance on a Cray T3E-900 are presented. On the model problems solved, the new method is a factor of 5–10 faster than reduced space quasi-Newton SQP, and is scalable provided a good forward preconditioner is available.

1. Introduction. Optimization problems that are constrained by partial differential equations (PDEs) arise naturally in many areas of science and engineering. In the sciences, such problems often appear as *inverse problems* in which some of the parameters in a simulation are unavailable, and must be estimated by comparison with physical data. These parameters are typically boundary conditions, initial conditions, sources, or coefficients of a PDE. Examples include empirically-determined parameters in a complex constitutive law, and material properties of a medium that is not directly observable. In engineering, PDE-constrained optimization problems often take the form of *optimal design* or *optimal control* problems. Such problems can occur across the life cycle of engineered systems, including the design phase (as an optimal design problem), the manufacturing phase (as an optimal manufacturing control problem), the operation phase (again as an optimal control problem), and the disposal phase (an optimal design problem if the system is disposal-critical).

The common denominator in inverse, optimal design, and optimal control problems is a nonlinear optimization problem that is constrained by the PDEs that govern behavior of the physical system. Thus, solving the PDEs is just a subproblem associated with optimization, which can be orders of magnitude more challenging computationally. We refer to the unknown PDE field quantities as the *state variables*; the PDE constraints as the *state equations*; solution of the PDE constraints as the *forward problem* or the *simulation problem*; the inverse, design, or control variables as the *decision variables*; and the problem of determining the optimal values of the inverse, design, or control variables as the *optimization problem*.

In contrast to the large body of work on parallel PDE solution, very little has been pub-

*This work is a part of the Terascale Algorithms for Optimization of Simulations (TAOS) project at CMU, with support from NASA grant NAG-1-2090, NSF grant ECS-9732301 (under the NSF/Sandia Life Cycle Engineering Program), and the Pennsylvania Infrastructure Technology Alliance. Computing services on the Pittsburgh Supercomputing Center's Cray T3E were provided under PSC grant BCS-960001P.

[†]Computational Mechanics Laboratory, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA (biros@cs.cmu.edu, oghattas@cs.cmu.edu).

This article will appear in *Proceedings of SC99*, Portland, OR, November 13-19, 1999.

lished on parallel algorithms for optimization of PDEs (but see [5], [12], [21], [23]). This is expected: it makes little sense to address the inverse problem until one has successfully tackled the forward problem. However, with the recent maturation of parallel PDE solvers for a number of problem classes, the time is ripe to begin focusing on parallel algorithms for large scale PDE-constrained optimization.

Sequential quadratic programming (SQP) methods [6] appear to offer the best hope for smooth optimization of large-scale systems governed by PDEs. SQP methods interleave optimization with simulation, simultaneously improving the design (or control or inversion) while converging the state equations. Thus, unlike popular reduced gradient methods, they avoid complete solution of the state equations at each optimization iteration. Additionally, SQP methods can be made to exploit the structure of the simulation problem, thus building on the advances in parallel PDE solvers over the past 20 years.

The current state-of-the-art for solving PDE-constrained optimization problems is reduced SQP (RSQP) methods. Both mathematical analysis of these methods [4], [7], [13], [17], [20], as well as applications to compressible flow airfoil design [21], [26], [27], heat equation boundary control [18], inverse parameter estimation [8], [16], Navier-Stokes flow control [11], and structural optimization [22], [24], [25], have appeared. In addition, parallel implementations of RSQP methods exhibiting high parallel efficiency and good scalability have been developed [12], [19].

Roughly speaking, RSQP methods project the optimization problem onto the space of decision variables (thereby eliminating the state variables), and then solve the resulting reduced system typically using a quasi-Newton method. The advantage of such an approach is that only two linearized forward problems[1] need to be solved at each iteration (e.g. [11]). For moderate numbers of decision variables, solution of the forward problems dominates an optimization iteration. Thus, when good parallel algorithms are available for the forward problem, RSQP methods inherit the parallel efficiency and scalability (with respect to state variables) of the PDE solvers.

However, the convergence of quasi-Newton methods often deteriorates as the number of decision variables increases. As a result, quasi-Newton-based RSQP methods (QN-RSQP) often exhibit poor *algorithmic* scalability with respect to the decision variables, despite their good parallel efficiency. Furthermore, the requirement of two solutions of the forward problem per optimization iteration can be very onerous when many iterations are taken, even though these solutions are just for the *linearized* forward operator.

The convergence can often be made independent of the number of decision variables m by using a Newton (as opposed to quasi-Newton) RSQP method. However, N-RSQP requires m linearized forward solves *per iteration*. The m linear systems share the same coefficient matrix; their righthand sides are derivatives of the state equation residuals with respect to each decision variable. Iterative solvers are required for the large, sparse, three-dimensional, multicomponent forward problems we target. However, with iterative solvers there is little opportunity to amortize costs over multiple righthand sides. Thus, N-RSQP's need for m forward solves per optimization iteration is unacceptable for large m.

The need for forward solutions results from the decomposition into state and decision spaces (which amounts to range and null spaces of the state equations), and this can be avoided by remaining in the full space of combined state and decision variables. This leaves of course the question of how to solve the resulting "Karush-Kuhn-Tucker" (KKT) full space system. For the large, sparse problems we contemplate, there is no choice but a Krylov method appropriate for symmetric indefinite systems. How to precondition the KKT matrix

[1]Strictly speaking, one involves the adjoint of the forward operator.

within the Krylov solver remains an important challenge, and is crucial for large-scale full space optimization methods to be effective.

