Inexactness Issues in the Lagrange-Newton-Krylov-Schur Method for PDE-constrained Optimization

George Biros[1] and Omar Ghattas[2]

[1] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA (biros@cs.nyu.edu).
[2] Laboratory for Mechanics, Algorithms, and Computing, Departments of Biomedical Engineering and Civil & Environmental Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA (oghattas@cs.cmu.edu).

Abstract. We present an overview of the Lagrange-Newton-Krylov-Schur (LNKS) method for solution of optimization problems that are governed by systems of partial differential equations. We discuss how to improve LNKS's work efficiency by carrying out certain computations inexactly, without compromising convergence. LNKS solves the Karush-Kuhn-Tucker optimality conditions using a Newton-Krylov algorithm. Its key component is a preconditioner based on variants of quasi-Newton reduced space Sequential Quadratic Programming (QN-RSQP) methods. LNKS combines the fast convergence properties of a Newton method with the ability of preconditioned Krylov methods to solve very large linear systems. Nevertheless, even with good preconditioners, solution of the optimization problem is several times more expensive than solution of the underlying PDE problem. To accelerate LNKS, its computational components are carried out inexactly: premature termination of iterative algorithms, inexact evaluation of gradients and Jacobians, and approximate line searches. We discuss several issues that arise with respect to these inexact computations, and the resulting trade-offs between speed and robustness.

1 Introduction

We discuss algorithmic and implementation aspects of the Lagrange-Newton-Krylov-Schur method for solution of large-scale nonlinearly-constrained optimization problems. The proposed techniques have applications to a broad category of optimization problems, including optimal control, optimal design, and parameter identification for systems governed by partial differential equations (PDEs). These are often known as *PDE-constrained optimization* problems.

In this article we will 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*; 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*.

Perhaps the most popular technique for large-scale nonlinearly-constrained optimization is the reduced space Sequential Quadratic Programming (RSQP) method [2], [17]. In its elaboration for PDE-constrained optimization, the linearized state equations and state variables are eliminated at each iteration, and a quadratic optimization problem is solved in the reduced space of decision variables. LNKS, which was introduced in [3,6,7], employs a family of methods that use Newton-Krylov algorithms to solve the Karush-Kuhn-Tucker optimality conditions in the full space of state and decision variables, and invokes a preconditioner motivated by reduced space ideas. We refer to the (nonlinear) Newton iterations as *outer* iterations, and use the term *inner* to denote the (linear) Krylov iterations for the Karush-Kuhn-Tucker (KKT) system that arises at each Newton iteration.

The inner iterative solver and the preconditioner that accelerate the computations of a Newton step for the KKT optimality conditions are essential ingredients of LNKS. We introduced and analyzed several variants of the preconditioner in [4,6]. The parallelizability and scalability of the LNKS algorithm were also examined in those papers. The basic form of the outer Newton solver is analyzed in [7].

Like adjoint-based quasi-Newton RSQP methods (see e.g. [13]), this approach requires just two linearized forward solves per iteration, and in addition exhibits the fast convergence associated with Newton methods. Moreover, the two forward solves can be approximate (since they are used within the LNKS preconditioner); for example we replace them by an appropriate PDE preconditioner. LNKS builds on existing parallel PDE preconditioners and generally parallelizes and scales as well as the forward solver itself. In this paper, we extend our earlier work with a discussion of the inexact components of the LNKS method—in particular the inexact solves within the QN-RSQP steps used for globalization.

The paper is organized as follows. In Section 2 we overview the LNKS method. In Section 3 we discuss globalization approaches for the outer iteration and briefly present line-search based QN-RSQP methods. Section 4 discusses several inexact computations within LNKS. We conclude in Section 5 with numerical results from the application of the method to the optimal boundary control of the steady incompressible Navier-Stokes equations.

Some notational conventions follow. We use boldface characters to denote vector-valued functions and vector-valued function spaces. We use Roman characters to denote discretized quantities and italics for their continuous counterparts. For example $\boldsymbol{u}$ will be the continuous velocity field and $\mathbf{u}$ will be its discretization. Greek letters are overloaded and whether we refer to the discretization or the continuous fields should be clear from context. We also use $(+)$ as a subscript or superscript to denote variable updates within an iterative algorithm. Finally, quantities with a tilde on the top indicate that they are results of inexact computations.

2 LNKS Method

Let us consider the constrained optimization problem,

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

where $\mathbf{x} \in \mathbb{R}^N$ are the optimization variables, $f : \mathbb{R}^N \to \mathbb{R}$ is the objective function and $\mathbf{c} : \mathbb{R}^N \to \mathbb{R}^n$ are the constraints, which we assume consist of only the discretized state equations.

In order to exploit the structure of the problem we partition $\mathbf{x}$ (which lies in the full space of optimization variables) into state variables $\mathbf{x}_s \in \mathbb{R}^n$, and decision variables $\mathbf{x}_d \in \mathbb{R}^m$,

$$\mathbf{x} = \begin{Bmatrix} \mathbf{x}_s \\ \mathbf{x}_d \end{Bmatrix}, \tag{2}$$

where $m + n = N$. The Lagrangian,

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{3}$$

is used to convert the constrained optimization problem into a system of nonlinear equations. For convenience we introduce the following notation:

$$\begin{array}{lll} \mathbf{A}(\mathbf{x}) := \partial_x \mathbf{c}(\mathbf{x}) & \in \mathbb{R}^{n \times N} & \text{Jacobian matrix of the constraints,} \\ \mathbf{W}(\mathbf{x}, \boldsymbol{\lambda}) := \partial_{xx} f(\mathbf{x}) + \sum_i \lambda_i \partial_{xx} \mathbf{c}_i(\mathbf{x}) & \in \mathbb{R}^{N \times N} & \text{Hessian matrix of the Lagrangian,} \\ \mathbf{g}(\mathbf{x}) := \partial_x f(\mathbf{x}) & \in \mathbb{R}^N & \text{gradient vector of the objective.} \end{array}$$

The first order optimality conditions state that at a local minimum the gradient of the Lagrangian must vanish:

$$\begin{Bmatrix} \partial_x \mathcal{L} \\ \partial_\lambda \mathcal{L} \end{Bmatrix} (\mathbf{x}, \boldsymbol{\lambda}) = \begin{Bmatrix} \mathbf{g}(\mathbf{x}) + \mathbf{A}(\mathbf{x})^T \boldsymbol{\lambda} \\ \mathbf{c}(\mathbf{x}) \end{Bmatrix} = \mathbf{0} \quad (\text{or } \mathbf{h}(\mathbf{q}) = \mathbf{0}). \tag{4}$$

Customarily, these equations are called the Karush-Kuhn-Tucker or KKT optimality conditions, and points at which the gradient of the Lagrangian vanishes are called *KKT points*.

A Newton step on the optimality conditions is given by

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

where $\mathbf{p}_x$ and $\mathbf{p}_\lambda$ are used to update $\mathbf{x}$ and $\boldsymbol{\lambda}$ from current to next iterations. The KKT optimality conditions (4) define a system of nonlinear equations. The Jacobian $\mathbf{K}$ of this system is termed the *KKT matrix*. Assuming sufficient smoothness, and that the initial guess is sufficiently close to a solution, updates obtained by (5) will converge quadratically to the solution [11]. Thus, the forward solves required for reduced methods 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 [13]. Nevertheless, the full space approach also presents difficulties: a descent 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 typically very ill-conditioned. Ill-conditioning results not only from the forward problem but also from the different scales between first and second derivative submatrices. Moreover, the system is indefinite; mixing negative and positive eigenvalues is known to slow down Krylov solvers. Therefore, a good preconditioner is essential for making the method efficient.

In LNKS we use a Newton method to solve the KKT optimality conditions. To compute the Newton step we solve the KKT system using an appropriate Krylov method. At the core of the algorithm lies the preconditioner $\mathbf{P}$ for the Krylov method, which is an inexact version of the QN-RSQP algorithm. An outline of the basic LNKS method is given by Algorithm 1.

