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, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA (oghattas@cs.cmu.edu).

Abstract. In this article we present an outline of the Lagrange-Newton-Krylov-Schur (LNKS) method and we discuss how we can improve its work efficiency by carrying out certain computations inexactly, without compromising convergence. LNKS has been designed for PDE-constrained optimization problems. It solves the Karush-Kuhn-Tucker optimality conditions by a Newton-Krylov algorithm. Its key component is a preconditioner based on quasi-Newton reduced space Sequential Quadratic Programming (QN-RSQP) variants. LNKS combines the fast-convergence properties of a Newton method with the capability of preconditioned Krylov methods to solve very large linear systems. Nevertheless, even with good preconditioners, the solution of an optimization problem has a cost which is several times higher than the cost of the 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, approximate line searches. Naturally, several issues arise with respect to the trade-offs between speed and robustness.

1 Introduction

In this article we discuss algorithmic and implementation aspects of large-scale PDE-constrained optimization methods. The proposed techniques have applications to a broad category of optimization problems: optimal control, optimal design, and parameter identification for systems governed by partial differential equations.

We refer to the unknown PDE field quantities as the *state variables*; the PDE constraints as the *state equations*; solution of the PDE constraints as the *forward problem*; 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*.

The most popular technique for PDE constrained optimization problems is the reduced space Sequential Quadratic Programming method (RSQP): at each iteration the linearized PDE-constraints are eliminated and an unconstrained optimization problems is solved in the reduced space—the decision variables space [2], [17]. LNKS, which was first introduced by the authors in [3,6,7], employs a family of methods that use Newton-Krylov algorithms to solve for the Karush-Kuhn-Tucker optimality conditions, 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 refer to 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 the most basic components of LNKS. We introduced and analyzed the preconditioner(s) in [4,6]. We also examined the parallelizability and scalability of the LNKS algorithm. The basic form of the LNKS algorithm and the Lagrange-Newton solver are analyzed in [7].

Like quasi-Newton RSQP [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 a preconditioner); for example we replace them by an appropriate PDE preconditioner. LNKS builds on existing parallel PDE preconditioners and to an extend it parallelizes and scales as well as the forward solver itself. In this paper we follow up with a discussion on the inexact components of the LNKS method—in particular the inexact solves within the QN-RSQP steps used for globalization.

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

Notation conventions: 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 in our context are the discretized state equations; we assume that these are the only constraints[1].

[1] The methodology can be extended to problems that include additional inequality constraints.

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

$$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{x}_s \\ \mathbf{x}_d \end{array} \right\}, \tag{2}$$

so that $N = m + n$.

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 to 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:

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

Customarily, these equations are called the Karush-Kuhn-Tucker or KKT optimality conditions. Points at which the gradient of the Lagrangian vanishes are often called *KKT points*[2].

A Newton step on the optimality conditions is given by

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

where $\mathbf{p}_x$ and $\mathbf{p}_\lambda$ are the 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. 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 expected to be 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 to make the method efficient.

[2] Saddle points and local maxima are also KKT points.

In LNKS we use a proper Newton method to solve for 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: an *inexact* version of the QN-RSQP algorithm. An outline of the LNKS method is given by Algorithm 1.

Algorithm 1

```
1: Choose x, λ
2: loop
3:   Check for convergence
4:   Compute c, g, A, W
5:   Solve P^{-1}Kp = P^{-1}h                    (Newton Step)
6:   Update x = x + p_x
7:   Update λ = λ + p_λ
8: end loop
```

To derive the preconditioner we rewrite the (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 variables update direction $\mathbf{p}_d$. 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 (or decision space) equations re-

spectively. A natural choice for $\tilde{\mathbf{W}}_z^{-1}$ is a BFGS like method [3] which is commonly used 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 based on the Steihaug modification [24] of the Conjugate Gradient (CG) algorithm. However, this approach works only with positive definite systems. Since the reduced Hessian is assumed to be positive definite CG can be used. It is not obvious how to use a trust-region method with an indefinite Krylov solver (which is required for the KKT system) and thus we have opted a line search algorithm.