In this paper we propose a full space approach that uses a Krylov method to converge the KKT system, but invokes a preconditioner motivated by reduced space ideas. Like QN-RSQP, this approach requires just two linearized forward solves per iteration, but it exhibits the fast convergence associated with Newton methods. Moreover, the two forward solves can be approximate (since they are used within a preconditioner); for example we replace them by an appropriate PDE preconditioner. In addition to building on parallel PDE preconditioning technology, the new KKT preconditioner is based on an exact factorization of the KKT matrix, deflating its spectrum very effectively. Finally, the method parallelizes and scales as well as the forward solver itself. This method is inspired by domain-decomposed Schur complement algorithms. In such techniques, reduction onto the interface space requires exact subdomain solves, so one often prefers to iterate within the full space while using a preconditioner based on approximate subdomain solution [15]. In our case, the decomposition is into states and decisions, as opposed to subdomain and interface spaces.

Battermann and Heinkenschloss have presented a related KKT-system preconditioner that also exploits RSQP methods [3]. However, it is based on congruence transformations of the original system and not on an exact factorization of it. The resulting preconditioned system has both positive and negative eigenvalues and its spectrum is less favorably distributed. Another difference is that our preconditioner includes a quasi-Newton approximation of the reduced Hessian.

Below we describe reduced space SQP and the proposed full space SQP method with two preconditioning variants, and give some performance and scalability results. To evaluate the proposed algorithms, we have chosen a problem that contains many of the features of the most challenging PDE-constrained optimization problems: three-dimensionality, multicomponent coupling, large scale, nonlinearity, and ill-conditioning. The problem is one of optimal control of a viscous incompressible fluid by boundary velocities, a problem of both theoretical and industrial interest. Our implementation is based on the PETSc library for parallel PDE solution [2], and makes use of PETSc parallel domain-decomposition preconditioners for the approximate forward solves. The results in the last section demonstrate that the method is characterized by good parallel efficiency, and is algorithmically efficient provided a scalable forward solver is available.

2. Reduced SQP methods. We begin with a typical constrained optimization problem formulation,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0},$$

where $\boldsymbol{x}$ are the optimization variables, f is the objective function and $\boldsymbol{c}$ are the constraints, which in our context are discretized state equations. We assume that the constraints consist of only state equations.[2] Using the Lagrangian $\mathcal{L}$ one can derive first and higher order optimality conditions. The Lagrangian is defined by

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) := f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{c}(\boldsymbol{x}),$$

and the first order optimality conditions state that the Lagrangian gradient must vanish:[3]

$$\left\{ \begin{array}{c} \boldsymbol{\partial}_x \mathcal{L} \\ \boldsymbol{\partial}_\lambda \mathcal{L} \end{array} \right\} = \left\{ \begin{array}{c} \boldsymbol{\partial}_x f + (\boldsymbol{\partial}_x \boldsymbol{c})^T \boldsymbol{\lambda} \\ \boldsymbol{c} \end{array} \right\} = \boldsymbol{0}.$$

[2] However, the methodology can be extended to problems that include additional inequality constraints.

[3] All vectors and matrices depend on the optimization variables $\boldsymbol{x}$ or the Lagrange multipliers $\boldsymbol{\lambda}$ or both. For clarity, we suppress this dependence.

This expression represents a system of nonlinear equations. SQP (in the absence of inequality constraints) can be viewed as Newton's method for the first order optimality conditions. Customarily, the Jacobian of the residual of this system of equations is called the Karush-Kuhn-Tucker (KKT) matrix of the optimization problem. To simplify the notation further, let us define:

$$\begin{array}{lll} \boldsymbol{A} & := \boldsymbol{\partial}_x \boldsymbol{c} & \text{Jacobian of the constraints,} \\ \boldsymbol{W} & := \boldsymbol{\partial}_{xx} f + \sum_i \boldsymbol{\lambda}_i \boldsymbol{\partial}_{xx} \boldsymbol{c}_i & \text{Hessian of the Lagrangian,} \\ \boldsymbol{g} & := \boldsymbol{\partial}_x f & \text{gradient of the objective.} \end{array}$$

A Newton step on the optimality conditions is give by

$$\begin{bmatrix} \boldsymbol{W} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_\lambda \end{Bmatrix} = - \begin{Bmatrix} \boldsymbol{g} + \boldsymbol{A}^T \boldsymbol{\lambda} \\ \boldsymbol{c} \end{Bmatrix} \quad \text{or} \quad \begin{bmatrix} \boldsymbol{W} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{p}_x \\ \boldsymbol{\lambda}_+ \end{Bmatrix} = - \begin{Bmatrix} \boldsymbol{g} \\ \boldsymbol{c} \end{Bmatrix},$$

where $\boldsymbol{p}_x$ and $\boldsymbol{p}_\lambda$ are the updates of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ from current to next iterations. To exploit the structure of the state constraints, it is useful to introduce a partition of the optimization variables into state variables $\boldsymbol{x}_s$ and decision variables $\boldsymbol{x}_d$. The above KKT system can be partitioned logically as follows:

$$\begin{bmatrix} \boldsymbol{W}_{ss} & \boldsymbol{W}_{sd} & \boldsymbol{A}_s^T \\ \boldsymbol{W}_{ds} & \boldsymbol{W}_{dd} & \boldsymbol{A}_d^T \\ \boldsymbol{A}_s & \boldsymbol{A}_d & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{p}_s \\ \boldsymbol{p}_d \\ \boldsymbol{\lambda}_+ \end{Bmatrix} = - \begin{Bmatrix} \boldsymbol{g}_s \\ \boldsymbol{g}_d \\ \boldsymbol{c} \end{Bmatrix}.$$