Algorithm 1

```
1: Choose $\mathbf{x}, \boldsymbol{\lambda}$
2: **loop**
3:   Check for convergence
4:   Compute $\mathbf{c}, \mathbf{g}, \mathbf{A}, \mathbf{W}$
5:   Solve $\mathbf{P}^{-1}\mathbf{K}\mathbf{p} = \mathbf{P}^{-1}\mathbf{h}$                    (Newton Step)
6:   Update $\mathbf{x} = \mathbf{x} + \mathbf{p}_x$
7:   Update $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \mathbf{p}_\lambda$
8: **end loop**
```

To derive the preconditioner we rewrite the KKT system (5) in a block-partitioned form:

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

RSQP is equivalent to a block-row elimination: given $\mathbf{p}_d$, solve the last block of equations for $\mathbf{p}_s$, then solve the first to find $\mathbf{p}_\lambda$, and finally solve the middle one for the decision variable update $\mathbf{p}_d$. Therefore RSQP can be written as a particular block-LU factorization of the KKT matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W}_{ss}\mathbf{A}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds}\mathbf{A}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T\mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{sd} - \mathbf{W}_{ss}\mathbf{A}_s^{-1}\mathbf{A}_d & \mathbf{A}_s^T \end{bmatrix}. \tag{7}$$

Note that these factors are permutable to block triangular form (this is why we refer to the factorization as block-LU) and that $\mathbf{W}_z$ is the Schur-complement for $\mathbf{p}_d$ and is given by

$$\mathbf{W}_z = \mathbf{W}_{ss} + \mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{W}_{ss}\mathbf{A}_s^{-1}\mathbf{A}_d - \mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{W}_{sd} - \mathbf{W}_{ds}\mathbf{A}_s^{-1}\mathbf{A}_d. \tag{8}$$

Based on the Schur-type factorization we use the following preconditioner for the KKT system:

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_d^T\tilde{\mathbf{A}}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{A}}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{W}}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_s^T \end{bmatrix}. \tag{9}$$

The key components of the preconditioner are $\tilde{\mathbf{A}}_s^{-1}$ and $\tilde{\mathbf{W}}_z^{-1}$, the preconditioners for the forward problem and the reduced space (i.e. decision space) equations respectively. A natural choice for $\tilde{\mathbf{W}}_z^{-1}$ is a BFGS-like method which is commonly used to approximate the reduced Hessian in QN-RSQP methods. For an analysis of the RSQP-based preconditioner and more details on the derivations, see [6]. In [4–7] we give theoretical and numerical evidence that these preconditioners work well.

3 Globalization

Algorithm 1 is only locally convergent. Popular methodologies to globalize Newton's method—that is, allow convergence to a local minimum from any initial guess—include line search, trust region, and filter algorithms. Details can be found in [23]. Trust region methods have been successfully applied to PDE-constrained optimization [17], [19], [20]. Global convergence proofs for these methods can be found in [8]. Trust region methods are often based on the Steihaug modification [24] of the Conjugate Gradient (CG) algorithm. However, CG requires a positive (semi)definite system. For reduced space approaches, this is not a problem, since the reduced Hessian is assumed to be positive definite. To employ a trust region method within the full space, one could minimize the l_2 norm of the KKT conditions, effectively applying CG to the normal equations, but this would entail a squaring of the already very large condition number of the KKT system. Instead, we have opted for a line search algorithm. Since the first order necessary conditions (4) we are solving admit non-optimal stationary points as solutions, the line search procedure may fail to find a point that sufficiently reduces the merit function. In this case, we revert from a full-space Newton step back to a quasi-Newton-based RSQP step, which under appropriate conditions guarantees finding such a point. Our final strategy for globalization is the use of continuation on a nonlinear parameter. Such techniques are very popular and effective for solution of nonlinear PDEs that are otherwise difficult to solve, and it makes sense to exploit them for PDE optimization. In the remainder of this section, we discuss our line search, combined QN-RSQP–LNKS, and continuation strategies.

3.1 The line search

The basic component of a line search algorithm is the choice of a merit function, a scalar function (of $\mathbf{x}$ and λ) that monitors the progress of the algorithm. In contrast with unconstrained optimization, the choice of a merit function is not straightforward, since we are trying to balance optimality with feasibility. The two most popular choices are the l_1 and the augmented Lagrangian exact merit functions. The l_1 merit function is given by

$$\phi(\mathbf{x}) := f + \rho_\phi \|\mathbf{c}\|_1, \tag{10}$$

and the augmented Lagrangian by

$$\phi(\mathbf{x}, \boldsymbol{\lambda}) := f + \mathbf{c}^T \boldsymbol{\lambda} + \frac{\rho_\phi}{2} \mathbf{c}^T \mathbf{c}. \tag{11}$$

The scalar ρ_ϕ is the *penalty parameter*—a weight chosen to balance minimization of the objective function and the minimization of the residuals of the constraints. Both merit functions are exact provided the penalty parameter is large enough. By exact we mean that if $(\mathbf{x}_*, \boldsymbol{\lambda}_*)$ is a minimizer for (1), then it is also a minimizer for the merit function. A crucial property of a merit function is that it should accept unit step lengths close to a solution, and therefore permit Newton quadratic convergence to be observed. The l_1 merit function often suffers from the "Maratos" effect, that is, it sometimes rejects good steps and slows down the algorithm. The augmented Lagrangian merit function does not exhibit such behavior, but its drawback is that it requires accurate estimates of the Lagrange multipliers to perform well. (This is not a problem in LNKS since second-order accurate Lagrange multipliers estimates are computed.) Both algorithms are sensitive to the penalty parameter, which must be chosen judiciously to avoid an unbounded merit function, or very slow convergence.

In LNKS we use an Armijo-type line search algorithm. A safeguarded backtracking procedure is used to search for a scalar $\alpha \in [\alpha_{min}, 1]$ so that the so-called Armijo criterion

$$\phi(\alpha) \leq \phi(0) + \alpha \delta_A \mathbf{p}^T \nabla \phi(0), \tag{12}$$

of sufficient descent is satisfied. The algorithm used to compute the search direction $\mathbf{p}$ is left intentionally unspecified. All that matters to ensure global convergence is the properties of the merit function and the properties of $\mathbf{p}$. If ϕ is bounded and takes its minimum at a finite point, and if $\mathbf{p}$ is bounded, the safeguarded Armijo search is guaranteed to converge to a local minimum [22]. The line search algorithm we use is simple backtracking.

3.2 Continuation

One of the standard assumptions in global convergence proofs is that the Jacobian of the constraints is nonsingular for all optimization iterations. For highly nonlinear PDEs this may be an unrealistic assumption. Even if the Jacobian is nonsingular, severe ill-conditioning will cause both QN-RSQP and LNKS algorithms to stall. Indeed, in our numerical experiments (for iterates far from the solution), difficulties in the line search algorithm are correlated with difficulties converging the $\mathbf{A}_s$ and $\mathbf{K}$ linear solves.

Continuation is a popular method for addressing such convergence difficulties. Many nonlinear problems are characterized by a scalar parameter whose increase tracks the shrinking diameter of the attraction basin of Newton's method. In its simplest form, one begins with a value of this parameter for which Newton's method converges easily, and then solves a sequence of problems with larger values of the parameter, until the target value is reached. Each problem is initialized with the converged state field of the previous iterate. Examples of such parameters are the Reynolds number in viscous flow, the Mach number in compressible flow, the Hartman number in magnetohydrodynamics, the load parameter for nonlinear solid mechanics problems, and more generally the grid size for many problems that exhibit increasing nonlinearity with finer numerical resolution. In problems where such a parameter is not natural, one can construct a surrogate problem that is easy to solve in the parameter limit; an example is the pseudo-transient continuation method [18].

