

Parallel Preconditioners for KKT Systems Arising in Optimal Control of Viscous Incompressible Flows*

G. Biros^a and O. Ghattas^a

^aComputational Mechanics Laboratory
Carnegie Mellon University, Pittsburgh PA, 15213, USA
biros@cs.cmu.edu, oghattas@cs.cmu.edu

1. INTRODUCTION

Recently, interest has increased in model-based optimal flow control of viscous fluids, that is, the determination of optimal values of parameters for systems governed by the fluid dynamics equations. For example, the objective could be minimizing drag on a solid body, and the controls might consist of velocities or tractions on some part of the boundary or of the shape of the boundary itself. Such problems are among the most computationally challenging optimization problems. The complexity stems from their being constrained by numerical approximations of the fluid equations, commonly the Navier-Stokes or the Euler equations. These constraints are highly nonlinear and can number in the millions for typical systems of industrial interest.

The current state-of-the-art for solving such flow-constrained optimization problems is reduced sequential quadratic programming (RSQP) methods. General mathematical analysis of these methods [3,9] as well as CFD-related research [5,7] have appeared. In addition, parallel implementations of RSQP methods exhibiting high parallel efficiency and good scalability have been developed [6,10]. These methods essentially project the optimization problem onto the space of control variables (thereby eliminating the flow variables), and then solve the resulting reduced system using a quasi-Newton method. The advantage of such an approach is that only two linearized flow problems need to be solved at each iteration. However, the convergence of quasi-Newton based RSQP methods (QN-RSQP) deteriorates as the number of control variables increases, rendering large-scale problems intractable.

The convergence can often be made independent of the number of control 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 right-hand sides are derivatives of the state equations with respect to each control variable. LU factorization of the linear system would be ideal here, but is

*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 were provided under grant number BCS-960001P from the Pittsburgh Supercomputing Center, which is supported by several federal agencies, the Commonwealth of Pennsylvania and private industry.

not viable for the large, sparse, three-dimensional, multicomponent forward problems we target. Instead, iterative solvers must be used, and 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 control spaces (range and null spaces of the state equations), and this can be avoided by remaining in the full space of combined state and control 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 contemplated, there is no choice but a Krylov method appropriate for symmetric indefinite systems. How to best precondition the KKT matrix within the Krylov solver remains an important challenge, and is crucial for the viability of large-scale full space optimization methods.

In this paper we propose a preconditioner for the KKT system based on a reduced space quasi-Newton algorithm. Battermann and Heinkenschloss [2] have also suggested a preconditioner that is motivated by reduced methods; the one we present can be thought of as a generalization of their method. As in reduced quasi-Newton algorithms, the new preconditioner requires just two linearized flow solves per iteration, but permits the fast convergence associated with full Newton methods. Furthermore, the two flow solves can be approximate, for example using any appropriate flow preconditioner. Finally, the resulting full space SQP parallelizes and scales as well as the flow solver itself. Our method is inspired by the domain-decomposed Schur complement algorithms. In these 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 [8]. Here, decomposition is performed into states and controls, as opposed to subdomain and interface spaces.

Below we describe reduced and full space SQP methods and the proposed reduced space-based KKT preconditioner. We also give some performance results on a Cray T3E for a model Stokes flow problem. Our implementation is based on the PETSc library for PDE solution [1], and makes use of PETSc domain-decomposition preconditioners for the approximate flow solves.

2. REDUCED SQP METHODS

We begin with a typical discretized constrained optimization problem,

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{1}$$

where \mathbf{x} are the optimization variables, f is the objective function and \mathbf{c} are the constraints, which in our context are discretized flow equations. Using the Lagrangian \mathcal{L} , one can derive first and higher order optimality conditions. The Lagrangian is defined by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{2}$$

and the first order optimality conditions are¹

$$\left\{ \begin{array}{l} \boldsymbol{\partial}_x \mathcal{L} \\ \boldsymbol{\partial}_\lambda \mathcal{L} \end{array} \right\} = \left\{ \begin{array}{l} \boldsymbol{\partial}_x f + (\boldsymbol{\partial}_x \mathbf{c})^T \boldsymbol{\lambda} \\ \mathbf{c} \end{array} \right\} = \mathbf{0}. \tag{3}$$

¹All vectors and matrices depend on the optimization variables \mathbf{x} or the Lagrange multipliers $\boldsymbol{\lambda}$ or both. For clarity, we suppress this dependence.

