SIAG/OPT Views-and-News

A Forum for the SIAM Activity Group on Optimization

Volume 11 Number 2 August 2000

A Lagrange-Newton-Krylov-Schur Method for PDE-Constrained Optimization

George Biros and Omar Ghattas
Mechanics, Algorithms, and Computing Laboratory
Department of Civil & Environmental Engineering
Carnegie Mellon University, Pittsburgh, PA, USA
Email: {biros,oghattas}@cs.cmu.edu
URL: http://www.cs.cmu.edu/~{gbiros,oghattas}

1. Introduction

PDE-constrained optimization is a frontier problem in computational science and engineering. All PDE-constrained problems share the difficulty that PDE solution is just a subproblem associated with optimization. Thus, the optimization problem is often significantly more difficult to solve than the simulation problem. We are particularly interested in large-scale problems that require parallel computing to make them tractable.

To illustrate the main issues, let's consider a model problem of optimal distributed control of a Navier-Stokes flow:

$$\min \mathcal{F}(\boldsymbol{u}, p, \boldsymbol{b}) = \frac{1}{2}\int_\Omega (\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) : (\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) d\Omega + \frac{\alpha}{2}\int_\Omega \boldsymbol{b}\cdot\boldsymbol{b}\, d\Omega$$

subject to:

$$\begin{aligned} -\nu\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) + (\boldsymbol{\nabla u})\boldsymbol{u} + \boldsymbol{\nabla} p + \boldsymbol{b} &= \boldsymbol{0} \quad \text{in} \quad \Omega \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 \quad \text{in} \quad \Omega \\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on} \quad \Gamma \end{aligned}$$

Here, $\boldsymbol{u}$ is the fluid velocity field, p the pressure field, $\boldsymbol{b}$ the body force control function, α a weighting parameter, and ν the inverse of the Reynolds number. The objective is to minimize the rate of dissipation of viscous energy and a cost associated with a body force control function. The constraints are the stationary incompressible Navier-Stokes equations with Dirichlet boundary conditions.

We can form a Lagrangian functional, and require its stationarity with respect to the state ($\boldsymbol{u}$,p) and optimization ($\boldsymbol{b}$) variables and the Lagrange multipliers. Taking variations and invoking the appropriate Green identities, we arrive at the following first-order necessary conditions:

Adjoint Equations:

$$\begin{aligned} -\nu\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla\lambda}_u + \boldsymbol{\nabla\lambda}_u^T) + (\boldsymbol{\nabla u})^T\boldsymbol{\lambda}_u - (\boldsymbol{\nabla\lambda}_u)\boldsymbol{u} \\ +\boldsymbol{\nabla}\lambda_p - \boldsymbol{\nabla}\cdot(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) &= \boldsymbol{0} \quad \text{in} \quad \Omega \\ \boldsymbol{\nabla}\cdot\boldsymbol{\lambda}_u &= \boldsymbol{0} \quad \text{in} \quad \Omega \\ \boldsymbol{\lambda}_u &= \boldsymbol{0} \quad \text{on} \quad \Gamma \end{aligned}$$

State Equations:

$$\begin{aligned} -\nu\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T) + (\boldsymbol{\nabla u})\boldsymbol{u} + \boldsymbol{\nabla} p + \boldsymbol{b} &= \boldsymbol{0} \quad \text{in} \quad \Omega \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= \boldsymbol{0} \quad \text{in} \quad \Omega \\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on} \quad \Gamma \end{aligned}$$

Control Equations:

$$\rho\,\boldsymbol{b} + \boldsymbol{\lambda}_u = \boldsymbol{0} \quad \text{in} \quad \Omega$$

The state equations are just the original Navier-Stokes PDEs. The *adjoint equations*, which result from stationarity with respect to state variables, are themselves PDEs, and are linear in the Lagrange multipliers $\boldsymbol{\lambda}_u$ and λ_p. Finally, the *control equations* are (in this case) algebraic.