The popularity of continuation for solving PDEs suggests their use in PDE optimization methods, which after all must converge the PDEs as part of the optimality conditions. The extension is natural; the continuation steps are promoted up to the level of the optimization iteration. Continuation allows uphill steps (unlike monotone line search methods) to be taken and generates good initial guesses, not only for the optimization variables, but also for the penalty parameter in the merit function. The most important feature of the continuation algorithm is that it globalizes trivially (when the initial optimum can be computed reliably, and when all iterates on the continuation path are far from turning and bifurcation points). If the continuation step places the next iterate outside the attraction basin of the Newton method then we simply reduce the continuation step size. In principle, the method can be made to work without incorporating any other globalization strategy. Nevertheless, taking a large number of continuation steps can significantly slow down the algorithm. Therefore additional globalization strategies are necessary.

3.3 Combining QN-RSQP with LNKS

Quasi-Newton methods are well known for their robustness. Since LNKS already uses the reduced space structure for preconditioning, it is natural to ask whether QN-RSQP can be utilized to enhance the robustness of the overall algorithm. Global convergence proofs require the reduced Hessian, $\mathbf{W}_z$, to be strictly positive definite. If $\mathbf{W}_z$ is positive definite (and assuming the system (5) is solved exactly), then the resulting step $\mathbf{p}$ satisfies the descent criterion. This is where quasi-Newton methods have an advantage over Newton methods. For example, by using a BFGS approximation, $\tilde{\mathbf{W}}_z$, positive definiteness can be guaranteed. LNKS does maintain a BFGS approximation—not for driving the outer iteration but for preconditioning purposes. Therefore, the remedy for an indefinite reduced Hessian is simple: if a computed search direction fails to satisfy the line search conditions, we discard it, and fall back on a search direction computed by QN-RSQP. For this reason, even when using a different preconditioner for the reduced Hessian, we insist on maintaining a BFGS approximation of $\mathbf{W}_z$ for globalization purposes.

As the factorization (7) illustrates, reduced space methods can be derived formally by a linear elimination of the state space step $\mathbf{p}_s$. The resulting unconstrained optimization problem has a gradient that includes second derivatives. These 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 [2]. The resulting QN-RSQP method is defined by Algorithm 2. An important advantage of this quasi-Newton method is that only two linearized forward prob-

Algorithm 2 Quasi-Newton RSQP

```
1: Choose $\mathbf{x}_s, \mathbf{x}_d, \tilde{\mathbf{W}}_z$
2: loop
3:   Evaluate $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$
4:   $\mathbf{A}_s^T \boldsymbol{\lambda} = -\mathbf{g}_s$                solve for $\boldsymbol{\lambda}$ (Adjoint Step)
5:   $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T \boldsymbol{\lambda}$
6:   Update $\tilde{\mathbf{W}}_z$                      (Quasi-Newton approximation)
7:   if $\|\mathbf{g}_z\| \leq tol$ and $\|\mathbf{c}\| \leq tol$ then
8:     Converged
9:   end if
10:  $\tilde{\mathbf{W}}_z \mathbf{p}_d = -\mathbf{g}_z$               solve for $\mathbf{p}_d$ (Decision step)
11:  $\mathbf{A}_s \mathbf{p}_s = -(\mathbf{A}_d \mathbf{p}_d + \mathbf{c})$     solve for $\mathbf{p}_s$ (State step)
12:  $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
13: end loop
```

lems need to be solved at each iteration, as opposed to the m needed by N-RSQP for constructing $\mathbf{A}_s^{-1}\mathbf{A}_d$ in $\mathbf{W}_z$, e.g. [13].

Let us review how QN-RSQP is used in combination with the l_1 and augmented Lagrangian merit functions. For the l_1 merit function we have

$$\nabla\phi^T \mathbf{p}_x = \mathbf{g}^T \mathbf{p}_x - \rho_\phi \|\mathbf{c}\|_1 = -\mathbf{g}_z^T \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z - \boldsymbol{\lambda}^T \mathbf{c} - \rho_\phi \|\mathbf{c}\|_1 .$$

If $\tilde{\mathbf{W}}_z^{-1}$ is positive definite then the first term is always positive and we can choose ρ_ϕ by making the remaining terms positive. By setting

$$\rho_\phi = \|\boldsymbol{\lambda}\|_\infty + \delta, \quad \delta > 0, \tag{13}$$

we obtain a descent direction.

Similarly for the augmented Lagrangian we get

$$\begin{aligned}\nabla\phi^T \mathbf{p} &= (\mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} + \rho_\phi \mathbf{A}^T \mathbf{c})^T \mathbf{p}_x + \mathbf{p}_\lambda, \\ &= -\mathbf{g}_z^T \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z - \boldsymbol{\lambda}^T (\mathbf{c} + \mathbf{A}^T \mathbf{p}_x) + \mathbf{c}^T \mathbf{A} \mathbf{p}_x + \mathbf{c}^T \mathbf{p}_\lambda, \\ &= -\mathbf{g}_z^T \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z - \rho_\phi \mathbf{c}^T \mathbf{c} + \mathbf{c}^T \mathbf{p}_\lambda .\end{aligned}$$

If we assume that $\tilde{\mathbf{W}}_z^{-1}$ is strictly positive definite and choose

$$\rho_\phi \geq \frac{\mathbf{c}^T \mathbf{p}_\lambda}{\mathbf{c}^T \mathbf{c}} + \delta, \quad \delta > 0, \tag{14}$$

then a descent direction is guaranteed. In our quasi-Newton formulations we assume that second derivatives are available. In this case,

$$\mathbf{p}_\lambda = (\partial_x \boldsymbol{\lambda}) \mathbf{p}_x = -\mathbf{A}_s^{-T} \begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} \end{bmatrix} \mathbf{p}_x \approx -\tilde{\mathbf{A}}_s^{-T} \begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} \end{bmatrix} \mathbf{p}_x . \tag{15}$$

QN-RSQP can be parallelized very efficiently for moderate numbers of decision variables [21]. But when QN-RSQP is used to globalize LNKS, the need for exact forward solves greatly increases the cost of the LNKS method. By carrying out the state and adjoint solves that make up QN-RSQP inexactly, we can make the globalization much more efficient. How to do this will be discussed in the next section.

4 Inexact computations within LNKS

In large-scale computations inexactness is a powerful way to accelerate computations. In addition, it is often the case that inexactness robustifies algorithms (e.g. by damping Newton steps). In LNKS both outer and inner iterations are performed inexactly. We use inexact Lagrange-Newton solves within continuation loops, inexact Krylov-Schur solves to compute the Newton direction, and an inexact reduced Hessian in the KKT preconditioner and the QN-RSQP globalization. The various types of inexactness influence the algorithm in two basic ways: global convergence to a KKT point and local convergence rates. Analysis is required not only to provide theoretical guarantees for the robustness of the proposed algorithms, but also to suggest choices for truncation tolerances.

4.1 Inexact Newton's method

Before we discuss inexact Newton's method in the context of LNKS, we briefly summarize a few results for a general nonlinear system of equations. Assume we want to solve $\mathbf{h}(\mathbf{q}) = \mathbf{0}$. Further assume the following: (1) $\mathbf{h}$ and $\mathbf{K} := \partial_q \mathbf{h}$ are sufficiently smooth in a neighborhood of a solution $\mathbf{q}_*$; (2) at each iteration an inexact Newton method computes a step $\mathbf{p}$ that satisfies

$$\|\mathbf{K}\mathbf{p} + \mathbf{h}\| \le \eta_N \|\mathbf{h}\|, \tag{16}$$

where η_N is often called the *forcing term*. It can be shown that if $\eta_N < 1$ then $\mathbf{q} \to \mathbf{q}^*$ linearly; if $\eta_N \to 0$ then $\mathbf{q} \to \mathbf{q}^*$ superlinearly; and if $\eta_N = \mathcal{O}(\|\mathbf{h}\|)$ then we recover the quadratic convergence rate of a Newton method. The forcing term is usually given by

$$\eta_N = \frac{\|\mathbf{h}_{(+)} - \mathbf{h} - \mathbf{K}\mathbf{p}\|}{\|\mathbf{h}\|}. \tag{17}$$

For other choices of η_N and details on inexact Newton methods, see [10] and the references therein.