The current practice is to avoid solution of the full KKT matrix by reduction to a problem of smaller dimension corresponding to the decision variables. Such *reduced space* methods eliminate the linearized state constraints and variables, and then solve an unconstrained optimization problem in the resulting decision space. RSQP can be derived by a block elimination on the KKT system: Given $\boldsymbol{p}_d$, solve the last block of equations for $\boldsymbol{p}_s$, then solve the first to find $\boldsymbol{\lambda}_+$, and finally solve the middle one for $\boldsymbol{p}_d$. For convenience let us define

$$\boldsymbol{W}_z := \boldsymbol{A}_d^T \boldsymbol{A}_s^{-T} \boldsymbol{W}_{ss} \boldsymbol{A}_s^{-1} \boldsymbol{A}_d - \boldsymbol{A}_d^T \boldsymbol{A}_s^{-T} \boldsymbol{W}_{sd} - \boldsymbol{W}_{ds} \boldsymbol{A}_s^{-1} \boldsymbol{A}_d + \boldsymbol{W}_{dd},$$

which is known as the reduced Hessian of the Lagrangian; $\boldsymbol{B}_z$, its quasi-Newton approximation; and

$$\boldsymbol{g}_z := \boldsymbol{g}_d - \boldsymbol{A}_d^T \boldsymbol{A}_s^{-T} \boldsymbol{g}_s,$$

the reduced gradient of the objective function. The algorithms for N-RSQP and QN-RSQP are:

1. **Initialize**: Choose $\boldsymbol{x}_s, \boldsymbol{x}_d, \boldsymbol{B}_z, \boldsymbol{\lambda}$
2. **Decision step**: solve for $\boldsymbol{p}_d$ from

$$\boldsymbol{W}_z \boldsymbol{p}_d = -\boldsymbol{g}_z + (\boldsymbol{W}_{ds} - \boldsymbol{A}_d^T \boldsymbol{A}_s^{-T} \boldsymbol{W}_{ss}) \boldsymbol{A}_s^{-1} \boldsymbol{c} \quad \text{N-RSQP}$$

$$\boldsymbol{B}_z \boldsymbol{p}_d = -\boldsymbol{g}_z \quad \text{QN-RSQP}$$

3. **State step**: solve for $\boldsymbol{p}_s$ from

$$\boldsymbol{A}_s \boldsymbol{p}_s = -\boldsymbol{A}_d \boldsymbol{p}_d - \boldsymbol{c} \quad \text{N-RSQP, QN-RSQP}$$

4. **Adjoint step**: solve for $\boldsymbol{\lambda}_+$ from

$$\boldsymbol{A}_s^T \boldsymbol{\lambda}_+ = -\boldsymbol{W}_{ss}\boldsymbol{p}_s - \boldsymbol{W}_{sd}\boldsymbol{p}_d - \boldsymbol{g}_s \quad \text{N-RSQP}$$

$$\boldsymbol{A}_s^T \boldsymbol{\lambda}_+ = -\boldsymbol{g}_s \quad \text{QN-RSQP}$$

5. **Update** (in the absence of a line search):

$$\boldsymbol{x}_s = \boldsymbol{x}_s + \boldsymbol{p}_s$$

$$\boldsymbol{x}_d = \boldsymbol{x}_d + \boldsymbol{p}_d$$

$$\text{update } \boldsymbol{B}_z$$

The QN-RSQP method defined here is a variant in which second derivative terms are dropped from the right hand sides of the decision and adjoint steps, at the expense of a reduction from one-step to two-step superlinear convergence [4]. An important advantage of this quasi-Newton method is that only two linearized forward problems need to be solved at each iteration, as opposed to the m needed by N-RSQP for constructing $\boldsymbol{A}_s^{-1}\boldsymbol{A}_d$ in $\boldsymbol{W}_z$ [11]. Furthermore, no second derivatives are needed. The combination of a sufficiently accurate line search and an appropriate quasi-Newton update guarantees a descent search direction and thus a globally convergent algorithm.

QN-RSQP has been very efficiently parallelized for moderate numbers of decision variables [19]. Unfortunately, the number of iterations taken by quasi-Newton methods often increases as the number of decision variables grows,[4] rendering large-scale problems intractable. Additional processors will not help since the bottleneck is in the iteration dimension.

On the other hand, convergence of the N-RSQP method can be independent of the number of decision variables m. However, the necessary m forward solves per iteration preclude its use, particularly on a parallel machine, where iterative methods for the forward problem must be used. These solves can be avoided by remaining in the full space of state and decision variables, since it is the reduction onto the decision space that necessitates the forward solves. Nevertheless, the full space approach also presents difficulties: a descent search direction is not guaranteed, second derivatives are required, and the KKT system itself is very difficult to solve. The size of the KKT matrix is more than twice that of the forward problem, and it is expected to be very ill-conditioned. Ill-conditioning results not only from the forward problem but also from the differing scales between first and second derivatives submatrices. Moreover the system is indefinite; mixing negative and positive eigenvalues is known to slow down Krylov solvers. Therefore a good preconditioner is essential to make the method efficient.