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-merit and the augmented Lagrangian exact penalty 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 bring the right balance between the 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 an 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, sometimes it 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 it provides second

[3] The name comes from the inventors of the method, Broyden, Fletcher, Goldfarb, and Shanno.

order accurate estimates of the Lagrange multipliers.) Both algorithms are sensitive to the penalty parameter which must be judiciously chosen; otherwise it can result in an unbounded merit function, or very slow convergence.

In LNKS we use an Armijo-type line search algorithm. In this class of algorithms 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.

3.1 Continuation

One of the standard assumptions in global convergence proofs is that the Jacobian of the constraints is non-singular for all optimization iterations. For highly nonlinear PDEs like the Navier-Stokes equations this is 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 where correlated to difficulties converging the $\mathbf{A}_s$ and $\mathbf{K}$ linear solves.

A method to deal with highly nonlinear problems is continuation. This idea (in its simplest form) works when we can express the nonlinearity of the problem as a function of a single scalar parameter. Continuation is particularly suitable for PDE-constrained optimization because it is quite typical for a PDE to have a parameter that scales the nonlinearities. Examples of such parameters are the Reynolds and Mach numbers in fluid mechanics, the Peclet number in general convection diffusion equations, and the Hartman number in magnetohydrodynamics. In problems where such a parameter cannot be found an alternative method is the pseudo-transient continuation [18].

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[4]. If the continuation step brings the next iterate outside the attraction basin of the Newton method then we can 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.

[4] This is true only when all iterates on the continuation path are far from turning and bifurcation points.

3.2 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 algorithm, we discard it, and we replace it with a search direction computed by QN-RSQP. For this reason, even if we use a different preconditioner for the reduced Hessian, we do maintain a BFGS approximation of $\mathbf{W}_z$ in order to used it with QN-RSQP.

As we saw, (7), reduced space methods can be formally derived 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]. An important advantage of this quasi-Newton method is that only two linearized

Algorithm 2 Quasi-Newton RSQP

Choose $\mathbf{x}_s, \mathbf{x}_d, \tilde{\mathbf{W}}_z$
loop
  Evaluate $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$
  $\mathbf{A}_s^T \boldsymbol{\lambda} = -\mathbf{g}_s$ solve for $\boldsymbol{\lambda}$ (**Adjoint Step**)
  $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T \boldsymbol{\lambda}$
  Update $\tilde{\mathbf{W}}_z$ (**Quasi-Newton approximation**)
  if $\|\mathbf{g}_z\| \leq tol$ and $\|\mathbf{c}\| \leq tol$ **then**
    Converged
  end if
  $\tilde{\mathbf{W}}_z \mathbf{p}_d = -\mathbf{g}_z$ solve for $\mathbf{p}_d$ (**Decision step**)
  $\mathbf{A}_s \mathbf{p}_s = -(\mathbf{A}_d \mathbf{p}_d + \mathbf{c})$ solve for $\mathbf{p}_s$ (**State step**)
  $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
end loop

forward problems 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 $\mathbf{W}_z$ [13].

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

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

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 we have second derivatives. 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 has been very efficiently parallelized for moderate numbers of decision variables [21]. However the need of exact forward solves greatly increases the cost of LNKS in case we need to use QN-RSQP for globalization. However, QN-RSQP can be made much more efficient by carrying the state and adjoint solves inexactly.

4 Inexact computations within LNKS

In very 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 dumping 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 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

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

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 rates 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 alternative selections for η_N and details in inexact Newton method look at [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 satisfy the merit function criteria. In [7] we show that for points which are close enough to the solution inexactness does not interfere with the line search.

