SIAM J. SCI. COMPUT.
Vol. 32, No. 6, pp. 3447–3475

AN INTERIOR-POINT ALGORITHM FOR LARGE-SCALE NONLINEAR OPTIMIZATION WITH INEXACT STEP COMPUTATIONS[*]

FRANK E. CURTIS[†], OLAF SCHENK[‡], AND ANDREAS WÄCHTER[§]

Abstract. We present a line-search algorithm for large-scale continuous optimization. The algorithm is matrix-free in that it does not require the factorization of derivative matrices. Instead, it uses iterative linear system solvers. Inexact step computations are supported in order to save computational expense during each iteration. The algorithm is an interior-point approach derived from an inexact Newton method for equality constrained optimization proposed by Curtis, Nocedal, and Wächter [*SIAM J. Optim.*, 20 (2009), pp. 1224–1249], with additional functionality for handling inequality constraints. The algorithm is shown to be globally convergent under loose assumptions. Numerical results are presented for nonlinear optimization test set collections and a pair of PDE-constrained model problems.

Key words. large-scale optimization, constrained optimization, interior-point methods, nonconvex optimization, trust regions, inexact linear system solvers, Krylov subspace methods

AMS subject classifications. 49M05, 49M37, 65K05, 90C06, 90C26, 90C30, 90C51

DOI. 10.1137/090747634

1. Introduction.

We consider nonlinear optimization problems of the form

$$\min_x f(x) \quad \text{subject to (s.t.)} \begin{cases} c_{\mathcal{E}}(x) = 0, \\ c_{\mathcal{I}}(x) \geq 0, \end{cases} \tag{1.1}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$ and the constraints $c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^p$ and $c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^q$ are continuously differentiable. We are particularly interested in applications where the number of variables n, the number of equality constraints p, and the number of inequality constraints q may be large, say, in the millions.

Contemporary optimization methods often require the use of derivatives throughout the solution process. Fast convergence from remote starting points requires first-order derivatives of the objective and the constraints, and fast local convergence in the neighborhood of solution points is possible with second-order derivatives in the form of the Hessian of the Lagrangian of problem (1.1). For the solution of many problems, the storage and computational requirements of factoring explicit derivative matrices are reasonable, which allows for the use of very sophisticated and efficient techniques. For this reason, optimization methods have spread and been successful throughout a number of scientific communities. In many application areas, however, the storage or factorization of derivative matrices is inefficient or intractable. This

[*]Received by the editors January 23, 2009; accepted for publication (in revised form) September 27, 2010; published electronically November 30, 2010.
http://www.siam.org/journals/sisc/32-6/74763.html

[†]Department of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA (frank.e.curtis@gmail.com). This author was supported in part by National Science Foundation grant DMS 0602235.

[‡]Department of Computer Science, University of Basel, Basel, Switzerland (olaf.schenk@unibas.ch).

[§]IBM T.J. Watson Research Center, Yorktown Heights, NY (andreasw@watson.ibm.com).

may occur, for example, if the problem is extremely large or if direct factorization creates a lot of fill-in. In such cases, many contemporary methods cannot be used, and alternative algorithms have to be developed.

One possible technique for solving large-scale problems is to reduce the original problem into one of smaller size through nonlinear elimination (e.g., see [1, 2, 19, 31, 48]). For example, this process may involve splitting the unknowns x into a set of independent variables x^i and a set of dependent variables x^d in such a way that the dependent variables can be computed uniquely by solving the equality constraints in (1.1) for a given choice of x^i. Once this operation is performed, an auxiliary system may be solved for the sensitivities of x^d with respect to x^i, and the remainder of the iteration involves only the computation of a displacement in x^i. Algorithms of this type, however, suffer from a number of drawbacks. For instance, if many iterations are required to find the optimal independent variables, then such a procedure requires numerous highly accurate solutions of the equality constraints and computations of the variable sensitivities. Furthermore, even simple bounds on dependent variables are transformed into complicated nonlinear inequality constraints in the reduced problem.

Our goal is thus to design a constrained optimization algorithm that emulates an efficient nonlinear programming approach without relying on nonlinear elimination techniques. The algorithm may utilize matrix-vector products with the constraint Jacobian, its transpose, and the Hessian of the Lagrangian of problem (1.1) together with appropriate preconditioners—quantities that are computable for many large-scale applications of interest—but should avoid exact factorizations of derivative matrices. Iterative linear system solvers present a viable alternative to direct factorization methods, but the benefits of these techniques are only realized if inexact step computations are allowed and controlled appropriately in order to guarantee global convergence of the algorithm.

A line of efficient and robust algorithms has been developed that meets these requirements for equality constrained problems [11, 12, 16]. These line-search methods illustrate how inexactness in the iterative step computations can be controlled to ensure global convergence to a solution point—first in the case of a sequential quadratic programming (SQP) framework, then when handling nonconvexity of the problem functions, and finally when the constraint functions may be ill-conditioned or even inconsistent. As an extension of these techniques, in this paper we propose and analyze an algorithm for large-scale continuous optimization problems with inequality constraints and investigate its practical performance. We show that with an appropriate scaling of the search space, the method developed in [16] can be enhanced in order to solve problems with inequality constraints. The resulting method is matrix-free and allows for inexact step computations. It is globally convergent to first-order optimal points of (1.1) or at least to stationary points of the feasibility problem

$$\min_{x} \ \tfrac{1}{2}\|c_{\mathcal{E}}(x)\|_2^2 + \tfrac{1}{2}\|c_{\mathcal{I}}(x)^-\|_2^2 \tag{1.2}$$

that yield a nonzero objective value; i.e., infeasible stationary points of problem (1.1). (Here, for a vector z we define $z^- = \max\{0, -z\}$, where the max is taken elementwise.) The method yields encouraging numerical results on the large and diverse constrained and unconstrained testing environment, revisited (CUTEr) [24, 25] and constrained optimization problem set (COPS) [18] test set collections, as well as on a pair of model PDE-constrained problems, implying that it has much potential for general-purpose large-scale nonlinear optimization.

We make particular note of the potential for our method to be employed for PDE-constrained problems; see [3, 4, 10, 29] and the references therein for discussions of recent work in this active and important field. In this area, our framework fits into the class of full-space ("all-at-once") algorithms, such as those in [5, 6, 7, 26, 38]. However, in contrast to the algorithms proposed in these papers, our method allows for inequality constraints and provides definitive and practical global convergence guarantees. Another class of PDE-constrained optimization methods consists of reduced-space (or decomposition) methods, such as the algorithms in [27, 28, 30, 39, 45]. Algorithms of this type have many strengths in terms of solving large-scale problems, but also suffer from some disadvantages when compared to full-space techniques. For example, a potential pitfall is the expense of computing approximate projections onto the null space of the constraint Jacobian multiple times during each iteration. In addition, these approaches do not allow for general inequality constraints or bound constraints on dependent (state) variables.