3. Full space SQP with reduced space preconditioner. We propose a Newton full space SQP method (N-FSQP) that uses Krylov iterations to solve the KKT system, but invokes a preconditioner inspired by reduced space quasi-Newton algorithms. We refer to the (nonlinear) Newton iterations as *outer* iterations, and use the term *inner* to refer to the (linear) Krylov iterations for the KKT system. Like QN-RSQP, we require at most several linearized forward solves per (inner) iteration—a large improvement over N-RSQP's m solves. But

[4]E.g. for the limiting quadratic case, the popular BFGS quasi-Newton method is equivalent to conjugate gradients, which scales with the square root of the condition number of the reduced Hessian.

unlike QN-RSQP (and like N-RSQP), the proposed method retains the fast convergence associated with Newton methods. Finally, since the preconditioner amounts to an application of QN-RSQP (albeit with approximate solves), the proposed method retains the structure-exploiting properties of reduced space methods, and should parallelize as well.

To motivate the new preconditioner, let us step back to the reduced Newton method. One can check that N-RSQP described above is equivalent to a particular block LU factorization of the KKT matrix:

$$\begin{bmatrix} \boldsymbol{W}_{ss}\boldsymbol{A}_s^{-1} & 0 & \boldsymbol{I} \\ \boldsymbol{W}_{ds}\boldsymbol{A}_s^{-1} & \boldsymbol{I} & \boldsymbol{A}_d^T\boldsymbol{A}_s^{-T} \\ \boldsymbol{I} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{A}_d & 0 \\ 0 & \boldsymbol{W}_z & 0 \\ 0 & \boldsymbol{W}_{sd} - \boldsymbol{W}_{ss}\boldsymbol{A}_s^{-1}\boldsymbol{A}_d & \boldsymbol{A}_s^T \end{bmatrix}.$$

Note that these factors are permutable to block triangular form. This block factorization suggests a preconditioner created by replacing the reduced Hessian $\boldsymbol{W}_z$ with its quasi-Newton approximation $\boldsymbol{B}_z$. The resulting preconditioned KKT matrix,

$$\begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \boldsymbol{W}_z\boldsymbol{B}_z^{-1} & 0 \\ 0 & 0 & \boldsymbol{I} \end{bmatrix},$$

would be the identity if $\boldsymbol{B}_z$ were equal to $\boldsymbol{W}_z$, and is thus expected to be a good preconditioner provided $\boldsymbol{B}_z$ is a reasonable approximation to the reduced Hessian (as will be the case asymptotically). However, we still require four forward solves per inner iteration. One way to restore the two solves per iteration of QN-RSQP is to, in addition, drop second order information from the preconditioner, exactly as one often does when going from N-RSQP to QN-RSQP. A further simplification of the preconditioner is to replace the exact forward operator $\boldsymbol{A}_s$ by an approximation $\tilde{\boldsymbol{A}}_s$, which could be any appropriate forward problem preconditioner. With these changes, *no* forward solves need to be performed at each inner iteration. Thus, the work per inner iteration becomes linear in the state variable dimension (e.g. when $\tilde{\boldsymbol{A}}_s$ is a constant-fill domain decomposition approximation). Furthermore, when $\boldsymbol{B}_z$ is based on a limited-memory quasi-Newton update (as in our implementation), the work per inner iteration is linear also in the decision variable dimension. Since all of the steps involved in an inner iteration not only require linear work but are also readily parallelized, we conclude that each inner (KKT) iteration will have high parallel efficiency and scalability.

On the other hand, scalability of the entire method additionally requires mesh-independence of both inner and outer iterations. Newton methods (unlike quasi-Newton) are often characterized by a number of iterations that is independent of problem size [1]. With an "optimal" forward preconditioner and a good $\boldsymbol{B}_z$ approximation, we can hope that the number of inner iterations is insensitive to the problem size. This is indeed observed in the next section. Scalability with respect to both state and control variables would then result.

To examine separately the effects of discarding the Hessian terms and approximating the forward solver, we define two different preconditioners:

- **Preconditioner I** takes $\boldsymbol{W}_z = \boldsymbol{B}_z$ and discards all of the other Hessian terms, resulting in two linearized solves per iteration. The preconditioner is

$$\begin{bmatrix} 0 & 0 & \boldsymbol{I} \\ 0 & \boldsymbol{I} & \boldsymbol{A}_d^T\boldsymbol{A}_s^{-T} \\ \boldsymbol{I} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{A}_d & 0 \\ 0 & \boldsymbol{B}_z & 0 \\ 0 & 0 & \boldsymbol{A}_s^T \end{bmatrix},$$

  and the preconditioned KKT matrix is

$$\begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ \boldsymbol{W}_y^T\boldsymbol{A}_s^{-1} & \boldsymbol{W}_z\boldsymbol{B}_z^{-1} & 0 \\ \boldsymbol{W}_{ss}\boldsymbol{A}_s^{-1} & \boldsymbol{W}_y\boldsymbol{B}_z^{-1} & \boldsymbol{I} \end{bmatrix}.$$

Note that the spectrum of the preconditioned KKT matrix is unaffected by dropping the Hessian terms.

- **Preconditioner II** takes $\boldsymbol{W}_z = \boldsymbol{B}_z$ and $\boldsymbol{A}_s = \tilde{\boldsymbol{A}}_s$, and retains all other Hessian terms, resulting in no forward solves. The preconditioner is

$$\begin{bmatrix} \boldsymbol{W}_{ss}\tilde{\boldsymbol{A}}_s^{-1} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{W}_{ds}\tilde{\boldsymbol{A}}_s^{-1} & \boldsymbol{I} & \boldsymbol{A}_d^T\tilde{\boldsymbol{A}}_s^{-T} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{A}}_s & \boldsymbol{A}_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_z & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{W}}_y & \tilde{\boldsymbol{A}}_s^T \end{bmatrix},$$