Thus we end up with a large, coupled, unstructured system of optimality conditions (or at least bigger, more coupled, and less structured than seen by a Navier-Stokes solvers). How to go about solving it? The usual way is to eliminate state variables and Lagrange multipliers and, correspondingly, the state equations and adjoint equations; to reduce the system to a manageable one in just the control (i.e. decision) variables $\boldsymbol{b}$. Here's one way to do this: given $\boldsymbol{b}$ at some iteration, we solve the state equations for the state variables $\boldsymbol{u}, p$. Knowing the state variables then permits us to solve the adjoint equations for the Lagrange multipliers $\boldsymbol{\lambda}_u, \lambda_p$. Finally, with the states and multipliers known, we can update $\boldsymbol{b}$ by iterating on the control equation. The whole process is repeated until

convergence. This elimination procedure is termed a *reduced space* method, in contrast to a *full space* method, in which one solves for the states, controls, and multipliers simultaneously.

Reduced space methods are attractive for several reasons. Solving the subsets of equations in sequence imparts some structure to the problem. State equation solvers build on years of development of large-scale parallel PDE solvers. Adjoint PDE solvers don't exactly grow on trees—but the strong similarities between the state and adjoint operators suggest that an existing PDE solver for the state equations can be modified easily to handle the adjoint system (at least on a good day). Finally, the control equations are usually reasonably tame, at least to evaluate. Another advantage of reduction is that the full space system is often very ill-conditioned, whereas the three subsystems are typically better conditioned.

On the other hand, the big disadvantage of reduced methods is the need to solve the state and adjoint equations *at each iteration* of the reduced system—a direct consequence of the reduction onto the decision variable space. So it's natural to go back to the full space, and ask if it's possible to solve the entire optimality system simultaneously, but retain the structure-inducing, condition-improving advantages of reduced space methods—while avoiding their disadvantages.

In this article, we present such a method. The key idea is to solve in the full space using a Newton method, but precondition with a quasi-Newton reduced space method. The Karush-Kuhn-Tucker system arising at each Newton iteration is solved using a Krylov iterative method, and it is this system to which the preconditioner is applied. We have found that the reduced space preconditioner is very effective in reducing the number of Krylov iterations, and applying it captures the favorable structure of reduced methods. On the other hand, since the reduction is used just as a preconditioner, we can cheat on the state and adjoint solves, replacing them with approximations which could be their own preconditioners. So we arrive at a method that combines rapid convergence in the outer Newton iteration (typically mesh-independent), with fast convergence of the inner Krylov iteration (which can be as good as mesh-independent). We don't even need to compute second derivatives—since a Krylov method is used to solve the KKT system, we can apply the usual directional differencing trick to approximate the Lagrangian Hessian–vector product.

Why the name *Lagrange-Newton-Krylov-Schur*? It is common in PDE-solver circles to use the phrase *Newton-Krylov-X* to refer to Newton methods for solving PDEs that employ Krylov linear solvers, with *X* as the preconditioner for the Krylov method. Since *Lagrange-Newton* is sometimes used to describe a Newton method for solving the optimality system (a.k.a. an SQP method), and since a reduced space method can be viewed as a Schur complement method for the KKT system, we arrive at the concatenation *LNKS*. It's a mouthful, but it preserves the tie to modern PDE solvers, whose use of approximate decompositions as preconditioners inspired this approach [6]. David Keyes suggested (a variation of) this name in his plenary talk at the 1999 combined SIAM Optimization/Annual meeting [5].

In the remainder of this article, we give a brief overview of the LNKS method and some sample results for an optimal flow control problem on a Cray T3E. Further details can be found in [2], and more extensive discussion and results in forthcoming articles that focus on the inner Krylov iteration [3] and the outer Newton iteration [4]. We note finally that Battermann and Heinkenschloss have presented a somewhat different method for preconditioning KKT matrices that also makes use of state and control space decompositions [1].