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

$$\begin{aligned} \mathbf{A} &:= \boldsymbol{\partial}_x \mathbf{c} && \text{Jacobian of the constraints,} \\ \mathbf{W} &:= \boldsymbol{\partial}_{xx} f + \sum_i \lambda_i \boldsymbol{\partial}_{xx} \mathbf{c}_i && \text{Hessian of the Lagrangian,} \\ \mathbf{g} &:= \boldsymbol{\partial}_x f && \text{Gradient of the objective function.} \end{aligned} \quad (4)$$

A Newton step on the optimality conditions (3) is given by

$$\begin{bmatrix} \mathbf{W} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_x \\ \mathbf{p}_\lambda \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} \\ \mathbf{c} \end{Bmatrix} \quad \text{or} \quad \begin{bmatrix} \mathbf{W} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_x \\ \boldsymbol{\lambda}_+ \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g} \\ \mathbf{c} \end{Bmatrix}, \quad (5)$$

where \mathbf{p}_x and \mathbf{p}_λ are the updates in \mathbf{x} and $\boldsymbol{\lambda}$ from current to next iterations and $\boldsymbol{\lambda}_+$ is the updated Lagrange multiplier. To exploit structure of the flow constraints, it is useful to induce a partition of the optimization variables into state \mathbf{x}_s and control (or decision) variables \mathbf{x}_d . The above KKT system can be partitioned logically as follows:

$$\begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} & \mathbf{A}_s^T \\ \mathbf{W}_{ds} & \mathbf{W}_{dd} & \mathbf{A}_d^T \\ \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_s \\ \mathbf{p}_d \\ \boldsymbol{\lambda}_+ \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g}_s \\ \mathbf{g}_d \\ \mathbf{c} \end{Bmatrix}. \quad (6)$$

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

$$\mathbf{W}_z := \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{ss} \mathbf{A}_s^{-1} \mathbf{A}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{sd} - \mathbf{W}_{ds} \mathbf{A}_s^{-1} \mathbf{A}_d + \mathbf{W}_{dd}, \quad (7)$$

the reduced Hessian of the Lagrangian; \mathbf{B}_z , its quasi-Newton approximation; and

$$\mathbf{g}_z := \mathbf{g}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{g}_s, \quad (8)$$

the reduced gradient of the objective function. The resulting algorithms for Newton (N-RSQP) and quasi-Newton (QN-RSQP) variants of RSQP are:

1. **Initialize:** Choose $\mathbf{x}_s, \mathbf{x}_d$
2. **Control step:** solve for \mathbf{p}_d from

$$\begin{aligned} \mathbf{W}_z \mathbf{p}_d &= -\mathbf{g}_z + (\mathbf{W}_{ds} - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{ss}) \mathbf{A}_s^{-1} \mathbf{c} && \text{N-RSQP} \\ \mathbf{B}_z \mathbf{p}_d &= -\mathbf{g}_z && \text{QN-RSQP} \end{aligned} \quad (9)$$

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

$$\mathbf{A}_s \mathbf{p}_s = -\mathbf{A}_d \mathbf{p}_d - \mathbf{c} \quad (10)$$

4. **Adjoint step:** solve for λ_+ from

$$\begin{aligned} \mathbf{A}_s^T \lambda_+ &= -\mathbf{W}_{ss} \mathbf{p}_s - \mathbf{W}_{sd} \mathbf{p}_d - \mathbf{g}_s && \text{N-RSQP} \\ \mathbf{A}_s^T \lambda_+ &= -\mathbf{g}_s && \text{QN-RSQP} \end{aligned} \quad (11)$$

5. **Update:**

$$\begin{aligned} \mathbf{x}_s &= \mathbf{x}_s + \mathbf{p}_s \\ \mathbf{x}_d &= \mathbf{x}_d + \mathbf{p}_d. \end{aligned} \quad (12)$$

The quasi-Newton method defined here is a variant in which second order terms are dropped from the right-hand sides of the control and adjoint steps, at the expense of a reduction from one-step to two-step superlinear convergence [3]. An important advantage of this quasi-Newton method is that only two linearized flow problems need to be solved at each iteration, as opposed to the m needed by Newton's method in constructing $\mathbf{A}_s^{-1} \mathbf{A}_d$ as can be seen in (7). Furthermore, no second derivatives are necessary, since a quasi-Newton approximation is made to the reduced Hessian. Finally, it can be shown that this method parallelizes very efficiently [10]. Unfortunately, the number of iterations required for a quasi-Newton method to converge increases with the number of control variables, rendering large-scale problems intractable. Additional processors will not help since the bottleneck is in the iteration dimension.