and the preconditioned KKT matrix is

$$\begin{bmatrix} \boldsymbol{I}_s & \mathcal{O}(\boldsymbol{E}_s) & \boldsymbol{0} \\ \mathcal{O}(\boldsymbol{E}_s) & \mathcal{O}(\boldsymbol{E}_s) + \boldsymbol{W}_z\boldsymbol{B}_z^{-1} & \mathcal{O}(\boldsymbol{E}_s) \\ \mathcal{O}(\boldsymbol{E}_s) & \mathcal{O}(\boldsymbol{E}_s) & \boldsymbol{I}_s^T \end{bmatrix},$$

where $\boldsymbol{E}_s := \boldsymbol{A}_s^{-1} - \tilde{\boldsymbol{A}}_s^{-1}$ and $\boldsymbol{I}_s := \boldsymbol{A}_s\tilde{\boldsymbol{A}}_s^{-1}$. Clearly, $\boldsymbol{E}_s = \boldsymbol{0}$ and $\boldsymbol{I}_s = \boldsymbol{I}$ for exact forward solves.

4. Results on an optimal flow control problem. The optimization method was initially tested on a quadratic optimization problem, that of a 3D interior Stokes flow optimal boundary control problem. Also studied was a general nonlinear optimization problem, that of 3D interior Navier-Stokes flow optimal boundary control. A sequence of Reynolds numbers was solved in order to investigate the effects of nonlinearity.

In both cases, the optimization problem is to minimize the $\boldsymbol{L}^2$ norm of the velocity error given a prescribed velocity field, and the constraints are the equations of viscous incompressible flow with appropriate boundary conditions:

$$\min_{(u,u_d,p)} \frac{1}{2}\int_\Omega (\boldsymbol{u}^* - \boldsymbol{u})^2 \, d\,\Omega$$

$$\begin{aligned} \text{subject to} \quad & -\nu\Delta\boldsymbol{u} + (\nabla\boldsymbol{u})\boldsymbol{u} + \nabla p = \boldsymbol{b} \text{ in } \Omega \\ & \nabla \cdot \boldsymbol{u} = 0 \text{ in } \Omega \\ & \boldsymbol{u} = \boldsymbol{u}^* \text{ on } \Gamma_1 \\ & \boldsymbol{u} = \boldsymbol{u}_d \text{ on } \Gamma_2. \end{aligned}$$

(In the Stokes flow case, the convective term $(\nabla\boldsymbol{u})\boldsymbol{u}$ is absent.) In the example below, $\boldsymbol{u}^*$ is taken as a Poiseuille flow solution in a pipe, and the control variables correspond to boundary velocities on the circumferential surface of the pipe. We discretize by the Galerkin finite element method, using tetrahedral Taylor-Hood elements.

4.1. Stokes flow. The conjugate residual (CR) method is used for solution of the resulting linear flow equations whenever $\boldsymbol{A}_s^{-1}$ is needed. For preconditioning the flow system, we apply a block diagonal matrix

$$\begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P} \end{bmatrix},$$

where $\boldsymbol{V}$ and $\boldsymbol{P}$ are domain decomposition approximations of the discrete Laplacian and discrete mass matrices, respectively. Our code is built on top of the PETSc library [2] and we use PETSc's block-Jacobi preconditioners with local ILU(0) for the domain decomposition approximation.

For comparison, both reduced and full space algorithms for the control problem have been implemented. Because both Preconditioners I and II are indefinite, we use a quasi-minimum residual (QMR) method that supports indefinite preconditioners [10]. Since the Stokes case is a quadratic optimization problem, only one outer iteration is performed, and we set $\boldsymbol{B}_z = \boldsymbol{I}$.

Numerical experiments on a Cray T3E that include scalability and performance assessment and comparisons with reduced SQP have yielded encouraging results. Table 4.1 provides typical results for a problem that grows proportionately in the number of state and control variables and processors utilized. Several findings are notable. We caution that these

TABLE 4.1

Performance of implementations of reduced and full-space SQP methods for a viscous flow optimal boundary control problem, as a function of increasing number of state and control variables and number of processors. Here, **QN-RSQP** *is quasi-Newton reduced-space SQP;* **N-FSQP** *is Newton full-space SQP;* **precond** *is the KKT preconditioner type;* **I** *is the two-exact-solves preconditioner and in* **II** *the exact solves have been replaced by approximate solves;* $\| \boldsymbol{g}_z \|$ *is the Euclidean norm of the reduced gradient and is a measure of optimality; and* **time** *is wall-clock time in hours on the Pittsburgh Supercomputing Center's Cray T3E-900. To prevent long execution times for QN-RSQP, the algorithm was terminated prematurely for the first four cases at 200 iterations, and for the last at 100 iterations. Similarly, unpreconditioned N-FSQP was terminated at 500,000 KKT iterations for the two largest problems. In contrast, preconditioned N-FSQP was completely converged to a reduced gradient norm of* 10^{-5} *in all cases. For this reason, the true relative performance of the two preconditioners is better than depicted in the table.*