In general, efficient implementations of PDE-constrained optimization methods exploit structure through the use of problem-specific preconditioners. Our proposed framework is flexible enough to allow for such preconditioning, but at this point our preliminary implementation uses an algebraic preconditioner for the full primal-dual matrix, which exploits only sparsity. Nevertheless, our numerical experiments clearly illustrate a performance improvement over an algorithm that uses direct factorizations, and there are other situations when even our current implementation may be generally competitive. For instance, for some PDEs like the Helmholtz equation, problem-specific preconditioners are not always available (e.g., see [8]). Moreover, our framework allows for general inequality constraints on the state variables (i.e., state constraints), the presence of which may significantly reduce the usefulness of preconditioners for a given PDE.

We organize the paper as follows. In section 2 we present our matrix-free method, at the heart of which are conditions dictating when a given trial search direction should be considered acceptable in order to ensure global convergence of the algorithm. Section 3 contains analysis of the global behavior of the approach, while section 4 illustrates the practical behavior of a preliminary implementation of our algorithm. Finally, in section 5 we present closing remarks and a discussion of future work.

Notation. All norms are considered ℓ_2 unless otherwise indicated. We drop function dependencies once values are clear from the context and use the expression $M_1 \succeq M_2$ to indicate that the matrix $M_1 - M_2$ is positive semidefinite. Parenthesized superscripts are used to indicate the component of a vector, and subscripts are used to indicate the current iteration number in an algorithm.

2. An interior-point algorithm.

In this section, we present our algorithm. We begin by describing an interior-point framework in an environment where steps can be computed exactly from subproblems and linear systems. We then introduce the algorithmic components that are necessary to allow inexact step computations. The resulting algorithm is a line-search interior-point method with safeguards for situations when the problem is nonconvex and when the constraint Jacobians are ill-conditioned or rank deficient. We note that the components of our interior-point framework are defined in a nonstandard fashion (as explained below), but that through our formulation we may more directly extend the techniques and theory from [16].

2.1. Interior-point framework. Our algorithm follows an interior-point strategy in that problem (1.1) is solved via a sequence of barrier subproblems of the form

$$\begin{aligned} &\min_{x,s}\ f(x) - \mu \sum_{i=1}^{q} \ln s^{(i)} \\ &\text{s.t.} \begin{cases} c_{\mathcal{E}}(x) = 0, \\ c_{\mathcal{I}}(x) = s,\ \ s > 0 \end{cases} \end{aligned} \tag{2.1}$$

for decreasing values of the barrier parameter $\mu > 0$. The Lagrangian for (2.1), with multipliers $\lambda_{\mathcal{E}} \in \mathbb{R}^p$ and $\lambda_{\mathcal{I}} \in \mathbb{R}^q$, is given by

$$\mathcal{L}(x, s, \lambda_{\mathcal{E}}, \lambda_{\mathcal{I}}; \mu) := f(x) - \mu \sum_{i=1}^{q} \ln s^{(i)} + \lambda_{\mathcal{E}}^T c_{\mathcal{E}}(x) + \lambda_{\mathcal{I}}^T (c_{\mathcal{I}}(x) - s), \tag{2.2}$$

so if f, $c_{\mathcal{E}}$, and $c_{\mathcal{I}}$ are first-order differentiable, the first-order optimality conditions for (2.1) are

$$\begin{aligned} \nabla f(x) + \nabla c_{\mathcal{E}}(x)\lambda_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x)\lambda_{\mathcal{I}} &= 0, \\ -\mu S^{-1} e - \lambda_{\mathcal{I}} &= 0, \\ c_{\mathcal{E}}(x) &= 0, \\ c_{\mathcal{I}}(x) - s &= 0, \end{aligned} \tag{2.3}$$

along with $s > 0$. Here, we have defined $S = \operatorname{diag}(s)$ and $e \in \mathbb{R}^q$ as a vector of ones.

If problem (2.1) is infeasible, then the algorithm is designed to converge toward a first-order optimal solution of the feasibility problem (1.2) so that a justified declaration of infeasibility can be made. Noting that $\|c_{\mathcal{I}}(x)^-\|^2$ is differentiable with $\nabla(\|c_{\mathcal{I}}(x)^-\|^2) = -2\nabla c_{\mathcal{I}}(x) c_{\mathcal{I}}(x)^-$, such a point can be characterized as a solution to the nonlinear system of equations

$$\nabla c_{\mathcal{E}}(x) c_{\mathcal{E}}(x) - \nabla c_{\mathcal{I}}(x) c_{\mathcal{I}}(x)^- = 0.$$

If $s \geq 0$ and $c_{\mathcal{I}}(x) - s \leq 0$, then this is equivalent to

$$\begin{aligned} \nabla c_{\mathcal{E}}(x) c_{\mathcal{E}}(x) + \nabla c_{\mathcal{I}}(x)(c_{\mathcal{I}}(x) - s) &= 0, \\ -S(c_{\mathcal{I}}(x) - s) &= 0. \end{aligned} \tag{2.4}$$

Indeed, in the algorithm, $s \geq 0$ and $c_{\mathcal{I}}(x) - s \leq 0$ are enforced explicitly; see (2.13) and (2.16), respectively, below.

Let j be the outer iteration counter for solving the nonlinear program (1.1), and let k be the inner iteration counter for solving the barrier subproblem (2.1). Defining the primal and dual iterates as

$$z := \begin{bmatrix} x \\ s \end{bmatrix} \quad \text{and} \quad \lambda := \begin{bmatrix} \lambda_{\mathcal{E}} \\ \lambda_{\mathcal{I}} \end{bmatrix},$$

respectively, the barrier objective and constraints as

$$\varphi(z; \mu) := f(x) - \mu \sum_{i=1}^{q} \ln s^{(i)} \quad \text{and} \quad c(z) := \begin{bmatrix} c_{\mathcal{E}}(x) \\ c_{\mathcal{I}}(x) - s \end{bmatrix}$$

with corresponding *scaled* first derivatives

$$\gamma(z;\mu) := \begin{bmatrix} \nabla f(x) \\ -\mu e \end{bmatrix} \quad \text{and} \quad A(z) := \begin{bmatrix} \nabla c_{\mathcal{E}}(x)^T & 0 \\ \nabla c_{\mathcal{I}}(x)^T & -S \end{bmatrix}, \tag{2.5}$$

respectively, and the *scaled* Hessian of the Lagrangian as

$$W(z,\lambda;\mu) := \begin{bmatrix} \nabla^2_{xx} f & 0 \\ 0 & \Sigma \end{bmatrix} + \sum_{i=1}^{p} \lambda_{\mathcal{E}}^{(i)} \begin{bmatrix} \nabla^2_{xx} c_{\mathcal{E}}^{(i)} & 0 \\ 0 & 0 \end{bmatrix} + \sum_{i=1}^{q} \lambda_{\mathcal{I}}^{(i)} \begin{bmatrix} \nabla^2_{xx} c_{\mathcal{I}}^{(i)} & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.6}$$

a Newton iteration for (2.1) amounts to solving the linear system

$$\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ c_k \end{bmatrix}. \tag{2.7}$$

In (2.6), Σ is assumed to be a positive definite diagonal matrix. In particular, the choice $\Sigma = \mu I$ is said to result in a *primal* interior-point iteration, whereas the choice $\Sigma = \Lambda_{\mathcal{I}} S$, where $\Lambda_{\mathcal{I}} = \text{diag}(\lambda_{\mathcal{I}})$, is said to result in a *primal-dual* iteration; e.g., see [13]. The precise value of Σ, however, is not important in our analysis. We simply assume that W_k represents a bounded symmetric approximation to (2.6), which is sufficient for ensuring global convergence; see Assumption 3.3.