2. Reduced Space Methods

In this section we discuss reduced space SQP methods, concentrating on the discrete form of a typical PDE-constrained optimization problem:

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

where $\boldsymbol{x}$ are the state and decision variables, f is the objective function and $\boldsymbol{c}$ are the discretized state equations. Using Lagrange multipliers $\boldsymbol{\lambda}$, we can define the Lagrangian function by

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

The first order optimality conditions require that the Lagrangian gradient vanish:

$$\begin{Bmatrix} \boldsymbol{\partial}_x \mathcal{L} \\ \boldsymbol{\partial}_\lambda \mathcal{L} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{g} + \boldsymbol{A}^T \boldsymbol{\lambda} \\ \boldsymbol{c} \end{Bmatrix} = \boldsymbol{0},$$

where $\boldsymbol{g}$ is the gradient of the f and $\boldsymbol{A}$ is the Jacobian matrix of the constraints. A Newton step on the optimality conditions (which, in the absence of inequality constraints, is Sequential Quadratic Programming) is given

by:

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

where $\boldsymbol{W}$ is the Hessian of the Lagrangian function with respect to the optimization variables, $\boldsymbol{p}_x$ is the search direction in $\boldsymbol{x}$, and $\boldsymbol{\lambda}_+$ is the updated Lagrange multiplier. This system is known as the Karush-Kuhn-Tucker (KKT) system, and its coefficient matrix as the KKT matrix. To exploit the structure of the state constraints, we partition the optimization variables into state variables $\boldsymbol{x}_s$ and decision variables $\boldsymbol{x}_d$. The partitioned KKT system becomes:

$$\begin{bmatrix} \boldsymbol{W}_{ss} & \boldsymbol{W}_{sd} & \boldsymbol{A}_s^T \\ \boldsymbol{W}_{ds} & \boldsymbol{W}_{dd} & \boldsymbol{A}_d^T \\ \boldsymbol{A}_s & \boldsymbol{A}_d & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{p}_s \\ \boldsymbol{p}_d \\ \boldsymbol{\lambda}_+ \end{Bmatrix} = - \begin{Bmatrix} \boldsymbol{g}_s \\ \boldsymbol{g}_d \\ \boldsymbol{c} \end{Bmatrix} \quad (0.1)$$

This system is of dimension $2n+m$, where n is the number of state variables and m the number of decision variables. State-of-the-art algorithms for PDE-constrained optimization exploit two facts. First, nobody wants to compute second derivatives—it's hard enough convincing the PDE solver community of the need for first derivatives. (No doubt this difficulty will be mitigated by continuing advances in automatic differentiation tools.) And second, everybody wants to use existing software for "inverting" the state Jacobian. Since this is the kernel step in a Newton-based PDE solver, there is a large body of work to draw from. For example, for elliptic PDEs, there exist optimal or nearly-optimal parallel algorithms (e.g. domain decomposition methods or multigrid) that require algorithmic work that is linear or weakly superlinear in n, and scale to thousands of processors and millions of variables.

One way to exploit existing PDE-solvers is to eliminate the state and adjoint equations and variables, and then solve an unconstrained optimization problem in the remaining decision space (this is similar to the argument of the previous section, except here we are linearizing first, then eliminating, as opposed to vice versa.) We refer to this as *Newton reduced SQP* (or N-RSQP), and it can be derived by block elimination on the KKT system: Given $\boldsymbol{p}_d$, solve the last block of equations (the state system) for $\boldsymbol{p}_s$; then solve the first (the adjoint system) to find $\boldsymbol{\lambda}_+$, and finally solve the middle (the decision system) for $\boldsymbol{p}_d$. It is easy to verify that this block elimination is equivalent to the following block factorization of the KKT matrix:

$$\begin{bmatrix} \boldsymbol{W}_{ss}\boldsymbol{A}_s^{-1} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{W}_{ds}\boldsymbol{A}_s^{-1} & \boldsymbol{I} & \boldsymbol{A}_d^T\boldsymbol{A}_s^{-T} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{A}_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_z & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{yz} & \boldsymbol{A}_s^T \end{bmatrix} \quad (0.2)$$

where the reduced Hessian matrix is defined by

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

and the "cross-Hessian" by

$$\boldsymbol{W}_{yz} := \boldsymbol{W}_{sd} - \boldsymbol{W}_{ss}\boldsymbol{A}_s^{-1}\boldsymbol{A}_d.$$