In this article we extend our discussion to the inexact QN-RSQP algorithm since, especially in the absence of a continuation scheme QN-RSQP 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 become

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

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, 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

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

If we choose

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

then[5]

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

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

4.3 Augmented Lagrangian

For the approximate QN-RSQP the directional derivative of the augmented Lagrangian merit function becomes

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

[5] 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.

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 $\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda} = \{0 \quad \mathbf{g}_z\}^T$ and (21) becomes

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

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

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

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

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

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

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

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

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

By insisting on (24) and (25) are satisfied, we guarantee a descent direction without a penalty parameter significantly larger 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 than 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)) we 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

We summarize by giving a high-level description of implementation details and heuristics we are using in LNKS (Alg. 3). We use he following notation: $\mathbf{q} = \{\mathbf{x}\ \boldsymbol{\lambda}\}^T$; $\phi(0) := \phi(\mathbf{q})$, $\mathbf{h}(0) := \mathbf{h}(\mathbf{q})$, $\phi(\alpha) := \phi(\mathbf{q} + \alpha\mathbf{p})$, $\mathbf{h}(\alpha) := \mathbf{h}(\mathbf{q} + \alpha\mathbf{p})$.

The algorithm uses a three-level iteration. In the outer iteration the continuation parameter number is gradually increased until the target number 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 algorithm conditions then we switch to QN-RSQP. If QN-RSQP fails too, then we reduce the continuation parameter Re and we return to the outer loop.

Linear solves at steps 8, 16 and 17 are performed inexactly. 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 do a line search (step 12) on the KKT residual (as if we were treating the KKT

Algorithm 3 Globalized LNKS

Choose $\mathbf{x}_s$, $\mathbf{x}_d$, ρ_ϕ, t, δ_A, set $Re = Re_{start}$, $tol = tol_0$
$\mathbf{A}_s^T\boldsymbol{\lambda} + \mathbf{g}_s \approx \mathbf{0}$ solve inexactly for $\boldsymbol{\lambda}$
while $Re \neq Re_{target}$ **do**
  loop
    Evaluate f, $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$
    $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T\boldsymbol{\lambda}$
    Check convergence: $\|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\| \leq tol$ and $\|\mathbf{c}\| \leq tol$
    $\mathbf{P}^{-1}\mathbf{K}\mathbf{p} + \mathbf{P}^{-1}\mathbf{h} \approx \mathbf{0}$ solve inexactly for $\mathbf{p}$
    Compute ρ_ϕ such that $\nabla\phi^T(0)\mathbf{p} \leq 0$
    Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{p})$
    if Line search failed **then**
      Compute α s.t. $\|\mathbf{h}(\alpha)\| < t\|\mathbf{h}(0)\|$
    end if
    if Line search failed **then**
      $\tilde{\mathbf{W}}_z\mathbf{p}_d = -\mathbf{g}_z$ solve inexactly for $\mathbf{p}_d$
      $\mathbf{A}_s\mathbf{p}_s + \mathbf{A}_d\mathbf{p}_d + \mathbf{c} \approx \mathbf{0}$ solve inexactly for $\mathbf{p}_s$
      $\mathbf{A}_s^T\boldsymbol{\lambda}_+ + \mathbf{g}_s \approx \mathbf{0}$ solve inexactly for $\boldsymbol{\lambda}_+$
      Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{p})$
      if Line search failed **then**
        Reduce Re and go to step 5.
      end if
    end if
    $\boldsymbol{\lambda}+ = \boldsymbol{\lambda} + \mathbf{p}_\lambda$ (only for LNKS step)
    $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
  end loop
  $Re = Re + \Delta Re$
  Tighten tol
end while

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.