Notice that our definition of the primal search direction d_k is nonstandard; i.e., with the formulation above, the primal iterate is to be updated with a line-search coefficient $\alpha_k \in (0, 1]$ as

$$z_{k+1} \leftarrow z_k + \alpha_k \widetilde{d}_k \quad \text{with} \quad \widetilde{d}_k \leftarrow \begin{bmatrix} d_k^x \\ S_k d_k^s \end{bmatrix}, \tag{2.8}$$

where d_k^x and d_k^s are the components of d_k corresponding to the x and s variables, respectively. This update follows from our use of *scaled* first and second derivatives; see [13] for an example of another algorithm that uses such a scaling for the slack variables. The scaling is critical in the design of the algorithm, as the criteria in section 2.2 for controlling inexactness will be defined in terms of a scaled system.

Line-search interior-point methods that compute search directions directly via (2.7) have been shown to fail to converge from remote starting points; see [46]. Moreover, in situations where the matrix A_k is (nearly) rank deficient, it is necessary to safeguard the step computation to avoid long, unproductive search directions or to handle situations when the system is inconsistent and the Newton step is thus undefined. Following the algorithm in [16], we avoid these difficulties by replacing d_k in (2.7) by the sum of a *normal* step $v_k \in \mathbb{R}^n$ and a *tangential* step $u_k \in \mathbb{R}^n$.

The *normal* step v_k is designed as a move toward the satisfaction of a linear model of the constraints within a trust region and is defined via the subproblem

$$\begin{aligned} &\min_{v} \ \tfrac{1}{2}\|c_k + A_k v\|^2 \\ &\text{s.t.} \ \|v\| \le \omega \|A_k^T c_k\|, \end{aligned} \tag{2.9}$$

where $\omega > 0$ is a given constant. Note that the radius of the trust region in this problem is related to the conditions (2.4), which are equivalent to $A_k^T c_k = 0$. Hence, at a stationary point of the feasibility measure we have $v_k = 0$. We choose this form of a trust-region constraint, as it simplifies our analysis in section 3; see [44] and references therein for similar approaches used by other authors. We also believe this approach has a nice benefit in that, unlike typical trust region methods, we do not need to define a heuristic for updating the trust region radius.

Our *tangential* step u_k is intended as a move toward optimality that does not inhibit the progress toward feasibility attained by the normal step. It is defined implicitly via the perturbed Newton system

$$\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ -A_k v_k \end{bmatrix}, \tag{2.10}$$

which yields

$$u_k := d_k - v_k. \tag{2.11}$$

Note that (2.10) has a consistent second block of equations even if A_k is rank deficient. Moreover, if the matrix W_k is positive definite in the null space of A_k, then a solution to (2.10) corresponds to a solution to the quadratic program

$$\begin{aligned} &\min_u \ (\gamma_k + W_k v_k)^T u + \tfrac{1}{2} u^T W_k u \\ &\text{s.t. } A_k u = 0. \end{aligned} \tag{2.12}$$

However, if W_k is not sufficiently positive definite, we modify or replace it by a sufficiently positive definite approximation matrix in order to ensure that the resulting solution to (2.10) is an appropriate search direction. If a factorization is performed, then the need for such a modification can be verified by observing the inertia of the primal-dual matrix (see [34] and, e.g., the algorithm in [42]), but for our purposes we leave consideration of this issue until a precise procedure is outlined in our algorithm with inexact step computations below.

With a search direction $d_k = u_k + v_k$ computed via (2.9) and (2.10), we perform a line search by first determining the maximum stepsize $\alpha_k^{\max} \in (0,1]$ satisfying the fraction-to-the-boundary rule

$$s_k + \alpha_k^{\max} S_k d_k^s \geq (1-\eta_1) s_k \tag{2.13}$$

for a constant $\eta_1 \in (0,1)$. Then an appropriate stepsize $\alpha_k \in (0, \alpha_k^{\max}]$ is determined, yielding progress in the penalty function

$$\phi(z;\mu,\pi) := \varphi(z;\mu) + \pi \|c(z)\|, \tag{2.14}$$

where $\pi > 0$ is a penalty parameter. Denoting $D\phi_k(\widetilde{d};\mu,\pi)$ as the directional derivative of ϕ at z_k along $\widetilde{d}$ (see (2.8)) and setting π_k so that $D\phi_k(\widetilde{d}_k;\mu_j,\pi_k) < 0$, a basic sufficient decrease requirement for $\alpha_k \in (0,1]$ is given by the Armijo condition

$$\phi(z_k + \alpha_k \widetilde{d}_k;\mu_j,\pi_k) \leq \phi(z_k;\mu_j,\pi_k) + \eta_2 \alpha_k D\phi_k(\widetilde{d}_k;\mu_j,\pi_k) \tag{2.15}$$

for a constant $\eta_2 \in (0,1)$. In fact, a stronger condition than $D\phi_k(\widetilde{d}_k;\mu_j,\pi_k) < 0$ is necessary for ensuring global convergence, but for simplicity in this brief description of our interior-point framework, we leave further consideration of this issue until section 2.2.

A framework that assumes exact solutions of subproblems and linear systems is summarized as Algorithm 2.1. A number of modifications to this framework are possible, and further details are necessary in order to guarantee global convergence. However, Algorithm 2.1 is suitable for our purposes of setting up the details that follow. Notice that the fraction-to-the-boundary rule (2.13) ensures $s_k > 0$ and that the slack reset, performed after the iterate has been updated in the inner `for` loop, ensures

$$c_{\mathcal{I}}(x_{k+1}) - s_{k+1} = c_{\mathcal{I}}(x_{k+1}) - \max\{s_k + \alpha_k S_k d_k^s, c_{\mathcal{I}}(x_{k+1})\} \leq 0 \tag{2.16}$$

for all k, so an iterate z_k yielding

$$A_k^T c_k = 0 \tag{2.17}$$

(equivalently, (2.4)) is a stationary point of problem (1.2).