states controls	**method**	**precond**	**N or QN iter**	**KKT iter**	$\| \boldsymbol{g}_z \|$	**time**
21,000	QN-RSQP	—	200	—	1×10^{-4}	3.6
3,900	N-FSQP	none	1	114,000	9×10^{-6}	2.5
(4 PEs)	N-FSQP	I	1	25	9×10^{-6}	1.2
	N-FSQP	II	1	3,200	9×10^{-6}	0.3
43,000	QN-RSQP	—	200	—	4×10^{-4}	8.1
4,800	N-FSQP	none	1	198,000	9×10^{-6}	4.4
(8 PEs)	N-FSQP	I	1	29	9×10^{-6}	1.7
	N-FSQP	II	1	5,500	9×10^{-6}	0.7
86,000	QN-RSQP	—	200	—	2×10^{-3}	12.8
6,400	N-FSQP	none	1	376,000	9×10^{-6}	9.0
(16 PEs)	N-FSQP	I	1	28	9×10^{-6}	2.4
	N-FSQP	II	1	8,200	9×10^{-6}	1.0
167,000	QN-RSQP	—	200	—	3×10^{-3}	18.4
12,700	N-FSQP	none	1	500,000	8×10^{-5}	12.3
(32 PEs)	N-FSQP	I	1	27	9×10^{-6}	2.7
	N-FSQP	II	1	11,100	9×10^{-6}	1.3
332,000	QN-RSQP	—	100	—	9×10^{-3}	11.0
23,500	N-FSQP	none	1	500,000	4×10^{-4}	13.0
(64 PEs)	N-FSQP	I	1	28	9×10^{-6}	3.1
	N-FSQP	II	1	14,900	9×10^{-6}	1.7

conclusions are based on a limited set of solved problems; on the other hand, we have no reason to believe that other problems should behave very differently. The first observation is that QN-RSQP has to be terminated prematurely to avoid excessive consumption of Cray time. This is particularly true for the largest-sized problem (332,000 flow and 23,500 control variables), which required 11 hours of execution on 64 processors to reach a reduced gradient of 10^{-2}. Problems of industrial interest involve the full Navier-Stokes equations and may be an order of magnitude larger; it is impossible to escape the conclusion that problems of this complexity are intractable with quasi-Newton optimization methods. For this problem, N-FSQP (with Preconditioner II) requires a factor of 6.5 less time to reach three orders of magnitude smaller reduced gradient.

Another observation is that Preconditioner I, despite its discarding of second order terms and the use of the identity to approximate the reduced Hessian, is very effective in reducing the number of iterations—note the difference in KKT iterations between the second (unpreconditioned) and third (preconditioned) lines of each problem instance. Of course, this reduction comes at the price of extra work per iteration, mainly in the solution of the forward problems.

A third conclusion is reached by tracking the execution time for preconditioned N-FSQP as the size of the problem and number of processors increase proportionately. Since problem size per processor is held constant, and execution time increases from 1.2/0.3 hours for 4 processors (21,000 states, 3900 controls) to 3.1/1.7 hours for 64 processors (332,000 states, 23,500 controls), one may conclude that the method is not scalable. However, a glance at the KKT iterations column reveals that the number of optimization iterations when using Preconditioner I is largely independent of problem size, and thus the algorithmic efficiency of N-FSQP should be very high.[5] Furthermore, the Mflop rate drops (probably due to a poorer-quality mesh partition) only slowly as the problem size increases, suggesting good implementation efficiency. What, then, accounts for the increase in execution time?

Table 4.2 provides the answer. Following [14], overall parallel efficiency has been decomposed into an implementation efficiency and an algorithmic efficiency for both the optimization algorithm and the forward solver. Whereas the optimization algorithm and the implementation are reasonably scalable, the forward solver's parallel efficiency drops to near 50% for the largest problem size. This is due to the increasing number of iterations taken by the forward solver as the problem size increases. The parallel inefficiency of the forward solver is easily addressed by switching to a more scalable preconditioner than the one we are currently using (non-overlapping local ILU(0) block-Jacobi), and by relaxing the need to solve the forward problem exactly at each KKT iteration (as we do in Preconditioner II). Indeed the latter can yield significant improvement in the execution time of the algorithm. One can observe a factor of 2–4 improvement over Preconditioner I. However, the iterations

TABLE 4.2

Isogranular scalability results for the N-FSQP algorithm with Preconditioner I. Per processor Mflop rates are maximum (across PEs) sustained Mflop/s. Implementation efficiency (**impl eff**) *is based on Mflop rate; optimization algorithmic efficiency* (**opt eff**) *is based on number of optimization iterations; forward solver algorithmic efficiency* (**forw eff**) *is deduced; overall efficiency* (**overall eff**) *is based on execution time, and is product of all three.*

PEs	Mflop/s/PE	Mflop/s	impl eff	opt eff	forw eff	overall eff
4	41.5	163	1.00	1.00	1.00	1.00
8	39.7	308	0.95	0.86	0.84	0.68
16	38.8	603	0.92	0.89	0.62	0.51
32	37.2	1130	0.87	0.93	0.55	0.45
64	36.8	2212	0.85	0.89	0.52	0.39

[5]This was better than expected for the quadratic case since the reduced Hessian is simply taken as the identity and therefore no preconditioning takes place in the design space.

for the KKT system do not scale as well as with Preconditioner I, but this is expected and again associated with the effectiveness of the forward problem preconditioner. With a better forward preconditioner, we anticipate good overall scalability. We hope to remedy this in future experiments by allowing overlap and subdomain fill-in in the forward preconditioner (which we avoided for memory reasons in the present tests).

4.2. Navier-Stokes flow. The same problem as in Section 4.1 is solved, but with the Navier-Stokes equations as constraints. Poiseuille flow is still a solution, theoretically for any Reynolds number, practically for moderate ones. A notable difference with the Stokes case is that a BFGS quasi-Newton update [9] is used to precondition the reduced Hessian matrix. Because the problem is nonlinear, N-FSQP takes several iterations and quasi-Newton curvature information is built up. Quasi-Newton theory predicts that $\boldsymbol{B}_z$ approaches $\boldsymbol{W}_z$ as the iterates get closer to the solution. Therefore, one expects the effectiveness of the preconditioner to improve as the optimization algorithm progresses.