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, the reduced Hessian is very expensive to be

compute exactly. Instead we can use an approximate reduced Hessian. It is given by

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

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 (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 Viscous Flows

In this section we present some indicative results for the performance of the algorithm. The forward problem is a viscous flow around a cylinder. A survey and articles on flow control can be found in [15]. More on the Navier-Stokes equations can be found in [14,16]. The objective function to be minimized is the dissipation functional. The cylinder is anchored inside a rectangular duct, much like a numerical wind tunnel. A quadratic velocity profile is used as an inflow Dirichlet condition and we prescribe a traction-free outflow. The decision variables are defined to be the velocities on the downstream portion of the cylinder surface. We use the velocity-pressure $(\boldsymbol{u}, p)$ form of the incompressible steady state Navier-Stokes equations. For a dissipation minimization problem, in which the decision variables are Dirichlet velocity conditions on part of the boundary, a possible objective function is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{d}) := \frac{\nu}{2}a(\boldsymbol{u}, \boldsymbol{u}) + \frac{\rho}{2}(\boldsymbol{d}, \boldsymbol{d})_{\Gamma_d}.$$

For this problem the strong form of the (infinite dimensional) KKT optimality conditions is given by the *forward* problem

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

the *adjoint* problem

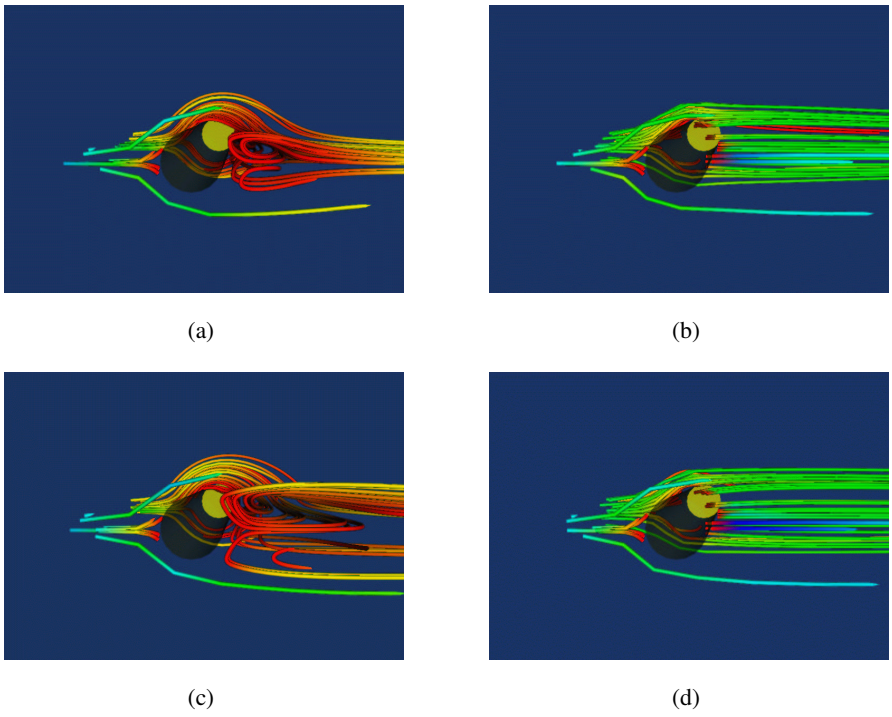
$$\begin{aligned}
-\nu\nabla\cdot(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T) + (\nabla\boldsymbol{u})^T\boldsymbol{\lambda} - (\nabla\boldsymbol{\lambda})\boldsymbol{u} + \nabla\mu = \nu\nabla\cdot(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) \quad &\text{in} \quad \Omega, \\
\nabla\cdot\boldsymbol{\lambda}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

and the *decision* or (reduced space) 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 $\boldsymbol{u}_d$ 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 not known.

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 problem inverse. For the Krylov solves of the forward and the adjoint problems we use quasi-minimum residual method (QMR) [12] and for the KKT Krylov solves we use a symmetric variant of QMR.