Algorithm 2.1 Interior-Point Framework.

```
Choose parameters $0 < \eta_1, \eta_2 < 1$ and initialize $\mu_0 > 0$
for $j = 0, 1, 2, \ldots,$ until termination criteria for (1.1) or for (1.2) is satisfied do
    if $j = 0$ then
        Initialize $(z_0, \lambda_0)$ with $s_0 > 0$ and $s_0 \geq c_{\mathcal{I}}(x_0)$
    end if
    Initialize $\pi_{-1} > 0$
    for $k = 0, 1, 2, \ldots,$ until termination criteria for (2.1) or for (1.2) is satisfied do
        Compute $d_k = u_k + v_k$ and $\delta_k$ via (2.9) and (2.10)
        Set $\widetilde{d}_k \leftarrow (d_k^x, S_k d_k^s)$
        Set $\pi_k \geq \pi_{k-1}$ so that $D\phi_k(\widetilde{d}_k; \mu_j, \pi_k) < 0$
        Choose $\alpha_k \in (0, 1]$ satisfying (2.13) and (2.15)
        Update $z_{k+1} \leftarrow z_k + \alpha_k \widetilde{d}_k$ and choose $\lambda_{k+1}$
        Set $s_{k+1} \leftarrow \max\{s_{k+1}, c_{\mathcal{I}}(x_{k+1})\}$
    end for
    Choose $\mu_{j+1}$ (so that $\{\mu_j\} \to 0$)
    Set $(z_0, \lambda_0) \leftarrow (z_k, \lambda_k)$ for the next barrier subproblem
end for
```

2.2. An interior-point method with inexact step computations. In the remainder of this section we present techniques for applying Algorithm 2.1 when the matrices A_k and W_k are not stored or factored explicitly, with the implication that exact solutions of (2.9) and (2.10) cannot be computed. The algorithm is closely related to the method for equality constrained problems presented in [16].

Our algorithm requires that the normal component v_k satisfies the following.

Normal component condition. *The normal component v_k must be feasible for problem* (2.9) *and satisfy the Cauchy decrease condition*

$$\|c_k\| - \|c_k + A_k v_k\| \geq \epsilon_v(\|c_k\| - \|c_k + \bar{\alpha}_k A_k \bar{v}_k\|) \tag{2.18}$$

for some constant $\epsilon_v \in (0, 1]$, where $\bar{v}_k = -A_k^T c_k$ is the steepest descent direction for the objective of problem (2.9) *at $v = 0$ and $\bar{\alpha}_k$ is chosen as the solution to the one-dimensional problem*

$$\begin{aligned} &\min_{\bar{\alpha}} \ \tfrac{1}{2}\|c + \bar{\alpha} A_k \bar{v}_k\|^2 \\ &s.t.\ \bar{\alpha} \leq \omega. \end{aligned} \tag{2.19}$$

A number of iterative techniques have been developed and analyzed for the inexact solution of (2.9) with solutions satisfying this condition, including the conjugate gradient or LSQR [35] algorithm with Steihaug stopping tests [43]. In our implementation described in section 4, we prefer an inexact dogleg approach [36, 37] that we found to outperform these methods, especially in cases where A_k is (nearly) rank deficient.

Given v_k satisfying the normal component condition, we next compute a tangential component and a displacement for the Lagrange multipliers by applying an iterative linear system solver to the primal-dual system (2.10). During each iteration of the iterative linear system solve, this process yields the residual vector

$$\begin{bmatrix}\rho_k\\ r_k\end{bmatrix} := \begin{bmatrix}W_k & A_k^T\\ A_k & 0\end{bmatrix}\begin{bmatrix}d_k\\ \delta_k\end{bmatrix} + \begin{bmatrix}\gamma_k + A_k^T\lambda_k\\ -A_k v_k\end{bmatrix}. \tag{2.20}$$

In reference to the dual and primal feasibility requirements, respectively, in the optimality conditions for problem (2.12), we refer to ρ_k as the *dual residual* vector and r_k as the *primal residual* vector.

We define three termination tests for the approximate solution of (2.10). A search direction will be acceptable if it satisfies at least one of these tests, where each is composed of a small set of conditions. The conditions are related to those commonly used in inexact Newton techniques for nonlinear equations [17]. However, since we seek a solution to an optimization problem and not just a root-finding problem, a number of crucial differences are introduced. Moreover, the primal-dual matrix may need to be modified within the solution process to guarantee descent for our penalty function, so we need to define conditions under which such a modification should be made. Following the terminology in [11, 12, 16], we refer to our three termination criteria for the primal-dual step computation as sufficient merit function approximation reduction termination (SMART) tests.

The first condition that appears in our tests requires that the dual residual is sufficiently small. In contrast to traditional inexact Newton methods, the primal residual vector r_k does not need to be controlled explicitly in this condition.

Dual residual condition. *The dual residual vector ρ_k must satisfy*

$$\|\rho_k\| \le \kappa \min\left\{\left\|\begin{bmatrix}\gamma_k + A_k^T\lambda_k\\ -A_k v_k\end{bmatrix}\right\|, \left\|\begin{bmatrix}\gamma_{k-1} + A_{k-1}^T\lambda_k\\ -A_{k-1} v_{k-1}\end{bmatrix}\right\|\right\} \tag{2.21}$$

for a given $\kappa \in (0,1)$.

The second condition guarantees that the tangential component will be bounded under loose assumptions. It also includes criteria that will be used to trigger a modification of W_k if problem (2.12) is nonconvex.

Tangential component condition. *The tangential component $u_k = d_k - v_k$ must satisfy*

$$\|u_k\| \le \psi \|v_k\| \tag{2.22}$$

or

$$\tfrac{1}{2}u_k^T W_k u_k \ge \theta\|u_k\|^2 \tag{2.23a}$$

$$\text{and}\quad (\gamma_k + W_k v_k)^T u_k + \tfrac{1}{2}u_k^T W_k u_k \le \zeta\|v_k\|, \tag{2.23b}$$

where $\psi \ge 0$, $\theta > 0$, and $\zeta \ge 0$ are given constants.

Finally, a third condition that appears in our tests represents a central tenet of our criteria: a nonzero primal search direction is acceptable if and only if it provides a sufficient reduction in the local model

$$m_k(d;\mu,\pi) := \varphi(z_k;\mu) + \gamma(z_k;\mu)^T d + \pi\|c(z_k) + A(z_k)d\|$$

of the penalty function ϕ at the current iterate z_k for an appropriate value of π_k. The reduction in m_k attained by d_k is defined as

$$\begin{aligned}\Delta m_k(d_k;\mu_j,\pi_k) &:= m_k(0;\mu_j,\pi_k) - m_k(d_k;\mu_j,\pi_k)\\ &= -\gamma_k^T d_k + \pi_k(\|c_k\| - \|c_k + A_k d_k\|)\end{aligned}$$

and can be computed easily for any given d_k. We show later on (see Lemma 3.5) that d_k corresponds to a search direction of sufficient descent in ϕ when Δm_k is sufficiently large. Thus, the following condition plays a crucial role in our termination conditions for the primal-dual step computation.