To solve a Navier-Stokes control problem with large Reynolds number, some kind of continuation scheme is usually needed. In our case we use a simple three-level iteration. In the outer iteration the Reynolds number is gradually increased until the target number is reached. Converged values of variables are used to initiate the iterations at the next Reynolds number. Additionally, quasi-Newton information is carried forward to the next Reynolds number. The middle iterations correspond to Newton linearizations of the optimality system for a fixed Reynolds number. Finally, the inner iterations comprise solution of the KKT linear system to compute the search direction.

In the numerical experiments of this section, we do not make use of continuation to initiate the state and control variables. The reason is that our initial guess is always inside the basin of attraction of the solution. However, BFGS information is carried forward to the next Reynolds number, in both QN-RSQP and N-FSQP methods. To approximate the reduced Hessian (for preconditioning the KKT system in the Newton method and for driving the iterations in the quasi-Newton method), we invoke a limited-memory BFGS formula in which the inverse of $\boldsymbol{B}_z$ is updated. In this way, only a limited number of vectors must be retained and both the update and application of $\boldsymbol{B}_z^{-1}$ involve only vector inner products, which parallelize well (this is important for the large number of control variables considered). In the results presented below, the maximum number of BFGS vectors is 30. For simplicity we do not perform a line search, again since in the cases run we are within the basin of attraction of a minimum. The forward problem preconditioner differs from the Stokes case: we found block-Jacobi to be very slow and instead opted for an overlapping additive Schwarz preconditioner with ILU(1) on each subdomain. Finally, since the forward problem is unsymmetric, general QMR is used for the linearized Navier-Stokes solves. Results for a problem with 21,000 state and 3,900 design variables on 4 processors and for a sequence of three Reynolds numbers are presented in Table 4.3. Overall, one can observe that N-FSQP reduces significantly the execution time relative to QN-RSQP, just as in the Stokes problem.

The Navier-Stokes test problem provides an opportunity to improve the reduced Hessian preconditioner by building up quasi-Newton curvature information, something not possible for the Stokes flow problem, since a single KKT system is solved in that case. Although the test problems were not sufficiently nonlinear to require a large number of iterations, even with three Newton iterations we can already observe increasing effectiveness of the quasi-Newton approximation. This can be seen with Preconditioner I as the number of KKT iterations drops with increasing continuation parameter (i.e. Reynolds number).

As in the Stokes control problem, Preconditioner II is faster than Preconditioner I, since it avoids exact forward solves. The number of KKT iterations slightly increases with increasing Reynolds number, since the forward problem condition number deteriorates (recall that

TABLE 4.3

*Algorithmic efficiency of the proposed preconditioners for a model Navier-Stokes optimal flow control problem. Recall that Preconditioner I requires two linearized forward solves per iteration, whereas Preconditioner II involves just application of the Schwarz approximation. The number of iterations for the KKT system is averaged across the optimization iterations. The problem has 21,000 state equations and 3,900 control variables; results are for 4 processors of the T3E-900. The Reynolds numbers (**Re**) tabulated also correspond to the continuation steps.*

Re	method	precond	N/QN iter	KKT iter	$\| \mathbf{g}_z \|$	time
100	QN-RSQP	—	262	—	1×10^{-4}	5.9
	N-FSQP	none	3	186,000	9×10^{-6}	7.1
	N-FSQP	I	3	48	9×10^{-6}	3.2
	N-FSQP	II	3	4200	9×10^{-6}	1.3
300	QN-RSQP	—	278	—	1×10^{-4}	6.4
	N-FSQP	none	3	198,000	9×10^{-6}	7.6
	N-FSQP	I	3	40	9×10^{-6}	3.1
	N-FSQP	II	3	4300	9×10^{-6}	1.4
500	QN-RSQP	—	289	—	1×10^{-4}	7.3
	N-FSQP	none	3	213,000	9×10^{-6}	9.0
	N-FSQP	I	3	38	9×10^{-6}	3.0
	N-FSQP	II	3	4410	9×10^{-6}	1.4

because of the inexact solves, the characteristics of the forward problem now affect the KKT system solution). Indicative of the difficulty of this problem is that despite its small size, its solution still took more than an hour on four processors.

Unlike a quasi-Newton method that can insure a descent direction by a sufficiently accurate line search, there is no guarantee of descent and hence global convergence for the present Newton method. However, in the experiments we conducted, we always converged to the global minimum, due most likely to a combination of weak nonlinearity and good initial guesses. No doubt more general problems may not be so kind. We are currently investigating remedies. A simple one is to revert to a quasi-Newton search direction when an uphill direction is encountered; this is easy to do, since reduced space quasi-Newton information is available for free from the preconditioner.

5. Conclusions. The problems we have chosen to investigate are relatively simple, yet provide a reasonable testbed for algorithmic tuning and experimentation. The results obtained thus far are very encouraging: the full space Newton-Krylov optimization method with reduced-space preconditioning is a factor of 5–10 faster than current reduced space methods. Moreover, the method can be parallelized efficiently, and algorithmic efficiency can be achieved provided a good forward preconditioner is available. Scalability then results from the combination of these two.

In future work, we will pursue better forward preconditioners; attempt to reproduce the current results on more realistic and demanding problems; address robustness issues stemming from a possible lack of global convergence for more nonlinear problems; investigate inexact solution of the KKT system; and examine matrix-free variants in an attempt to obviate the potentially onerous need for second derivatives.