The extension of inexact methods to optimization is immediate, especially for unconstrained optimization. In [17] a global analysis is provided for a trust region RSQP-based algorithm. Close to a KKT point the theory for Newton's method applies and one can use the analysis presented in [9] to show that the inexact version of the LNKS algorithm converges. However, the line search we are using is not based on the residual of the KKT equations but instead on the merit function discussed in the previous session. That means that an inexact step that simply reduces $\|\mathbf{h}\|$ may not necessarily satisfy the merit function criteria. In [7] we show that for points that are close enough to the solution inexactness does not interfere with the line search.

Here we extend our discussion to the inexact QN-RSQP algorithm since, especially in the absence of a continuation scheme, QN-RSQP globalization is a crucial component of the LNKS method. In the analysis that follows we compare the inexact computation with an exact one. We assume that we have chosen a penalty parameter that gives sufficient decrease (based on the exact steps), and we establish conditions for the inexact computation so that this sufficient decrease is not compromised.

In the presence of inexactness the QN-RSQP steps (Algorithm 2) become

$$\begin{aligned}
\mathbf{A}_s^T\tilde{\boldsymbol{\lambda}} + \mathbf{g}_s &= \mathbf{r}_z,\\
\tilde{\mathbf{g}}_z &= \mathbf{g}_d + \mathbf{A}_d^T\tilde{\boldsymbol{\lambda}},\\
\tilde{\mathbf{W}}_z\tilde{\mathbf{p}}_z &= -\tilde{\mathbf{g}}_z,\\
\mathbf{A}_s\tilde{\mathbf{p}}_s &= -(\mathbf{A}_d\tilde{\mathbf{p}}_z + \mathbf{c}) + \mathbf{r}_c,\\
\tilde{\mathbf{p}}_d &= \tilde{\mathbf{p}}_z.
\end{aligned}$$

We have introduced two vectors, $\mathbf{r}_z$ and $\mathbf{r}_c$, to account for the inexactness in the adjoint and forward solves. The following equations give the Lagrange multipliers, reduced gradient, and state and control steps for the exact (left column) and the inexact case (right column).

$$\begin{aligned}
\boldsymbol{\lambda} &= -\mathbf{A}_s^{-T}\mathbf{g}_s, & \tilde{\boldsymbol{\lambda}} &= \boldsymbol{\lambda} + \mathbf{A}_s^{-T}\mathbf{r}_z,\\
\mathbf{g}_z &= \mathbf{g}_d - \mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{g}_s, & \tilde{\mathbf{g}}_z &= \mathbf{g}_z + \mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z,\\
\mathbf{p}_d &= -\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z, & \tilde{\mathbf{p}}_d &= \mathbf{p}_d - \tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z,\\
\mathbf{p}_s &= \mathbf{A}_s^{-1}\mathbf{A}_d\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z - \mathbf{A}_s^{-1}\mathbf{c}, & \tilde{\mathbf{p}}_s &= \mathbf{p}_s + \mathbf{A}_s^{-1}\mathbf{A}_d\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z + \mathbf{A}_s^{-1}\mathbf{r}_c.
\end{aligned}$$

Let us define the following constants : $\kappa_1 := \max(\|\tilde{\mathbf{W}}_z^{-1}\|)$, $\kappa_2 := \max(\|\mathbf{A}_s^{-1}\mathbf{A}_d\|)$, $\kappa_3 := \min(\sigma_{\min}(\tilde{\mathbf{W}}_z^{-1}))$, and $\kappa_4 := \max(\|\mathbf{A}_s^{-1}\|)$; σ denotes singular values and the $\min, \max$ operations are across optimization iterations. We assume that these quantities are uniformly bounded.

4.2 l_1 merit function

The directional derivative of the merit function is given by

$$\begin{aligned}
\nabla\phi^T\mathbf{p}_x &= \nabla\phi^T\mathbf{p}_x + \mathbf{g}_s^T(\mathbf{A}_s^{-1}\mathbf{A}_d\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z + \mathbf{A}_s^{-1}\mathbf{r}_c) - \mathbf{g}_d^T\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z,\\
&= \nabla\phi^T\mathbf{p}_x + (\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{g}_s - \mathbf{g}_d)^T\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z + (-\mathbf{A}_s^{-T}\mathbf{g}_s)^T\mathbf{r}_c,\\
&= -\mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z - \boldsymbol{\lambda}^T\mathbf{c} - \rho_\phi\|\mathbf{c}\|_1 - \mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z - \boldsymbol{\lambda}^T\mathbf{r}_c.
\end{aligned}$$

To ensure that $\nabla\phi^T\mathbf{p}_x < 0$ we can set

$$\begin{aligned}
-\mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z - \mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z &< 0,\\
-\boldsymbol{\lambda}^T\mathbf{c} - \boldsymbol{\lambda}^T\mathbf{r}_c - \rho_\phi\|\mathbf{c}\|_1 &< 0,
\end{aligned} \tag{18}$$

and therefore if we choose

$$\|\mathbf{r}_z\| < \frac{\kappa_3}{\kappa_1\kappa_2}\|\mathbf{g}_z\|, \tag{19}$$

we satisfy the first inequality in (18).

If we assume that the penalty parameter is given by (13) then

$$\begin{aligned}
&-\boldsymbol{\lambda}^T\mathbf{c} - \boldsymbol{\lambda}^T\mathbf{r}_c - \rho_\phi\|\mathbf{c}\|_1 < 0,\\
&-\boldsymbol{\lambda}^T\mathbf{c} - \boldsymbol{\lambda}^T\mathbf{r}_c - \|\boldsymbol{\lambda}\|_\infty\|\mathbf{c}\|_1 - \delta\|\mathbf{c}\|_1 <\\
&\qquad \|\boldsymbol{\lambda}\|_\infty\|\mathbf{c}\|_1 + \|\boldsymbol{\lambda}\|_\infty\|\mathbf{r}_c\|_1 - \|\boldsymbol{\lambda}\|_\infty\|\mathbf{c}\|_1 - \delta\|\mathbf{c}\|_1.
\end{aligned}$$

If we choose

$$\|\mathbf{r}_c\|_1 < \frac{1}{2}\frac{\delta}{\|\boldsymbol{\lambda}\|_\infty}\|\mathbf{c}\|_1, \tag{20}$$

then[1]

$$-\boldsymbol{\lambda}^T\mathbf{c} - \boldsymbol{\lambda}^T\mathbf{r}_c - \rho_\phi\|\mathbf{c}\|_1 < -\frac{1}{2}\delta\|\mathbf{c}\|_1.$$

For each iterate we therefore compute sufficient descent without having to increase the penalty parameter.