Model reduction condition. *The search direction $d_k = u_k + v_k$ must satisfy*

$$\Delta m_k(d_k;\mu_j,\pi_k) \geq \max\{\tfrac{1}{2}u_k^T W_k u_k, \theta\|u_k\|^2\} + \sigma\pi_k(\|c_k\| - \|c_k + A_k v_k\|) \tag{2.24}$$

for $\pi_k \geq \pi_{k-1} > 0$, where $\sigma \in (0,1)$ is a given constant and $\theta > 0$ is given in (2.23a).

We are now ready to present our three termination tests for the iterative primal-dual step computation. The first termination test is simply a compilation of the above conditions. A step satisfying this condition is a sufficiently accurate solution to the Newton equations (2.10), has a bounded tangential component u_k, and yields a direction of sufficient descent for the penalty function ϕ for the most recent value of the penalty parameter π.

Termination test 1. *A search direction (d_k, δ_k) is acceptable if the dual residual condition* (2.21) *is satisfied, the tangential component condition* (2.22) *or* (2.23) *is satisfied, and if the model reduction condition* (2.24) *holds for $\pi_k = \pi_{k-1}$.*

The second termination test amounts to an adjustment of the multiplier estimates where we temporarily suspend movement in the primal space. Inclusion of this test is necessary for situations when z_k is a stationary point for the feasibility problem (1.2) but $\gamma_k + A_k^T\lambda_k \neq 0$, as in these situations an exact solution to (2.10) may be required to produce an acceptable step. The test may also save computational expense when z_k is in the neighborhood of such points. It is important to note that for search directions satisfying only this test, the primal step components are reset to zero vectors; i.e., we set $(v_k, u_k, d_k) \leftarrow (0, 0, 0)$ and choose $\alpha_k \leftarrow 1$.

Termination test 2. *If for a given constant $\epsilon_2 > 0$ we have*

$$\|A_k^T c_k\| \leq \epsilon_2\|\gamma_k + A_k^T\lambda_k\|, \tag{2.25}$$

then a search direction $(d_k, \delta_k) \leftarrow (0, \delta_k)$ is acceptable if the dual residual condition (2.21) *holds; i.e.,*

$$\|\gamma_k + A_k^T(\lambda_k + \delta_k)\| \leq \kappa \min\left\{\|\gamma_k + A_k^T\lambda_k\|, \left\|\begin{bmatrix}\gamma_{k-1} + A_{k-1}^T\lambda_k\\ -A_{k-1}v_{k-1}\end{bmatrix}\right\|\right\}. \tag{2.26}$$

for $\kappa \in (0,1)$ in (2.21).

Unless $A_k^T c_k = 0$ for all large k, condition (2.25) ensures the algorithm will allow only a finite number of consecutive iterations where only Termination Test 2 is satisfied; i.e., it ensures that the algorithm does not focus only on reducing dual infeasibility.

The third termination test is necessary for situations when the model reduction condition cannot be satisfied without an accompanying increase in the penalty parameter. Such an increase may be needed, for example, if the current primal iterate is a stationary point for $\phi(\cdot;\mu_j,\pi_{k-1})$ that is not a solution of the barrier subproblem (2.1) for $\mu = \mu_j$ or the feasibility problem (1.2). The test requires a reduction in a local linear model of the constraints, so for this test to be considered during iteration k, we require $\|c_k\| - \|c_k + A_k v_k\| > 0$.

Termination test 3. *A search direction (d_k, δ_k) is acceptable if the dual residual condition* (2.21) *is satisfied, if the tangential component condition* (2.22) *or* (2.23) *is satisfied, and if*

$$\|c_k\| - \|c_k + A_k d_k\| \geq \epsilon_3(\|c_k\| - \|c_k + A_k v_k\|) > 0 \tag{2.27}$$

for some constant $\epsilon_3 \in (0, 1)$.

For steps satisfying only Termination Test 3, we choose

$$\pi_k \geq \frac{\gamma_k^T d_k + \max\left\{\frac{1}{2}u_k^T W_k u_k, \theta\|u_k\|^2\right\}}{(1-\tau)(\|c_k\| - \|c_k + A_k d_k\|)} =: \pi_k^{trial} \tag{2.28}$$

for a given constant $\tau \in (0, 1)$. Along with (2.27) this bound yields

$$\begin{aligned}\Delta m_k(d_k; \mu_j, \pi_k) \geq\ & \max\{\tfrac{1}{2}u_k^T W_k u_k, \theta\|u_k\|^2\} + \tau\pi_k(\|c_k\| - \|c_k + A_k d_k\|) \\ \geq\ & \max\{\tfrac{1}{2}u_k^T W_k u_k, \theta\|u_k\|^2\} + \tau\epsilon_3\pi_k(\|c_k\| - \|c_k + A_k v_k\|),\end{aligned}$$

so (2.24) is satisfied for $\sigma = \tau\epsilon_3$. From now on we assume τ, σ, and ϵ_3 are chosen to satisfy this relationship for consistency between Termination Tests 1 and 3.

As previously mentioned, W_k may need to be modified or replaced by a sufficiently positive definite matrix in order to ensure that an acceptable search direction will eventually be computed. Since an iterative linear system solver applied to (2.10) usually does not provide any information about the inertia of the primal-dual matrix, we may not be able to guarantee that W_k is positive definite in the null space of A_k. However, our global convergence analysis illustrates that even if W_k is indefinite in this space, a nonzero search direction d_k can be considered acceptable if (2.22) or (2.23a) is satisfied. Thus, during the iterative primal-dual step computation, we call for a modification of W_k according to the following rule, which at least guarantees that eventually (2.23a) will hold.

Hessian modification strategy. *Let W_k be the current scaled Hessian approximation, and let a trial step (d_k, δ_k) be given. If $u_k = d_k - v_k$ satisfies* (2.22) *or* (2.23a)*, then maintain the current W_k; otherwise, modify W_k to increase its smallest eigenvalue.*

We do not require that the modifications take on a specific form, though in our implementation we employ the common technique of adding a multiple of a positive definite diagonal matrix to W_k to increase all of its eigenvalues. For theoretical purposes, our only stipulation is that after a finite number of modifications, W_k is sufficiently positive definite and uniformly bounded; see Assumptions 3.1 and 3.3.

Once a suitable search direction has been computed, we perform a backtracking line search to compute $\alpha_k \in (0, 1]$ so that the fraction-to-the-boundary rule (2.13) is satisfied and a sufficient decrease in the penalty function ϕ is made. Rather than computing the directional derivative $D\phi_k(\widetilde{d}_k; \mu_j, \pi_k)$—which is nontrivial if $c_k = 0$—in place of (2.15) we impose the related condition

$$\phi(z_k + \alpha_k\widetilde{d}_k; \mu_j, \pi_k) \leq \phi(z_k; \mu_j, \pi_k) - \eta_2\alpha_k\Delta m_k(d_k; \mu_j, \pi_k). \tag{2.29}$$

This sufficient decrease condition makes use of the easily computed model reduction $\Delta m_k(d_k; \mu_j, \pi_k)$ and is justified by Lemma 3.5 below. Also note that the directional derivative satisfies $D\phi_k(d_k; \mu_j, \pi_k) \leq -\Delta m_k(d_k; \mu_j, \pi_k)$, and therefore (2.29) is a weaker requirement than (2.15) (see [16, Lemma 3.8]) and might allow larger steps.