Note that these factors can be permuted to block triangular form, so we can think of this as a block *LU* factorization of the KKT matrix. It is clear that the only linear systems that need to be solved have either the state Jacobian $\boldsymbol{A}_s$ or its transpose as their coefficient matrix—a "solved problem"—or else the reduced Hessian $\boldsymbol{W}_z$, which is dense and of dimension of the decision space. Thus, reduced methods are particularly attractive when the the decision variables are much fewer than the states.

But two problems remain. First are the second derivative terms. Second, and more problematic, is the need for m solutions of the (linearized) state equations for construction of $\boldsymbol{A}_s^{-1}\boldsymbol{A}_d$ in $\boldsymbol{W}_z$. This is particularly troublesome for large-scale 3D problems, where (linearized) PDE systems are usually solved iteratively, and solution costs cannot be amortized over multiple right hands as effectively as with direct solvers. When the simulation problem is an overnight run on a large parallel machine, this requirement effectively rules out the use of N-RSQP.

A popular technique that addresses these two difficulties is a quasi-Newton RSQP (QN-RSQP) method that replaces the reduced Hessian $\boldsymbol{W}_z$ with a quasi-Newton approximation $\boldsymbol{B}_z$, and discards all other Hessian terms. This corresponds to the following approximation of the KKT block factors:

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{A}_d^T\boldsymbol{A}_s^{-T} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_s & \boldsymbol{A}_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_z & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_s^T \end{bmatrix} \quad (0.3)$$

It is easy to verify that just two state solves per iteration are required (actually one linearized state, and one adjoint), as opposed to the m of N-RSQP. And with Hessian terms either approximated or dropped, no second derivatives are needed. A measure of the success

of QN-RSQP is its application to numerous optimal control, optimal design, and inverse problems governed by PDEs from linear and nonlinear elasticity, incompressible and compressible flow, heat conduction and convection, phase changes, flow through porous media, etc. Of course, something has to give, and that is the convergence rate: a reduction from quadratic in the Newton case to two-step superlinear. Moreover, the number of iterations taken by QN-RSQP depends on the conditioning of the reduced Hessian, and often increases as the number of decision variables grows, rendering large-scale problems intractable. In the next section, we propose a method that combines the fast convergence of Newton's method with the structure-exploiting properties of reduced methods.

3. LNKS: Krylov solution of the KKT system with approximate QN-RSQP preconditioning

In this section, we present a method for solving the KKT system (0.1). For optimization problems constrained by 3D PDEs, sparse factorization of the KKT matrix is not an option—such methods are not viable for $\boldsymbol{A}_s$, let alone the entire matrix. Instead, we use a Krylov iterative method, specifically the quasi-minimum residual (QMR) method. However, the varying scales between Hessian and Jacobian terms in the KKT matrix, and its indefiniteness, demand an effective preconditioner. This preconditioner must be capable of exploiting the structure of the state constraints (specifically that good preconditioners exist for $\boldsymbol{A}_s$), must be cheap to apply, and must be effective in reducing the number of Krylov iterations. The QN-RSQP method described in the previous section fits the bill. Applying the preconditioner amounts to solving with the QN-RSQP factorization (0.3), except that state Jacobians are replaced by their approximations $\tilde{\boldsymbol{A}}_s$:

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{A}_d^T \tilde{\boldsymbol{A}}_s^{-T} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{A}}_s & \boldsymbol{A}_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_z & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \tilde{\boldsymbol{A}}_s^{T} \end{bmatrix} \tag{0.4}$$

Replacing $\boldsymbol{A}_s$ with $\tilde{\boldsymbol{A}}_s$ is permissible, since QN-RSQP is being used as a preconditioner. A good choice for $\tilde{\boldsymbol{A}}_s$ is, in turn, one of the available preconditioners for $\boldsymbol{A}_s$—for many PDE operators, there exist near-spectrally-equivalent preconditioners that are both cheap to apply (typically linear or weakly superlinear in problem size) and effective (resulting in iteration numbers that are independent of, or increase very slowly in, problem size).