In Newton's method, the convergence is often independent of the number of control variables m . However, the necessary m flow solves per iteration preclude its use, particularly on a parallel machine, for which iterative methods will be used for the forward solves. These solves can be avoided by remaining in the full space of flow and control variables, since it is the reduction onto the control space that necessitates the flow solves. Nevertheless, this also presents difficulties: exploitation of structure is much more difficult with the KKT matrix, which is over twice the size, highly indefinite, and contains scattered blocks, each having mesh-based sparsity structure.

3. FULL SPACE SQP WITH REDUCED SPACE PRECONDITIONER

In this section we present a new preconditioner for KKT systems arising in full space SQP, based on reduced space quasi-Newton algorithms. The preconditioner recovers the small (constant) number of linearized flow solves per iteration of the quasi-Newton method, but permits the fast convergence associated with Newton methods. The preconditioner retains the structure-exploiting properties of RSQP, and parallelizes as well.

To motivate the derivation, let us return to the reduced Newton method. As stated before, N-RSQP is equivalent to solving with a permuted block-LU factorization of the KKT matrix; this factorization can be also viewed as a Schur-complement reduction for the control space step \mathbf{p}_d . The unpermuted form is given by

$$\begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} & \mathbf{A}_s^T \\ \mathbf{W}_{ds} & \mathbf{W}_{dd} & \mathbf{A}_d^T \\ \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{ss} \mathbf{A}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds} \mathbf{A}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T \mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_y & \mathbf{A}_s^T \end{bmatrix}, \quad (13)$$

where $\mathbf{W}_y := \mathbf{W}_{sd} - \mathbf{W}_{ss}\mathbf{A}_s^{-1}\mathbf{A}_d$. This block factorization suggests its use as a preconditioner by replacing \mathbf{W}_z with \mathbf{B}_z . The resulting preconditioned KKT matrix,

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (14)$$

would be the identity if \mathbf{B}_z were equal to \mathbf{W}_z .

There are two main differences between this preconditioner and the one suggested in [2]. The first is that this preconditioner is based on an exact factorization of the KKT matrix, i.e. it is *indefinite*. The second is that it incorporates BFGS as a sub-preconditioner to deflate the reduced Hessian. The preconditioned KKT matrix is positive definite, with $2n$ unit eigenvalues and the remaining m determined by the effectiveness of BFGS. However, we still require four forward solves per iteration. One way to restore the two solves per iteration of QN-RSQP is to 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 \mathbf{A}_s by an approximation $\tilde{\mathbf{A}}_s$, which could be for example any appropriate flow preconditioner. With these changes, *no* forward solves need to be performed at each KKT iteration. Thus, the work per KKT iteration becomes linear in the state variable dimension (e.g. when $\tilde{\mathbf{A}}_s$ is a constant-fill domain decomposition approximation). Furthermore, when \mathbf{B}_z is based on a limited-memory quasi-Newton update (as in our implementation), the work per KKT iteration is also linear in the control variable dimension. With an ‘‘optimal’’ forward preconditioner and the assumption that the \mathbf{B}_z approximation is a good one, one would expect the number of KKT (inner) iterations to be insensitive to the problem size. Mesh-independence of SQP (outer) iterations would then lead to scalability with respect to both state and control variables. To examine the effects of discarding the Hessian terms and approximating the forward solver, we define two different preconditioners:

- Preconditioner I: $\mathbf{W}_z = \mathbf{B}_z$, all of the Hessian terms in (13) discarded
2 (linearized) solves/iteration

$$\begin{array}{cc} \text{Preconditioner} & \text{Preconditioned KKT matrix} \\ \left[\begin{array}{ccc} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_d^T\mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{array} \right] \left[\begin{array}{ccc} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_s^T \end{array} \right], & \left[\begin{array}{ccc} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_y^T\mathbf{A}_s^{-1} & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{W}_{ss}\mathbf{A}_s^{-1} & \mathbf{W}_y\mathbf{B}_z^{-1} & \mathbf{I} \end{array} \right] \end{array} \quad (15)$$