4.3 Augmented Lagrangian merit function

For the inexact QN-RSQP method, the directional derivative of the augmented Lagrangian merit function becomes

$$\begin{aligned}
\nabla\phi^T\tilde{\mathbf{p}} &= \tilde{\mathbf{p}}_x^T(\mathbf{g} + \mathbf{A}^T\tilde{\boldsymbol{\lambda}} + \rho_\phi\mathbf{A}^T\mathbf{c}) + \mathbf{c}^T\tilde{\mathbf{p}}_\lambda,\\
&= -\mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z - \rho_\phi\mathbf{c}^T\mathbf{c} + \mathbf{e}^T(\mathbf{g} + \mathbf{A}^T\tilde{\boldsymbol{\lambda}} + \rho_\phi\mathbf{A}^T\mathbf{c}) + \mathbf{c}^T\tilde{\mathbf{p}}_\lambda +\\
&\qquad + \mathbf{p}_x^T\mathbf{A}^T\mathbf{e}_\lambda + \mathbf{e}^T\mathbf{A}^T\mathbf{e}_\lambda,
\end{aligned} \tag{21}$$

where

$$\begin{aligned}
\mathbf{e}_s &:= \mathbf{A}_s^{-1}\mathbf{A}_d\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z + \mathbf{A}_s^{-1}\mathbf{r}_c,\\
\mathbf{e}_d &:= -\tilde{\mathbf{W}}_z^{-1}\mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{r}_z,\\
\mathbf{e}_\lambda &:= \mathbf{A}_s^{-T}\mathbf{r}_z.
\end{aligned}$$

Now we examine the different terms of the gradient of the Augmented Lagrangian function. We use $\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda} = \{\mathbf{0} \quad \mathbf{g}_z\}^T$ and (21) becomes

$$\begin{aligned}
\nabla\phi^T\tilde{\mathbf{p}} = -\mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z - \rho_\phi\mathbf{c}^T\mathbf{c} +&\\
+ \mathbf{g}_z^T\mathbf{e}_d + \mathbf{p}_x^T\mathbf{A}^T\mathbf{e}_\lambda + \mathbf{e}^T\mathbf{A}^T\mathbf{e}_\lambda + \rho_\phi\mathbf{e}^T\mathbf{A}^T\mathbf{c} + \mathbf{c}^T\tilde{\mathbf{p}}_\lambda.&
\end{aligned} \tag{22}$$

[1] In our implementation we use $\|\mathbf{r}_c\|_1 < \frac{1}{2}\delta/(\|\boldsymbol{\lambda}\|_\infty + 1)\|\mathbf{c}\|_1$ and the right hand side becomes $\delta/2(1/(1/\|\boldsymbol{\lambda}\|_\infty + 1) - 2)\|\mathbf{c}\|_1$. By noticing that $0 < \frac{1}{1/\|\boldsymbol{\lambda}\|_\infty + 1} < 1$ we get the sufficient descent condition.

It is easy to check that the terms in the right hand of (22) simplify to:

$$
\begin{aligned}
\mathbf{g}_z^T \mathbf{e}_d &= -\mathbf{r}_z^T \mathbf{A}_s^{-1} \mathbf{c}, \\
\mathbf{p}_x^T \mathbf{A}^T \mathbf{e}_\lambda &= -\mathbf{r}_z^T \mathbf{A}_s^{-1} \mathbf{c}, \\
\rho_\phi \mathbf{e}^T \mathbf{A}^T \mathbf{c} &= \rho_\phi \mathbf{c}^T \mathbf{r}_c, \\
\mathbf{e}^T \mathbf{A}^T \mathbf{e}_\lambda &= \mathbf{r}_z^T \mathbf{A}_s^{-1} \mathbf{r}_c.
\end{aligned}
$$

Thus

$$
\begin{aligned}
\nabla \phi^T \tilde{\mathbf{p}} = &- \mathbf{g}_z^T \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z - \rho_\phi \mathbf{c}^T \mathbf{c} + \rho_\phi \mathbf{c}^T \mathbf{r}_c - \\
&- \mathbf{r}_z^T (\mathbf{A}_s^{-1} \mathbf{c} + \mathbf{A}_s^{-1} \mathbf{A}_d \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z) + \mathbf{c}^T \tilde{\mathbf{p}}_\lambda + \mathbf{r}_z^T \mathbf{A}_s^{-1} \mathbf{r}_c. \qquad (23)
\end{aligned}
$$

In the following we assume that $\mathbf{c}^T \tilde{\mathbf{p}}_\lambda$ is absorbed in the penalty parameter ρ_ϕ. We use the following inequality

$$
\mathbf{g}_z^T \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z + \rho_\phi \mathbf{c}^T \mathbf{c} \leq 2 \max(\kappa_3 \|\mathbf{g}_z\|^2, \rho_\phi \|\mathbf{c}\|^2) =: \gamma.
$$

If we choose

$$
\mathbf{r}_z^T (\mathbf{A}_s^{-1} \mathbf{c} + \mathbf{A}_s^{-1} \mathbf{A}_d \tilde{\mathbf{W}}_z^{-1} \mathbf{g}_z) < \eta \frac{1}{2} \gamma, \quad 0 < \eta < 1
$$

then

$$
\|\mathbf{r}_z\| < \frac{\eta}{2} \frac{\max(\kappa_3 \|\mathbf{g}_z\|^2, \rho_\phi \|\mathbf{c}\|^2)}{\max(\kappa_4 \|\mathbf{c}\|, \kappa_2 \kappa_3 \|\mathbf{g}_z\|)}. \qquad (24)
$$

Similarly if we choose

$$
\mathbf{r}_c^T (\rho_\phi \|\mathbf{c}\| + \kappa_4 \mathbf{r}_z) < \eta \frac{1}{2} \gamma, \quad 0 < \eta < 1
$$

then

$$
\|\mathbf{r}_c\| < \frac{\eta}{2} \frac{\max(\kappa_3 \|\mathbf{g}_z\|^2, \rho_\phi \|\mathbf{c}\|^2)}{\max(\rho_\phi \|\mathbf{c}\|, \kappa_4 \|\mathbf{r}_z\|)}. \qquad (25)
$$

By insisting that (24) and (25) are satisfied, we guarantee a descent direction without a penalty parameter significantly larger than that of the exact case. Of course these relations are not implemented in this form since we do not know $\|\mathbf{g}_z\|$. Instead of $\|\mathbf{g}_z\|$ we use the reduced gradient norm from the previous iteration—scaled by a standard inexact Newton forcing parameter η_N computed from

$$
\eta_N^+ = \frac{\|\tilde{\mathbf{g}}_z^+ - \tilde{\mathbf{g}}_z - \alpha \tilde{\mathbf{p}}_d\|}{\|\tilde{\mathbf{g}}_z\|}
$$

at the end of each SQP step.

Notice that the analysis is different from the case of the l_1 merit function. In the latter the errors $\mathbf{r}_z$ and $\mathbf{r}_c$ have been compared to $-\mathbf{g}_z^T\tilde{\mathbf{W}}_z^{-1}\mathbf{g}_z$ and $\rho_\phi\mathbf{c}^T\mathbf{c}$ respectively, whereas in the augmented Lagrangian both errors are compared to $\frac{1}{2}\gamma$. This allows a balance between feasibility and optimality. If, for example, we start very close to a feasible point but far from the optimum, an inexact Newton's criterion based only on the residual of the constraints (as in (20)) will oversolve, since far from the optimum we do not need the constraints to be satisfied. Of course it is very easy to extend the analysis for the augmented Lagrangian to the l_1 merit function.

4.4 The globalized inexact LNKS method

Algorithm 3 gives a high-level description of the globalized inexact version of LNKS. Some of the implementation details and heuristics are discussed in this section. We use the following notation: $\mathbf{q} = \{\mathbf{x} \;\; \boldsymbol{\lambda}\}^T$; $\phi(0) := \phi(\mathbf{q})$; $\mathbf{h}(0) := \mathbf{h}(\mathbf{q})$; $\phi(\alpha) := \phi(\mathbf{q} + \alpha\mathbf{p})$; and $\mathbf{h}(\alpha) := \mathbf{h}(\mathbf{q} + \alpha\mathbf{p})$.

The algorithm uses a three-level iteration. In the outer iteration the value of the continuation parameter (here symbolized by Re, the Reynolds number for viscous flows, following the examples of the next section) is gradually increased until the target value is reached. The middle iterations correspond to Lagrange-Newton linearizations of the optimality system for a fixed continuation number. Finally, the inner iteration consists of two core branches: the computation of a Newton direction and the computation of the search direction with QN-RSQP. The default branch is the Newton step. If this step fails to satisfy the line search conditions, we then switch to QN-RSQP. If QN-RSQP fails too, then we reduce the continuation parameter Re and return to the outer loop.