With (0.4) used as a preconditioner, the preconditioned KKT matrix ends up having the following form:

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

where

$$\begin{aligned} \boldsymbol{E}_s &:= \boldsymbol{A}_s^{-1} - \tilde{\boldsymbol{A}}_s^{-1} \\ \boldsymbol{I}_s &:= \boldsymbol{A}_s \tilde{\boldsymbol{A}}_s^{-1} \\ \tilde{\boldsymbol{W}}_{yz} &:= \boldsymbol{W}_{sd} - \boldsymbol{W}_{ss} \tilde{\boldsymbol{A}}_s^{-1} \boldsymbol{A}_d \end{aligned}$$

For exact state equation solution, $\boldsymbol{E}_s = \boldsymbol{0}$ and $\boldsymbol{I}_s = \boldsymbol{I}$, and we see that the QN-RSQP preconditioner clusters the spectrum of the KKT matrix, with all eigenvalues either unit or belonging to $\boldsymbol{W}_z \boldsymbol{B}_z^{-1}$. Therefore, when $\tilde{\boldsymbol{A}}_s$ is a good preconditioner for the state Jacobian, and when $\boldsymbol{B}_z$ is a good approximation of the reduced Hessian (as it should be asymptotically), we might expect the QN-RSQP preconditioner (0.4) to be effective in reducing the number of Krylov iterations (but note that the preconditioned KKT matrix is non-normal, so a rigorous analysis requires well-conditioned eigenvectors).

How scalable is the method, with respect to increasing problem size and number of processors? For scalability, we require that the work increase near-linearly with problem size (algorithmic scalability) and that it parallelizes well (parallel scalability). Let us examine the major components:

Formation of the KKT matrix–vector product. For PDE-constrained optimization, the Hessian of the Lagrangian function and the Jacobian of the constraints are usually sparse with structure dictated by the mesh (particularly when the decision variables are mesh-related). Thus, formation of the matrix-vector product at each QMR iteration is linear in both state and decision variables, and parallelizes well due to a high computation-to-communication ratio and minimal sequential bottlenecks.

Application of the QN-RSQP preconditioner. The main work involved is application of the state Jacobian preconditioner $\tilde{\boldsymbol{A}}_s$ and its transpose, and "inversion" of the quasi-Newton approximation to the reduced Hessian, $\boldsymbol{B}_z$. We can often make use of scalable, parallel state Jacobian preconditioners that requires $\mathcal{O}(n)$ work to apply (as in various domain decomposition preconditioners for elliptic problems). Furthermore, when $\boldsymbol{B}_z$ is based on a

limited-memory quasi-Newton update (as in our implementation), its work is also linear in the decision variables, and the vector operations are easily parallelized (or as easily as vector inner products can be). Therefore, we conclude that application of the QN-RSQP preconditioner requires linear work and parallelizes well.

The Krylov (inner) iteration. As argued above, with an "optimal" state preconditioner and a good $\boldsymbol{B}_z$ approximation, we can anticipate that the number of inner, Krylov iterations will be relatively insensitive to the problem size.

The Lagrange-Newton (outer) iteration. The number of outer, Newton iterations is often independent of problem size for PDE-type problems, and the problems we have solved exhibit this type of behavior as well.

This combination of linear work per Krylov iteration, weak dependence of Krylov iterations on problem size, and independence of Lagrange-Newton iterations on problem size suggest a method that scales well with increasing problem size and number of processors.