- Preconditioner II: $\mathbf{W}_z = \mathbf{B}_z$, $\mathbf{A}_s = \tilde{\mathbf{A}}_s$, Hessian terms retained in (13)
no forward solves, $\mathbf{E}_s := (\mathbf{A}_s^{-1} - \tilde{\mathbf{A}}_s^{-1})$, $\mathbf{I}_s := \mathbf{A}_s\tilde{\mathbf{A}}_s^{-1}$

$$\begin{array}{cc} \text{Preconditioner} & \text{Preconditioned KKT matrix} \\ \left[\begin{array}{ccc} \mathbf{W}_{ss}\tilde{\mathbf{A}}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds}\tilde{\mathbf{A}}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T\tilde{\mathbf{A}}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{array} \right] \left[\begin{array}{ccc} \tilde{\mathbf{A}}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{W}}_y & \tilde{\mathbf{A}}_s^T \end{array} \right], & \left[\begin{array}{ccc} \mathbf{I}_s & \mathcal{O}(\mathbf{E}_s) & \mathbf{0} \\ \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) + \mathbf{W}_z\mathbf{B}_z^{-1} & \mathcal{O}(\mathbf{E}_s) \\ \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) & \mathbf{I}_s^T \end{array} \right]. \end{array} \quad (16)$$

4. RESULTS ON AN OPTIMAL FLOW CONTROL PROBLEM

The preconditioner was tested on a model quadratic programming problem (QP), that of a 3D interior Stokes flow boundary control problem. The objective is to minimize the L^2 norm of the velocity error given a prescribed velocity field, and the constraints are the Stokes equations:

$$\begin{aligned}
 & \text{minimize } \frac{1}{2} \int_{\Omega} (\mathbf{u}_{\text{exact}} - \mathbf{u})^2 d\Omega \\
 & \text{subject to:} \\
 & -\nu \Delta \mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega \\
 & \nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega \\
 & \mathbf{u} = \mathbf{u}_{\text{exact}}, \quad \text{on } \Gamma / \Gamma_{\text{control}} \\
 & \mathbf{u} = \mathbf{u}_d, \quad \text{on } \Gamma_{\text{control}}.
 \end{aligned} \tag{17}$$

Here, $\mathbf{u}_{\text{exact}}$ 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. The minimum residual method (MINRES) is used for solving the resulting linear flow equations whenever \mathbf{A}_s^{-1} is needed. To precondition the flow system, we apply a two-block diagonal matrix where the two blocks are domain decomposition approximations of the discrete Laplacian and discrete mass matrices, respectively. Our code is built on top of the PETSc library [1] and the domain decomposition approximations we use are PETSc's block-Jacobi preconditioners with local ILU(0).

Both reduced and full space algorithms for the control problem have been implemented. Since the preconditioner we propose is indefinite, we use a quasi minimum residual (QMR) method that supports indefinite preconditioners [4]. The reduced Hessian is approximated by a limited-memory BFGS formula² in which we update the inverse of \mathbf{B}_z . In this way, only a limited number of vectors need to be retained and both the update and the application of \mathbf{B}_z^{-1} involve only vector inner products.

Numerical experiments that include scalability and performance assessment on a Cray T3E and comparisons with RSQP have yielded very encouraging results. A comparison between the different methods is presented in Table 1. A simple scalability analysis, depicted in Table 2, shows very good efficiency with respect to the optimization algorithm and the parallel implementation. On the other hand, the algorithmic efficiency of the forward preconditioner is not very good; we intend to remedy this with a more sophisticated preconditioner for \mathbf{A}_s . We are extending the implementation to encompass Navier-Stokes flows. Since the problem is no longer a QP, issues of robustness and global convergence become crucial. We believe that a hybrid SQP method that combines QN-RSQP far from the minimum and full space N-FSQP (preconditioned by RSQP) close to the minimum will prove to be robust and powerful. The order-of-magnitude improvement in execution time and high parallel efficiency observed for the Stokes flow control problem encourage further development and application of the new KKT preconditioner.

²This is done only in QN-RSQP. For a QP problem, full Newton takes one iteration to converge so N-FSQP cannot create curvature information for \mathbf{B}_z . We have set $\mathbf{B}_z = \mathbf{I}$ in the KKT preconditioner.