Finally, the new dual iterate λ_{k+1} is required to satisfy

$$\|\gamma_k + A_k^T \lambda_{k+1}\| \leq \|\gamma_k + A_k^T(\lambda_k + \delta_k)\|. \tag{2.30}$$

Using the dual space direction δ_k to obtain $\lambda_{k+1} = \lambda_k + \beta_k \delta_k$ with a steplength coefficient $\beta_k \in [0,1]$, this inequality may be satisfied by simply setting $\beta_k \leftarrow 1$ or by performing a one-dimensional minimization of the dual feasibility measure along δ_k. A third viable option is described with our implementation in section 4.

The details of our algorithm are specified as Algorithm 2.2.

Algorithm 2.2 Interior-point algorithm with SMART tests.

```
Choose $\psi, \zeta \geq 0$, $0 < \eta_1, \eta_2, \epsilon_v, \kappa, \epsilon_3, \tau < 1$, $0 < \omega, \theta, \epsilon_2, \delta_\pi$, and set $\sigma \leftarrow \tau\epsilon_3$
Initialize $\mu_0 > 0$
for $j = 0, 1, 2, \ldots,$ until termination criteria for (1.1) or for (1.2) is satisfied do
  if $j = 0$ then
    Initialize $(z_0, \lambda_0)$ with $s_0 > 0$ and $s_0 \geq c_{\mathcal{I}}(x_0)$
  end if
  Initialize $\pi_{-1} > 0$
  for $k = 0, 1, 2, \ldots,$ until termination criteria for (2.1) or for (1.2) is satisfied do
    Compute $v_k$ satisfying the normal component condition
    Compute an approximate solution to (2.10) with the unmodified $W_k$
    while $(d_k, \delta_k)$ does not satisfy any of termination tests 1, 2, or 3 do
      Run the Hessian modification strategy to modify $W_k$ if necessary
      Compute an improved approximate solution to (2.10) with the current $W_k$
    end while
    if $(d_k, \delta_k)$ satisfies Termination Test 3 and (2.28) does not hold then
      Set $\pi_k \leftarrow \pi_k^{trial} + \delta_\pi$
    end if
    if $(d_k, \delta_k)$ satisfies only Termination Test 2 then
      Set $d_k \leftarrow 0$
    end if
    Set $\widetilde{d}_k \leftarrow (d_k^x, S_k d_k^s)$
    if $\widetilde{d}_k \neq 0$ then
      Compute the maximum $\alpha_k^{\max} \in (0,1]$ satisfying (2.13)
      Compute the smallest $l \in \mathbb{N}_0$ such that $\alpha_k \leftarrow 2^{-l}\alpha_k^{\max}$ satisfies (2.29)
    else
      Set $\alpha_k \leftarrow 1$
    end if
    Update $z_{k+1} \leftarrow z_k + \alpha_k \widetilde{d}_k$ and choose $\lambda_{k+1}$ satisfying (2.30)
    Set $s_{k+1} \leftarrow \max\{s_{k+1}, c_{\mathcal{I}}(x_{k+1})\}$
  end for
  Choose $\mu_{j+1}$ (so that $\{\mu_j\} \to 0$)
  Set $(z_0, \lambda_0) \leftarrow (z_k, \lambda_k)$ for the next barrier subproblem
end for
```

3. Algorithm analysis.

In this section we analyze the global behavior of Algorithm 2.2. As the formulation of our approach was intentionally constructed to resemble the method presented in [16], our analysis is facilitated by referring directly to results in that work. In particular, the quantities in the matrix and right-hand side of the linear system (2.7) have the same properties as those in the primal-dual system presented in [16] (i.e., the primal-dual matrix is symmetric indefinite, and its components are bounded under reasonable assumptions; see Assumption 3.3 below). Therefore, all results pertaining to the (inexact) solution of this system carry over. The major differences, however, are that the algorithm now involves slack variables, a logarithmic barrier term in the objective of (2.1), and the fraction-to-the-boundary rule (2.13). These changes require that theoretical results that depend on the line search must be reanalyzed.

We begin with the following assumption concerning a particular iteration in the solution of a given barrier subproblem (2.1).

Assumption 3.1. After a finite number of perturbations of W_k made via the Hessian modification strategy, we have $W_k \succeq 2\theta I$ for the constant $\theta > 0$ defined in the model reduction condition (2.24). Moreover, the iterative linear system solver can solve (2.10) to an arbitrary accuracy for each W_k.

This assumption is reasonable for a number of iterative linear system solvers, even in situations where the primal-dual matrix in (2.10) is singular. We note that in practice, for a given W_k the primal-dual system may be inconsistent, but as this can only be due to the singularity of W_k, it can be remedied by further modifications of W_k as in the Hessian modification strategy.

The following lemma, proved as Lemma 3.2 in [16] under the same assumptions and Hessian modification strategy, states that Algorithm 2.2 is well-posed.

LEMMA 3.2. *During iteration k of the inner* `for` *loop in Algorithm* 2.2, *one of the following holds:*

(a) *The primal-dual pair (z_{k-1}, λ_k) is first-order optimal for* (2.1) *or* (1.2)*, i.e.,*

$$A_{k-1}^T c_{k-1} = 0 \quad \textit{and} \quad \gamma_{k-1} + A_{k-1}^T \lambda_k = 0; \tag{3.1}$$

(b) *The primal-dual pair (z_k, λ_k) is first-order optimal for* (2.1) *or* (1.2)*, i.e.,*

$$A_k^T c_k = 0 \quad \textit{and} \quad \gamma_k + A_k^T \lambda_k = 0; \tag{3.2}$$

(c) *The* `while` *loop terminates finitely with a primal-dual search direction (d_k, δ_k) satisfying at least one of Termination Tests* 1, 2, *or* 3.

In cases (a) and (b) in Lemma 3.2, we state that the inner iteration has terminated finitely, so the algorithm can proceed by updating the barrier parameter and moving on to solving the next barrier subproblem (2.1) or by terminating at an infeasible stationary point of the nonlinear program (1.1). Case (c), on the other hand, guarantees that when we are not at a solution point of (2.1) or (1.2), the algorithm will produce a suitable search direction.

We split the remainder of our analysis into two subsections; in the first we consider the convergence behavior for a given barrier subproblem assuming the method does not terminate finitely, and in the second we consider the convergence behavior for the outer `for` loop of Algorithm 2.2. For convenience in the first subsection, we define T_1, T_2, and T_3 as the sets of iteration indices during which Termination Tests 1, 2, and 3 are satisfied, respectively.