How well does the LNKS method work in practice? Here, we quote a set of representative results from many we have obtained for up to 1.5 million state variables and 50,000 control variables on up to 256 processors. The problem is optimal Navier-Stokes flow control, similar to that of Section 1, except that the controls are boundary velocities. The specific problem is control of 3D flow around a cylinder at subcritical conditions, with controls on the downstream side of the cylinder. Approximation is by Galerkin finite elements, both for state and control variables. We have implemented the LNKS method on top of the PETSc library for parallel solution of PDEs from Argonne. The table shows results for 64 and 128 processors of a Cray T3E for a roughly doubling of problem size. Results for the QN-RSQP and LNKS algorithms are presented. In the table LNKS-EX refers to exact solution of the linearized Navier-Stokes equation within the QN-RSQP preconditioner, whereas LNKS-PR refers to application of a block-Jacobi (with local ILU(0)) approximation of the linearized Navier-Stokes operator. LNKS-PR-TR uses a truncated Newton method and avoids fully converging the KKT system for iterates that are far from a solution.

The results in the table reflect the independence of Newton iterations on problem size, the mild dependence of KKT iterations on problem size, and the resulting reasonable scalability of the method. It is important to point out here that the Navier-Stokes discrete operator is very ill-conditioned, and there is room for improvement of its domain-decomposition preconditioner. The performance of the QN-RSQP KKT preconditioner would improve correspondingly. A dramatic acceleration of the LNKS algorithm is achieved by truncating the Krylov iterations.

states controls	**preconditioning**	**Newton iter**	**average KKT iter**	**time (hours)**
389,440	QN-RSQP	189	—	46.3
6,549	LNKS-EX	6	19	27.4
(64 procs)	LNKS-PR	6	2,153	15.7
	LNKS-PR-TR	13	238	3.8
615,981	QN-RSQP	204	—	53.1
8,901	LNKS-EX	7	20	33.8
(128 procs)	LNKS-PR	6	3,583	16.8
	LNKS-PR-TR	12	379	4.1

More detailed results are given in [2, 3, 4]. These references also discuss the important topics of globalization and the details of the inexactness in solving the KKT system, which were not mentioned here for reasons of space. Another issue is additional inequality constraints; we have recently implemented with Andreas Wächter and Larry Biegler a parallel version of their interior point method for treating such constraints, within the context of LNKS. Finally, this summer we will be releasing a publicly-available software library for parallel solution of PDE-constrained optimization problems, built on top of the PETSc system, and including LNKS and other methods.

Acknowledgments

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 and NSF grant ECS-9732301 (under the NSF/Sandia Life Cycle Engineering Program). Computing services on the Pittsburgh Supercomputing Center's Cray T3E were provided under PSC grant ASC-990003P. 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 Old Dominion University/ICASE/Lawrence Livermore, David Young of Boeing, and other members of the TAOS project—Roscoe Bartlett, Larry Biegler, Ivan Malčević, Andreas Wächter—for their useful comments.

REFERENCES

[1] A. BATTERMANN AND M. HEINKENSCHLOSS, *Preconditioners for Karush–Kuhn–Tucker matrices arising in optimal control of distributed systems*, Tech. Rep. TR96-34, Department of Computational and Applied Mathematics, Rice University, November 1996.

[2] G. BIROS AND O. GHATTAS, *Parallel Newton-Krylov algorithms for PDE–constrained optimization*, in Proceedings of SC99, Portland, Oregon, 1999.

[3] ———, *Lagrange-Newton-Krylov-Schur algorithms for PDE-constrained optimization. Part I: KKT preconditioners*, 2000. In preparation.

[4] ———, *Lagrange-Newton-Krylov-Schur algorithms for PDE-constrained optimization. Part II: Truncated Newton solver*, 2000. In preparation.

[5] D. E. KEYES, *Krylov, Lagrange, N]ewton, and Schwarz: Combinations and Permutations, note = Plenary talk to joint session of the 1999 SIAM Annual Meeting and 1999 SIAM Conference on Optimization, address = Atlanta*.

[6] 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.