Acknowledgments. We thank Satish Balay, Bill Gropp, Lois McInnes, and Barry Smith of Argonne National Lab for their work in making PETSc available to the research community. We also thank Jonathan Shewchuk of UC Berkeley for providing the meshing and partitioning routines Pyramid and Slice. Finally, we thank David Keyes of ICASE/Old Dominion University, David Young of Boeing, and the other members of the TAOS project—Roscoe

Bartlett, Larry Biegler, Greg Itle, and Ivan Malčević—for their useful comments.

REFERENCES

[1] E. L. ALLGOWER, K. BÖHMER, F.-A. POTRA, AND W. C. RHEINBOLDT, *A mesh-independence principle for operator equations and their discretizations*, SIAM Journal on Numerical Analysis, 23 (1986), pp. 160–169.

[2] S. BALAY, W. D. GROPP, L. C. MCINNES, AND B. F. SMITH, *PETSc Home Page*, 1999. `http://www.mcs.anl.gov/petsc`.

[3] A. BATTERMANN AND M. HEINKENSCHLOSS, *Preconditioners for Karush-Kuhn-Tucker matrices arising in the optimal control of distributed systems*, in Optimal Control of Partial Differential Equations, W. Desch, F. Kappel, and K. Kunisch, eds., vol. 126 of International Series of Numerical Mathematics, Birkhäuser Verlag, 1998, pp. 15–32.

[4] L. T. BIEGLER, J. NOCEDAL, AND C. SCHMID, *A reduced Hessian method for large-scale constrained optimization*, SIAM Journal on Optimization, 5 (1995), pp. 314–347.

[5] L. T. BIEGLER AND I. B. TJOA, *A parallel implementation for data regression with implicit models*, Annals of Operations Research, 42 (1993), p. 1.

[6] P. BOGGS AND J. TOLLE, *Sequential quadratic programming*, in Acta Numerica 1995, Cambridge, 1995, pp. 1–51.

[7] R. H. BYRD AND J. NOCEDAL, *An analysis of reduced Hessian methods for constrained optimization*, Mathematical Programming, 49 (1991), pp. 285–323.

[8] J. E. DENNIS AND R. M. LEWIS, *A comparison of nonlinear programming approaches to an elliptic inverse problem and a new domain decomposition approach*, Tech. Rep. TR94-33, Department of Computational and Applied Mathematics, Rice University, 1994.

[9] R. FLETCHER, *Practical Methods of Optimization*, Wiley-Interscience, 1987.

[10] R. W. FREUND AND N. M. NACHTIGAL, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM Journal of Scientific Computing, 15 (1994), pp. 313–337.

[11] O. GHATTAS AND J.-H. BARK, *Optimal control of two- and three-dimensional incompressible Navier–Stokes flows*, Journal of Computational Physics, 136 (1997), pp. 231–244.

[12] O. GHATTAS AND C. E. OROZCO, *A parallel reduced Hessian SQP method for shape optimization*, in Multidisciplinary Design Optimization: State-of-the-Art, N. Alexandrov and M. Hussaini, eds., SIAM, 1997, pp. 133–152.

[13] M. HEINKENSCHLOSS, *Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet boundary control of Navier-Stokes flow*, Tech. Rep. TR97-14, Department of Computational and Applied Mathematics, Rice University, May 1997.

[14] D. E. KEYES, *How scalable is domain decomposition in practice?*, in Proceedings of the 11th International Conference on Domain Decomposition Methods, C.-H. Lai, P. Bjorstad, M. Cross, and O. Widlund, eds., 1998.

[15] D. E. KEYES AND W. D. GROPP, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. S166–S202.

[16] K. KUNISCH AND E. W. SACHS, *Reduced SQP methods for parameter identification problems*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 1793–1820.

[17] F.-S. KUPFER, *An infinite-dimensional convergence theory for reduced SQP methods in Hilbert space*, SIAM Journal on Optimization, 6 (1996), pp. 126–163.

[18] F.-S. KUPFER AND E. W. SACHS, *Numerical solution of a nonlinear parabolic control problem by a reduced SQP method*, Computational Optimization and Applications, 1 (1992), pp. 113–135.

[19] I. MALČEVIĆ, *Large-scale unstructured mesh shape optimization on parallel computers*, Master's thesis, Carnegie Mellon University, 1997.

[20] J. NOCEDAL AND M. OVERTON, *Projected Hessian updating algorithms for nonlinearly constrained optimization*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 821–850.

[21] C. OROZCO AND O. GHATTAS, *Massively parallel aerodynamic shape optimization*, Computing Systems in Engineering, 1–4 (1992), pp. 311–320.

[22] C. E. OROZCO AND O. GHATTAS, *A reduced SAND method for optimal design of nonlinear structures*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 2759–2774.

[23] J. REUTHER, J. J. ALONSO, M. J. RIMLINGER, AND A. JAMESON, *Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on parallel computers*, Tech. Rep. 96.17, RIACS, NASA Ames Research Center, Sept. 1996.

[24] U. RINGERTZ, *Optimal design of nonlinear shell structures*, Tech. Rep. FFA TN 91-18, The Aeronautical Research Institute of Sweden, 1991.

[25] ———, *An algorithm for optimization of nonlinear shell structures*, International Journal for Numerical Meth-

ods in Engineering, 38 (1995), pp. 299–314.

[26] V. H. SCHULZ AND H. G. BOCK, *Partially reduced SQP methods for large-scale nonlinear optimization problems*, in Proceedings of the Second World Congress of Nonlinear Analysis, Elsevier Verlag, 1997.

[27] A. R. SHENOY, M. HEINKENSCHLOSS, AND E. M. CLIFF, *Airfoil design by an all-at-once method*, International Journal for Computational Fluid Mechanics, 11 (1998), pp. 3–25.