Table 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, *precond* is the KKT preconditioner; *N or QN iter* is the number of optimization iterations; $\|g_z\|$ is the Euclidean norm of the reduced gradient; and *time* is wall-clock hours on the Pittsburgh Supercomputing Center’s Cray T3E-900. To prevent long execution times for QN-RSQP, the algorithm was terminated for the first four cases at 200 iterations, and for the last at 100 iterations. Similarly, N-FSQP was terminated at 500,000 KKT iterations for the two largest problems. In contrast, both preconditioned N-FSQP methods were allowed to completely converge to a reduced gradient norm of 10^{-6} in all cases. For this reason, the true performance of preconditioned N-FSQP is better than depicted in the table. Even with the more stringent tolerance, the new preconditioner improves wall-clock time by a factor of 10 over QN-RSQP.

<i>states</i>	<i>controls</i>	<i>method</i>	<i>precond</i>	<i>N or QN iter</i>	<i>KKT iter</i>	$\ g_z\ $	<i>time</i>
21,000 (4 PEs)	3900	QN-RSQP	—	200	—	1×10^{-4}	3.6
		N-FSQP	none	1	114,000	9×10^{-6}	2.5
		N-FSQP	I	1	25	9×10^{-6}	1.2
		N-FSQP	II	1	3,200	9×10^{-6}	0.3
43,000 (8 PEs)	4800	QN-RSQP	—	200	—	4×10^{-4}	8.1
		N-FSQP	none	1	198,000	9×10^{-6}	4.4
		N-FSQP	I	1	29	9×10^{-6}	1.7
		N-FSQP	II	1	5,500	9×10^{-6}	0.7
86,000 (16 PEs)	6400	QN-RSQP	—	200	—	2×10^{-3}	12.8
		N-FSQP	none	1	376,000	9×10^{-6}	9.0
		N-FSQP	I	1	28	9×10^{-6}	2.4
		N-FSQP	II	1	8,200	9×10^{-6}	1.0
167,000 (32 PEs)	12,700	QN-RSQP	—	200	—	3×10^{-3}	18.4
		N-FSQP	none	1	500,000	8×10^{-5}	12.3
		N-FSQP	I	1	27	9×10^{-6}	2.7
		N-FSQP	II	1	11,100	9×10^{-6}	1.3
332,000 (64 PEs)	23,500	QN-RSQP	—	100	—	9×10^{-3}	11.0
		N-FSQP	none	1	500,000	4×10^{-4}	13.0
		N-FSQP	I	1	28	9×10^{-6}	3.1
		N-FSQP	II	1	14,900	9×10^{-6}	1.7

Table 2

Isogranular scalability results for N-FSQP with Preconditioner I. Per processor Mflop rates are average (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.

<i>PEs</i>	<i>Mflop/s/PE</i>	<i>Mflop/s</i>	<i>impl eff</i>	<i>opt eff</i>	<i>forw eff</i>	<i>overall eff</i>
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

ACKNOWLEDGMENTS

We thank the authors of PETSc, Satish Balay, Bill Gropp, Lois McInnes, and Barry Smith of Argonne National Lab. We also 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. S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. PETSc home page. <http://www.mcs.anl.gov/petsc>, 1999.
2. A. Battermann and M. Heinkenschloss. Preconditioners for Karush-Kuhn-Tucker matrices arising in the optimal control of distributed systems. In W. Desch, F. Kappel, and K. Kunisch, editors, *Optimal control of partial differential equations*, volume 126 of *International Series of Numerical Mathematics*, pages 15–32. Birkhäuser Verlag, 1998.
3. L. T. Biegler, J. Nocedal, and C. Schmid. A reduced Hessian method for large-scale constrained optimization. *SIAM Journal on Optimization*, 5:314–347, 1995.
4. 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(2):313–337, March 1994.
5. O. Ghattas and J.-H. Bark. Optimal control of two- and three-dimensional incompressible Navier-Stokes flows. *Journal of Computational Physics*, 136:231–244, 1997.
6. O. Ghattas and C. E. Orozco. A parallel reduced Hessian SQP method for shape optimization. In N. Alexandrov and M. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 133–152. SIAM, 1997.
7. M. Heinkenschloss. Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet boundary control of Navier-Stokes flow. In W. W. Hager and P. M. Pardalos, editors, *Optimal Control: Theory, Algorithms, and Applications*, pages 178–203. Kluwer Academic Publishers B.V., 1998.
8. 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(2):S166–S202, March 1987.
9. K. Kunisch and E. W. Sachs. Reduced SQP methods for parameter identification problems. *SIAM Journal on Numerical Analysis*, 29(6):1793–1820, December 1992.
10. I. Malčević. Large-scale unstructured mesh shape optimization on parallel computers. Master’s thesis, Carnegie Mellon University, 1997.