The linear solves in Steps 8, 16 and 17 are performed inexactly (that is, by early termination of iterative solvers). In Step 8 we follow [10] in choosing the forcing term. In Steps 16 and 17 the forcing term is based on the formulas developed in Section 4.3.

In Step 6 we use the adjoint variables to update the reduced gradient. This is equivalent to $\mathbf{g}_z = \mathbf{g}_d - \mathbf{A}_d^T\mathbf{A}_s^{-T}\mathbf{g}_s$, if $\boldsymbol{\lambda}$ is computed by solving exactly $\mathbf{A}_s^T\boldsymbol{\lambda} + \mathbf{g}_s = \mathbf{0}$. When $\boldsymbol{\lambda}$ is taken from LNKS, it includes second order terms (which reduce to zero as we approach the solution), and when $\boldsymbol{\lambda}$ is taken from QN-RSQP it also introduces extra error since we never solve the linear systems exactly. In our numerical experiments this approximation has not caused problems.

We allow for non-monotone line searches. If the LNKS step is rejected by the merit function line search we do not switch immediately to QN-RSQP. Instead, we perform a line search (Step 12) on the KKT residual (as if we were treating the KKT conditions as nonlinear equations) and if the step is accepted we use it to update the variables for the next iteration. However, we do store the iterate and the merit function gradient, and we insist that some step satisfies the conditions of the merit line search (evaluated at the failure point) after a fixed number of iterations. Otherwise, we switch to QN-RSQP. This heuristic has been very successful. Typically, we permit two steps before we demand reduction of the merit function.

```
Algorithm 3 Globalized LNKS
1: Choose $\mathbf{x}_s,\ \mathbf{x}_d,\ \rho_\phi,\ t,\ \delta_A$, set $Re = Re_{start},\ tol = tol_0$
2: $\mathbf{A}_s^T\boldsymbol{\lambda} + \mathbf{g}_s \approx \mathbf{0}$                    solve inexactly for $\boldsymbol{\lambda}$
3: while $Re \neq Re_{target}$ do
4:   loop
5:     Evaluate $f,\ \mathbf{c},\ \mathbf{g},\ \mathbf{A},\ \mathbf{W}$
6:     $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T\boldsymbol{\lambda}$
7:     Check convergence: $\|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\| \leq tol$ and $\|\mathbf{c}\| \leq tol$
8:     $\mathbf{P}^{-1}\mathbf{K}\mathbf{p} + \mathbf{P}^{-1}\mathbf{h} \approx \mathbf{0}$                    solve inexactly for $\mathbf{p}$
9:     Compute $\rho_\phi$ such that $\nabla\phi^T(0)\mathbf{p} \leq 0$
10:    Compute $\alpha$ s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{p})$
11:    if Line search failed then
12:      Compute $\alpha$ s.t. $\|\mathbf{h}(\alpha)\| < t\|\mathbf{h}(0)\|$
13:    end if
14:    if Line search failed then
15:      $\tilde{\mathbf{W}}_z\mathbf{p}_d = -\mathbf{g}_z$                    solve inexactly for $\mathbf{p}_d$
16:      $\mathbf{A}_s\mathbf{p}_s + \mathbf{A}_d\mathbf{p}_d + \mathbf{c} \approx \mathbf{0}$                    solve inexactly for $\mathbf{p}_s$
17:      $\mathbf{A}_s^T\boldsymbol{\lambda}_+ + \mathbf{g}_s \approx \mathbf{0}$                    solve inexactly for $\boldsymbol{\lambda}_+$
18:      Compute $\alpha$ s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{p})$
19:      if Line search failed then
20:        Reduce Re and go to step 5.
21:      end if
22:    end if
23:    $\boldsymbol{\lambda}+ = \boldsymbol{\lambda} + \mathbf{p}_\lambda$                    (only for LNKS step)
24:    $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
25:   end loop
26:   $Re = Re + \Delta Re$
27:   Tighten $tol$
28: end while
```

We use various heuristics to bound the penalty parameter and if possible reduce it. A new penalty parameter ρ_ϕ^+ is computed using the LNKS step and formula (14). If $\rho_\phi^+ > 4\rho_\phi$ we update the penalty parameter and we switch to QN-RSQP. If $\rho_\phi^+ < \rho_\phi/4$ we *reduce* the penalty parameter and set $\rho_\phi^+ = 0.5\rho_\phi$. We also reduce the penalty parameter after successful steps in the KKT residual.

We use the BFGS method for the quasi-Newton approximation of the reduced Hessian. To precondition $\mathbf{W}_z$ we use either BFGS or a matrix-free method we introduced in [6]. This preconditioner, which requires the action of $\mathbf{W}_z$ on a vector, can be also used as a driver for the reduced space globalization step. Although we have the luxury of second derivatives, computing the reduced Hessian exactly is very expensive. Instead we use an approximate reduced Hessian, given by

$$\tilde{\mathbf{W}}_z = \mathbf{W}_{dd} + \tilde{\mathbf{A}}_s^{-T}\mathbf{A}_d^T\mathbf{W}_{ss}\tilde{\mathbf{A}}_s^{-1}\mathbf{A}_d - \mathbf{W}_{ds}\tilde{\mathbf{A}}_s^{-1}\mathbf{A}_d - \mathbf{A}_d^T\tilde{\mathbf{A}}_s^{-T}\mathbf{W}_{sd}, \qquad (26)$$

and $\tilde{\mathbf{A}}_s^{-1}$ is the preconditioner to the forward problem. We employ a Lanczos process to estimate the lower and upper eigenvalues of $\tilde{\mathbf{W}}_z$. In case of a negative eigenvalue (i.e. negative curvature) we can use a modified reduced Hessian, $\mu\mathbf{I} + \tilde{\mathbf{W}}_z$, where the parameter μ is chosen to shift the spectrum to the positive real axis.

5 Application to optimal boundary control of viscous flows

In this section we present results that typify the performance of the LNKS algorithm. The PDE-constrained optimization problem we consider is finding the optimal boundary control (suction/injection) that minimizes the rate of energy dissipation in a viscous flow. The flow (and hence state constraints) is described by the stationary incompressible Navier-Stokes equations. A survey and a number articles on flow control can be found in [15]. More on numerical approximation of the incompressible Navier-Stokes equations can be found in [14,16]. We consider flow around a circular cylinder, which is anchored inside a rectangular duct, much like a numerical wind tunnel. A quadratic velocity profile is used as an inflow Dirichlet boundary condition and we prescribe a traction-free outflow. The control variables (i.e. decision variables) are the velocities $\boldsymbol{d}$ on the downstream portion of the cylinder surface. We use the velocity-pressure $(i.e.\ \boldsymbol{u}, p)$ form of the incompressible steady state Navier-Stokes equations. The objective function(al) is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{d}) := \frac{\nu}{2}\int_{\Omega} \nabla\boldsymbol{u} \cdot \nabla\boldsymbol{u}^T + \frac{\rho}{2}\int_{\Gamma_d} \boldsymbol{d}\cdot\boldsymbol{d},$$

where the first term is the energy dissipation, and the second reflects the "cost" of the boundary velocity controls. The usual approach of defining a Lagrangian functional and requiring its stationarity with respect to Lagrange multipliers, state variables, and decision variables gives (the strong, infinite dimensional form of) the KKT optimality conditions, which consist of the *forward* problem

$$\begin{gathered}
-\nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + (\nabla\boldsymbol{u})\boldsymbol{u} + \nabla p = \boldsymbol{b} \quad \text{in} \quad \Omega, \\
\nabla\cdot\boldsymbol{u} = 0 \quad \text{in} \quad \Omega, \\
\boldsymbol{u} = \boldsymbol{u}_g \quad \text{on} \quad \Gamma_u, \\
\boldsymbol{u} = \boldsymbol{d} \quad \text{on} \quad \Gamma_d, \\
-p\boldsymbol{n} + \nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\boldsymbol{n} = \boldsymbol{0} \quad \text{on} \quad \Gamma_N,
\end{gathered} \tag{27}$$