3.1. Global convergence for a barrier subproblem.

We make the following assumption for the solution of a given barrier subproblem (2.1). In the assumption and throughout the remainder of our analysis, W_k refers exclusively to the value of this matrix used to compute the search direction (d_k, δ_k) (i.e., the value of W_k after all perturbations via the Hessian modification strategy have been made).

Assumption 3.3. The infinite sequence $\{(z_k, \lambda_k)\}$ generated by Algorithm 2.2 is contained in a convex set over which the functions f, $c_{\mathcal{E}}$, and $c_{\mathcal{I}}$ and their first derivatives are bounded and Lipschitz continuous. The sequence $\{W_k\}$ is also bounded over all k.

The theorem we prove at the end of this subsection is consistent with Theorem 3.3 in [16].

THEOREM 3.4. *If all limit points of* $\{A_k\}$ *have full row rank, then* $\{\pi_k\}$ *is bounded and*

$$\lim_{k\to\infty}\left\|\begin{bmatrix}\gamma_k + A_k^T\lambda_{k+1}\\ c_k\end{bmatrix}\right\| = 0. \tag{3.3}$$

Otherwise,

$$\lim_{k\to\infty}\|A_k^T c_k\| = 0, \tag{3.4}$$

and if $\{\pi_k\}$ *is bounded, then*

$$\lim_{k\to\infty}\left\|\begin{bmatrix}\gamma_k + A_k^T\lambda_{k+1}\\ A_k^T c_k\end{bmatrix}\right\| = 0. \tag{3.5}$$

This theorem states that all limit points of the algorithm are either feasible or stationary points of the feasibility measure; see (2.17). If all limit points are feasible and satisfy the linear independence constraint qualification (LICQ) for (1.1), then the limit points of $\{A_k\}$ have full row rank and (3.3) holds.

Our first result illustrates that our local model m_k of the penalty function ϕ is sufficiently accurate in a neighborhood of z_k along directions of the type that are computed in the algorithm.

LEMMA 3.5. *There exists* $\xi_1 > 0$ *such that for all* $d \in \mathbb{R}^n$ *and* $\alpha \in (0,1]$ *with* $\alpha d^s \geq -\eta_1$ *we have*

$$\phi(z + \alpha\widetilde{d};\mu,\pi) - \phi(z;\mu,\pi) \leq -\alpha\Delta m(d;\mu,\pi) + \xi_1\pi\alpha^2\|d\|^2. \tag{3.6}$$

Proof. For any scalars ξ and ξ' satisfying $\xi > 0$ and $\xi' \geq -\eta_1\xi$, we have

$$\left|\ln(\xi+\xi') - \ln\xi - \frac{\xi'}{\xi}\right| \leq \sup_{\xi''\in[\xi,\xi+\xi']}\left|\frac{\xi'}{\xi''} - \frac{\xi'}{\xi}\right| = \frac{\xi}{\xi+\xi'}\left(\frac{\xi'}{\xi}\right)^2 \leq \frac{1}{1-\eta_1}\left(\frac{\xi'}{\xi}\right)^2.$$

Under Assumption 3.3, the Lipschitz continuity of $\nabla f(x)$ and $A(z)$ then implies that for some constant $\xi_1 > 0$ we have

$$\begin{aligned}
&\phi(z + \alpha\widetilde{d};\mu,\pi) - \phi(z;\mu,\pi)\\
&= f(x+\alpha d^x) - f(x) - \mu\sum_{i=1}^n \ln(s + \alpha S d^s)^{(i)} + \mu\sum_{i=1}^n \ln s^{(i)}\\
&\quad + \pi\left(\left\|\begin{bmatrix} c_{\mathcal{E}}(x+\alpha d^x)\\ c_{\mathcal{I}}(x+\alpha d^x) - (s+\alpha S d^s)\end{bmatrix}\right\| - \left\|\begin{bmatrix} c_{\mathcal{E}}(x)\\ c_{\mathcal{I}}(x) - s\end{bmatrix}\right\|\right)\\
&\leq \alpha\nabla f(x)^T d^x - \alpha\mu d^s\\
&\quad + \pi\left(\left\|\begin{bmatrix} c_{\mathcal{E}}(x) + \alpha\nabla c_{\mathcal{E}}(x)^T d^x\\ c_{\mathcal{I}}(x) + \alpha\nabla c_{\mathcal{I}}(x)^T d^x - (s+\alpha S d^s)\end{bmatrix}\right\| - \left\|\begin{bmatrix} c_{\mathcal{E}}(x)\\ c_{\mathcal{I}}(x) - s\end{bmatrix}\right\|\right) + \xi_1\pi\alpha^2\|d\|^2\\
&= \alpha\gamma^T d + \pi(\|c(z) + \alpha A(z)d\| - \|c(z)\|) + \xi_1\pi\alpha^2\|d\|^2\\
&= \alpha\gamma^T d + \pi(\|(1-\alpha)c(z) + \alpha(c(z) + A(z)d)\| - \|c(z)\|) + \xi_1\pi\alpha^2\|d\|^2\\
&\leq \alpha\left(\gamma^T d - \pi(\|c(z)\| - \|c(z) + A(z)d\|)\right) + \xi_1\pi\alpha^2\|d\|^2,
\end{aligned}$$

which is (3.6). ∎

Although the algorithm makes use of the penalty function ϕ, it will be convenient in part of our analysis to work with the scaled and shifted penalty function

$$\hat{\phi}(z;\mu,\pi) := \tfrac{1}{\pi}(\varphi(z;\mu)-\chi) + \|c(z)\|, \tag{3.7}$$

where χ is a given constant. A useful property of the function $\hat{\phi}$ for a particular value of χ is the subject of the next lemma.

LEMMA 3.6. *The sequence $\{s_k\}$ is bounded, and so $\{\phi_k\}$ is bounded below. Moreover, with χ in* (3.7) *set as the infimum of φ over all k, we find that for each k we have*

$$\hat{\phi}(z_k;\mu_j,\pi_k) \le \hat{\phi}(z_{k-1};\mu_j,\pi_{k-1}) - \tfrac{1}{\pi_{k-1}}\eta_2\alpha_{k-1}\Delta m_{k-1}(d_{k-1};\mu_j,\pi_{k-1}), \tag{3.8}$$

so $\{\hat{\phi}(z_k;\mu_j,\pi_k)\}$ is monotonically decreasing.