(a) (b)

(c) (d)

Fig. 1. This is an example of a PDE-constrained optimal control problem. The constraints are the steady incompressible Navier-Stokes equations; they model a viscous flow around a cylinder. The objective is to minimize the energy dissipation. The controls are injection points (velocity Dirichlet boundary conditions) on the downstream portion of the cylinder surface. The left images depicts the uncontrolled flow at Reynolds numbers 20 and 40. The right images depicts the controlled flow after the optimizer was switched on (same Reynolds number). Injecting fluid entirely eliminates recirculation at the wake of the cylinder, thus minimizing dissipation. This numerical experiment took place on 256 processors on a Cray T3E-900 at the Pittsburgh Supercomputing Center.

Figure 1 illustrates the optimization results for different Reynolds numbers. LNKS manages to eliminate the recirculation region in the downstream region of the cylinder. This is achieved by injecting fluid on the downstream portion of the cylinder. We observed a tenfold relative reduction on the dissipation functional.

Table 1 shows results for 32, 64, and 128 processors of a 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 initial guess

Table 1. The table shows results for 32, 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. (**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; (**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 on a T3E-900. 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

at Reynolds number 60 by using the solution from Reynolds number 30 as the initial guess. The reduced Hessian preconditioner is a combination of the BFGS and 2-step

preconditioners. For this problem, QN-RSQP successfully converged but only after a quite significant amount of time[6]. LNKS does much better—4 to 5 times faster than QN-RSQP. The most notable finding in Table 1 is the dramatic acceleration of the LNKS algorithm which is achieved by using IN-LNKS—the inexact version of the Newton method. The inexactness did not interfere at any point with the merit function and in all cases we observed quadratic convergence. For both Reynolds number 30 and 60 IN-LNKS runs more than 10 times faster 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—we did it to increase the nonlinearity of the problem. We use BFGS both as a driver and as a preconditioner.

We look at the number of Newton or quasi-Newton steps and on the effect of the inexact computations. As we can see in this example the number of outer iterations depends very mildly on the nonlinearity of the problem. The inexact versions are much faster; 30% for the QN-RSQP and more than 50% for the full space approach. In the third column we measure the success of the line search algorithms. Its meaning is overloaded; for the IN-QN-RSQP method it indicates the number of times the merit function penalty parameter was increased twofold the maximum penalty parameter during the exact QN-RSQP solves; for the LNKS method it indicates failures on the merit function line search. If the latter happens we switch to a line search on the KKT residual. The fifth column 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 method performs much better than the exact case.

For the LNKS steps we see that in the case of 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 the inexact solves this approach helps and the relatively expensive quasi-Newton steps are avoided.

6 Conclusions

We presented the basic algorithmic components of the LNKS method and we applied to the optimal control of a viscous flow around a cylinder. Our tests indicate that LNKS is a robust and scalable algorithm for PDE-constrained optimization. The Lagrange-Newton method exhibited the well known mesh-independence convergence properties—combined with the inner Krylov-Schur iteration results in very fast solvers.

The numerical experiments on the effects inexactness are of limited scope—nevertheless the give an indication of the effectiveness of inexact computations in reducing wall-clock time. Undoubtedly the external cylinder flow problem is highly

[6] 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. This table shows results for three different Reynolds numbers for the 117,048 states problem on 32 processors. Results for the QN-RSQP and LNKS algorithms are presented. (**QN-RSQP**) is the quasi-Newton reduced-space SQP; (**IN-QN-RSQP**) is the inexact reduced space 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 a truncated Newton method; (**itr**) is the number for Newton (or quasi-Newton) steps; (**ls failed**): for IN-QN-RSQP it indicates number of excessive increase on ρ_ϕ the merit function penalty parameter, and for LNKS 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 a 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

nonlinear. The augmented Lagrangian globalization performed robustly and we did not have problems converging the equations. The results reveal at least an order of magnitude improvement in time over popular quasi-Newton methods, rendering tractable some problems that were out of reach previously. Indeed, the optimum is often found in a *small* multiple of the cost of a single simulation.

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.