the *adjoint* problem

$$\begin{gathered}
-\nu\nabla\cdot(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T) + (\nabla\boldsymbol{u})^T\boldsymbol{\lambda} - (\nabla\boldsymbol{\lambda})\boldsymbol{u} + \nabla\mu = \nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) \quad \text{in} \quad \Omega, \\
\nabla\cdot\boldsymbol{\lambda} = 0 \quad \text{in} \quad \Omega, \\
\boldsymbol{\lambda} = \boldsymbol{0} \quad \text{on} \quad \Gamma_u, \\
\boldsymbol{\lambda} = \boldsymbol{0} \quad \text{on} \quad \Gamma_d, \\
-\mu\boldsymbol{n} + \nu(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + (\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{\lambda} = -\nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\boldsymbol{n} \quad \text{on} \quad \Gamma_N,
\end{gathered} \tag{28}$$

and the *control* (i.e. decision) problem

$$\nu(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + \nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\boldsymbol{n} - \rho\boldsymbol{d} = \boldsymbol{0} \quad \text{on} \quad \Gamma_d. \tag{29}$$

Here $\nu = 1/Re$ and the decision variables are the velocities $\boldsymbol{d}$ on Γ_d; $\boldsymbol{\lambda}$ are the adjoint velocities and μ are the adjoint pressures. For a forward solve we need not distinguish between Γ_d and Γ_u. In the optimization problem, however, $\boldsymbol{u}_d$ is to be determined.

We discretize by the Galerkin finite element method, using tetrahedral Taylor-Hood elements (quadratic velocities, linear pressures). Our software is built on top of the PETSc library [1] and we use PETSc's block-Jacobi preconditioners with local ILU(0) for the domain decomposition approximation of the forward and adjoint operators. For the Krylov solves of the forward and adjoint problems, we use the quasi-minimum residual method (QMR) [12], and for the KKT Krylov solves we use a symmetric variant of QMR.

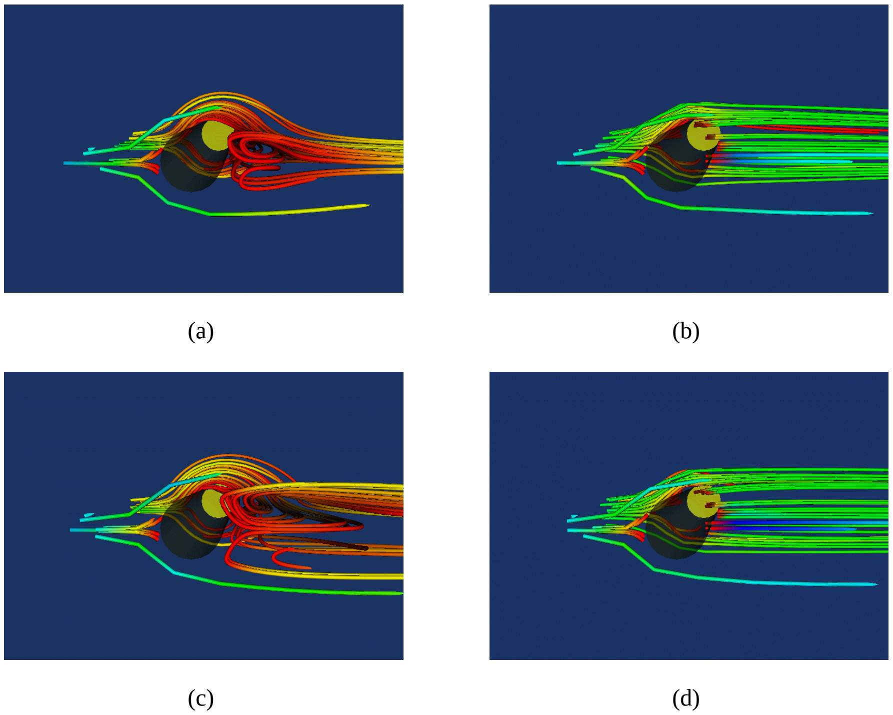

Fig. 1. PDE-constrained optimal control problem. The constraints are the steady three-dimensional incompressible Navier-Stokes equations modeling viscous flow around a cylinder. The objective is to minimize a linear combination of the energy dissipation in the fluid and the cost of the controls. The controls are injection/suction velocities on the downstream portion of the cylinder surface. The left images depict streamtubes for the uncontrolled flow at Re 20 (top) and 40 (bottom). The right images depict streamtubes of the optimally-controlled flow (same Reynolds numbers). Injecting fluid entirely eliminates recirculation within the wake of the cylinder, thus minimizing dissipation. The optimization problem was solved on 256 processors of the Cray T3E-900 at the Pittsburgh Supercomputing Center.

Figure 1 illustrates the optimization results for different Reynolds numbers. The optimal controls eliminate the recirculation region within the cylinder wake. This is achieved by injecting fluid on the downstream portion of the cylinder. We observe a tenfold relative reduction of the dissipation functional (which is proportional to drag on the cylinder).

Table 1 shows results for 32, 64, and 128 processors of a Cray T3E-900 for a roughly doubling of problem size. We compare QN-RSQP (exact solves), with LNKS (exact solves) and IN-LNKS (inexact solves). Continuation was used for the

Table 1. Scalability of QN-RSQP and LNKS algorithms for solution of the optimal flow control problem on 32, 64, and 128 processors of the Cray T3E-900. Results correspond to a roughly doubling of problem size for each doubling of number of processors. **QN-RSQP** is quasi-Newton reduced-space SQP; in **LNKS** we terminate the KKT Krylov iterations when the Euclidean norm of residual is less than 0.9×10^{-7}; in **IN-LNKS** we use a inexact Newton method on the KKT conditions; **N or QN iter** is the number of Newton or quasi-Newton steps; **KKT iter** is the number of inner iterations averaged across the outer iterations; **time** is wall-clock time in hours. Continuation was used for Re=60.

$Re = 30$

states controls	**method**	**N or QN iter**	**KKT iter**	**time**
117,048	QN-RSQP	161	—	32.1
2,925	LNKS	6	1,367	5,7
(32 procs)	IN-LNKS	11	163	1.4
389,440	QN-RSQP	189	—	46.3
6,549	LNKS	6	2,153	15.7
(64 procs)	IN-LNKS	13	238	3.8
615,981	QN-RSQP	204	—	53.1
8,901	LNKS	6	3,583	16.8
(128 procs)	IN-LNKS	12	379	4.1

$Re = 60$

states controls	**preconditioning**	**Newton iter**	**average KKT iter**	**time (hours)**
117,048	QN-RSQP	168	—	33.4
2,925	LNKS	7	1,391	6,8
(32 procs)	IN-LNKS	11	169	1.5
389,440	QN-RSQP	194	—	49.1
6,549	LNKS	7	2,228	18.9
(64 procs)	IN-LNKS	15	256	4.8
615,981	QN-RSQP	211	—	57.3
8,901	LNKS	8	3,610	13.5
(128 procs)	IN-LNKS	16	383	5.1

initial guess at Re 60 by using the solution from Re 30 as the initial guess. The reduced Hessian preconditioner is a BFGS approximation, initialized using several iterations of a 2-step stationary iterative method [6]. For this problem, QN-RSQP successfully converged but only after a significant amount of time.[2] LNKS does much better—4 to 5 times faster than QN-RSQP. The most notable finding in Table 1 is the dramatic acceleration of LNKS that is achieved by allowing inexactness (IN-LNKS). The inexactness did not interfere at any point with the merit function and in all cases we observed quadratic convergence. For both Re 30 and 60, IN-LNKS converges over an order of magnitude more quickly than QN-RSQP.

In Table 2 we compare LNKS with the inexact version of the QN-RSQP method. In these numerical tests we have chosen Reynolds numbers in which the steady state model of the flow is not correct physically (i.e. a steady state does not exist for this value of Re). This was done to increase the nonlinearity of the problem, to stress the globalizations used in LNKS. BFGS is chosen as the quasi-Newton update in both the QN-RSQP method and the QN-RSQP preconditioner for LNKS.

The effect of inexact computations is examined in Table 2. As can be seen from the iterations column (**N or QN itr**), the number of outer iterations depends very mildly on the nonlinearity of the problem. In the fourth column (**ls failed**) we measure the success of the line search algorithms. Its meaning is overloaded; for the IN-QN-RSQP method it indicates how many times the penalty parameter increased to greater than twice that of the maximum penalty parameter encountered in the exact QN-RSQP solves; for the LNKS methods it indicates failures of the merit function line search. If the latter happens we switch to a line search on the KKT residual. The fifth column (**KKT failed**) indicates the number of times that this approach failed (and as a result we had to backtrack the outer iteration and switch to a quasi-Newton step). For this numerical experiment we observe that the number of excessive penalty parameter increments within the inexact QN-RSQP is relatively small. Overall the inexact methods are decidedly faster than exact versions, as shown in the last column, which gives CPU timings.

For the LNKS methods we see that for exact solves switching to a line search on the KKT residual does not help (in fact it slows down the algorithm). On the contrary, for inexact solves this approach helps, and the relatively expensive quasi-Newton steps are usually avoided.

6 Conclusions

We presented the basic algorithmic components of the LNKS method, and considered inexact variants that speed up the method without compromising convergence. We considered an application to the optimal control of a viscous flow around a cylinder by boundary suction/injection. Our tests illustrate that LNKS is a robust and scalable algorithm for PDE-constrained optimization. It exhibits the well-known

[2] Here we used the l_1 merit function with second order correction line search. In past experiments we used standard l_1 and after 48 hours QN-RSQP was terminated with just two orders of magnitude reduction in the reduced gradient.

Table 2. Results for three different Reynolds numbers for the 117,048 states problem on 32 processors. Exact and inexact variants of both QN-RSQP and LNKS algorithms are presented. **QN-RSQP** is quasi-Newton reduced-space SQP; **IN-QN-RSQP** is the inexact QN-RSQP method; in **LNKS** we terminate the KKT Krylov iterations when the Euclidean norm of residual is less than 0.9×10^{-7}; in **IN-LNKS** we use an inexact Newton method for the KKT conditions; **itr** is the number of Newton (or quasi-Newton) steps; for IN-QN-RSQP, **ls failed** indicates the number of excessive increases of the merit function penalty parameter ρ_ϕ; for the two LNKS methods, it indicates the number of unsuccessful augmented Lagrangian line search attempts; **KKT failed** indicates the number of iterations in which the KKT steps had to be rejected; **time** is wall-clock time in hours on the T3E-900. In this example we did not employ continuation.

Reynolds	**method**	**N or QN itr**	**ls failed**	**KKT failed**	**time**
90	QN-RSQP	181	-	-	35.4
	IN-QN-RSQP	184	5	-	22.1
	LNKS	9	1	1	7.2
	IN-LNKS	14	4	0	1.5
120	QN-RSQP	185	-	-	36.1
	IN-QN-RSQP	192	5	-	23.2
	LNKS	10	2	2	8.1
	IN-LNKS	15	6	1	2.3
150	QN-RSQP	184	-	-	36.3
	IN-QN-RSQP	194	6	-	25.1
	LNKS	11	2	2	8.6
	IN-LNKS	15	6	2	2.9

mesh-independence convergence properties of Newton methods, which combined with the inner Krylov-Schur iteration results in a very fast method. Often, the optimal solution is found in a small multiple of the cost of a single forward problem solve.

Inexactness is introduced into the LNKS method through inexact Lagrange-Newton solves within continuation loops, early termination of Krylov-Schur iterations to compute the Newton direction, and a BFGS approximation of the reduced Hessian based on inexact state Jacobian solves (used both as a reduced space preconditioner within the KKT solves as well as within the QN-RSQP globalization). The numerical experiments on the effects of inexactness are of limited scope, yet give an indication of effectiveness of inexact computations in reducing wall-clock time while retaining robustness. Typically, introducing inexactness into LNKS led to a factor of 3 to 4 reduction in cost relative to exact LNKS. Moreover, the augmented Lagrangian globalization we tested performed robustly and we did not have convergence problems for the (highly) nonlinear problem of separated flow around a cylinder. The results reveal at least an order of magnitude improvement in time over conventional quasi-Newton methods, rendering tractable some problems that otherwise would have required unacceptable amounts of parallel supercomputing time.

References

1. Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc home page. `http://www.mcs.anl.gov/petsc`, 1999.
2. Lorenz T. Biegler, Jorge Nocedal, and Claudia Schmid. A reduced Hessian method for large-scale constrained optimization. *SIAM Journal on Optimization*, 5:314–347, 1995.
3. George Biros. *Parallel Algorithms for PDE-Constrained Optimization and Application to Optimal Control of Viscous Flows*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 2000.
4. George Biros and Omar Ghattas. Parallel Newton-Krylov algorithms for PDE-constrained optimization. In *Proceedings of SC99*, The SCxy Conference series, Portland, Oregon, November 1999. ACM/IEEE.
5. George Biros and Omar Ghattas. Parallel preconditioners for KKT systems arising in optimal control of viscous incompressible flows. In D. E. Keyes, A. Ecer, J. Periaux, and N. Satofuka, editors, *Parallel Computational Fluid Dynamics 1999*. North-Holland, 1999.
6. George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver. Technical report, Laboratory for Mechanics, Algorithms, and Computing, Carnegie Mellon University, 2000.
7. George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange Newton solver, and its application to optimal control of steady viscous flows. Technical report, Laboratory for Mechanics, Algorithms, and Computing, Carnegie Mellon University, 2000.
8. John Dennis E., Jr., Mahmoud El-Alem, and Maria C. Magiel. A global convergence theory for general trust-region-based algorithms for equality constrained optimization. *SIAM Journal on Optimization*, 7(1):177–207, 1997.

9. Stanley C. Eisenstat and Homer F. Walker. Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.
10. Stanley C. Eisenstat and Homer F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.
11. Roger Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, second edition, 1987.
12. Roland W. Freund and Noël 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.
13. Omar Ghattas and Jai-Hyeong Bark. Optimal control of two- and three-dimensional incompressible Navier-Stokes flows. *Journal of Computational Physics*, 136:231–244, 1997.
14. Max D. Gunzburger. *Finite Element for Viscous Incompressible Flows*. Academic Press, 1989.
15. Max D. Gunzburger, editor. *Flow Control*, volume 68 of *IMA Math. Appl.* Springer-Verlag, New York, 1995.
16. Max D. Gunzburger and Roy A. Nicolaides, editors. *Incompressible Computational Fluid Dynamics*. Cambridge University Press, 1993.
17. Matthias Heinkenschloss and Luis N. Vicente. Analysis of inexact trust-region SQP algorithms. Technical Report TR99-18, Rice University, Department of Computational and Applied Mathematics, 1999.
18. C. T. Kelley and David E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35:508–523, 1998.
19. C.T. Kelley and Ekkehard W. Sachs. Truncated Newton methods for optimization with inaccurate functions and gradients. *SIAM Journal on Optimization*, 10(1):43–55, 1999.
20. F. Leibritz and E. W. Sachs. Inexact SQP interior point methods and large scale optimal control problems. *SIAM Journal on Control and Optimization*, 38(1):272–293, 1999.
21. Ivan Malčević. Large-scale unstructured mesh shape optimization on parallel computers. Master's thesis, Carnegie Mellon University, 1997.
22. Stephen G. Nash and Ariela Sofer. *Linear and Nonlinear Programming.* McGraw-Hill, 1996.
23. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.
24. T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20:626–637, 1983.