Proof. By (2.29) and the fact that the slack reset (see (2.16)) decreases only ϕ, it follows that

$$\hat{\phi}(z_k;\mu_j,\pi_{k-1}) \le \hat{\phi}(z_{k-1};\mu_j,\pi_{k-1}) - \tfrac{1}{\pi_{k-1}}\eta_2\alpha_{k-1}\Delta m_{k-1}(d_{k-1};\mu_j,\pi_{k-1}),$$

which implies

$$\begin{aligned}\hat{\phi}(z_k;\mu_j,\pi_k) \le{}& \hat{\phi}(z_{k-1};\mu_j,\pi_{k-1})\\ &+\left(\tfrac{1}{\pi_k}-\tfrac{1}{\pi_{k-1}}\right)(\varphi(z_k;\mu_j)-\chi) - \tfrac{1}{\pi_{k-1}}\eta_2\alpha_{k-1}\Delta m_{k-1}(d_{k-1};\mu_j,\pi_{k-1}).\end{aligned} \tag{3.9}$$

Let ξ be an upper bound for $-f$ and $\|c_{\mathcal{I}}\|$, the existence of which follows from Assumption 3.3. By (3.9), the fact that $\{\pi_k\}$ is monotonically nondecreasing, the inequalities

$$\sum_{i=1}^{q}\ln s_k^{(i)} \le q\ln\|s_k\|_\infty \le q\ln\|s_k\|, \tag{3.10}$$

and the nonnegativity of Δm_k, we then have that

$$\hat{\phi}(z_k;\mu_j,\pi_k) \le \hat{\phi}(z_0;\mu_j,\pi_0) + \left(\tfrac{1}{\pi_0}-\tfrac{1}{\pi_k}\right)(\xi+\chi+\mu_j q\max_{0\le l\le k}\ln\|s_l\|). \tag{3.11}$$

On the other hand, from the definition of $\hat{\phi}$ and (3.10) we have that for any k,

$$\hat{\phi}(z_k;\mu_j,\pi_k) \ge -\tfrac{1}{\pi_k}(\xi+\chi+\mu_j q\ln\|s_k\|) + \|s_k\| - \xi, \tag{3.12}$$

since $\|c(z_k)\| \ge \|c_{\mathcal{I}}(x_k)-s_k\| \ge \|s_k\|-\xi$. Consider the indices l_i such that $\|s_{l_i}\| = \max_{k\le l_i}\|s_k\|$. Combining (3.11) and (3.12) for k given by any such l_i, we then obtain

$$-\tfrac{1}{\pi_{l_i}}(\xi+\chi+\mu_j q\ln\|s_{l_i}\|)+\|s_{l_i}\|-\xi \le \hat{\phi}(z_0;\mu_j,\pi_0)+\left(\tfrac{1}{\pi_0}-\tfrac{1}{\pi_{l_i}}\right)(\xi+\chi+\mu_j q\ln\|s_{l_i}\|),$$

which implies

$$\|s_{l_i}\| \le \hat{\phi}(z_0;\mu_j,\pi_0)+\xi+\tfrac{1}{\pi_0}(\xi+\chi+\mu_j q\ln\|s_{l_i}\|).$$

Since the ratio $(\ln\|s\|)/\|s\|$ tends to zero as $\|s\|\to\infty$, this relation implies that $\{s_{l_i}\}$ must be bounded. By the definition of the indices l_i we conclude that the entire sequence $\{s_k\}$ is bounded.

For the second part of the lemma, we first note that if $k \in T_2$, then $d_k = 0$, the model reduction is $\Delta m_k(d_k; \mu_j, \pi_k) = 0$, and $\hat{\phi}(z_{k+1}; \mu_j, \pi_{k+1}) = \hat{\phi}(z_k; \mu_j, \pi_k)$, and so (3.8) follows trivially. Otherwise, with χ chosen as the infimum of φ over all k, it follows from (3.9) and the fact that π_k is nondecreasing that (3.8) holds. ∎

We now state two results, proved as Lemmas 3.6–3.7 and 3.9–3.10 in [16]. The algorithmic components that contribute to these results (namely, the normal component condition (2.18), the tangential component conditions (2.22) and (2.23), and the model reduction condition (2.24)) are identical in [16]. Furthermore, the primary assumption required in [16] to derive these results corresponds in our setting to the assumption that the quantities in the primal-dual matrix and right-hand side vector in (2.7) are uniformly bounded. Indeed, we have these conditions here under Assumption 3.3 since $\{s_k\}$ is bounded in light of Lemma 3.6.

The first result relates to the norms of the normal and tangential components computed in the algorithm.

LEMMA 3.7. *There exists $\xi_2 > 0$ such that, for all k,*

$$\xi_2 \|A_k^T c_k\|^2 \le \|v_k\| \le \omega \|A_k^T c_k\|, \tag{3.13}$$

and hence v_k is bounded in norm over all k. Moreover, u_k is bounded in norm over all k, so together these results imply that d_k is bounded in norm over all k.

The second result provides related bounds for the model reduction Δm_k and the primal step component.

LEMMA 3.8. *There exists $\xi_3 > 0$ such that, for all $k \notin T_2$, we have*

$$\Delta m_k(d_k; \mu_j, \pi_k) \ge \xi_3 \left(\|u_k\|^2 + \pi_k \|A_k^T c_k\|^2 \right).$$

Similarly, there exists $\xi_4 > 0$ such that, for all k, we have

$$\|d_k\|^2 \le \xi_4 \left(\|u_k\|^2 + \max\{1, \pi_k\} \|A_k^T c_k\|^2 \right). \tag{3.14}$$

The limit (3.4) now follows from the above results.

LEMMA 3.9. *The sequence $\{z_k\}$ yields*

$$\lim_{k \to \infty} \|A_k^T c_k\| = 0.$$

Proof. Consider an arbitrary constant $\xi > 0$ and define the set

$$\Xi = \{z : \xi \le \|A(z)^T c(z)\|\}. \tag{3.15}$$

We prove the result by showing that there can only be a finite number of iterates $z_k \in \Xi$. Since ξ is chosen arbitrarily, this will prove the result.

We first show that if the algorithm computes an iterate in the set Ξ, then we eventually have an iteration not in the set T_2. By contradiction, suppose that there exists an iteration number $k' \ge 0$ such that $z_{k'} \in \Xi$ and for all $k \ge k'$ we have $k \in T_2$. Then, since $d_k = 0$ for $k \in T_2$, we have $z_k = z_{k'}$ for all $k \ge k'$ and the inequalities (2.26) and (2.30) yield

$$\|\gamma_{k+1} + A_{k+1}^T \lambda_{k+1}\| \le \|\gamma_k + A_k^T (\lambda_k + \delta_k)\| \le \kappa \|\gamma_k + A_k^T \lambda_k\|.$$

Thus, since $\kappa \in (0,1)$, we have the limit

$$\|\gamma_k + A_k^T \lambda_k\| \to 0. \tag{3.16}$$

On the other hand, by the conditions of Termination Test 2 and the fact that $z_{k'} \in \Xi$, we have that for all $k \geq k'$,

$$0 < \xi \leq \|A_{k'}^T c_{k'}\| = \|A_k^T c_k\| \leq \epsilon_2 \|\gamma_k + A_k^T \lambda_k\|,$$

which contradicts (3.16). Therefore, there exists $k'' \geq k'$ such that $k'' \notin T_2$. By choosing the first such iterate, we have $\|A_{k''}^T c_{k''}\| = \|A_{k'}^T c_{k'}\|$, and thus we have proven the existence of $z_{k''} \in \Xi$ with $k'' \notin T_2$.

Now consider $z_k \in \Xi$ with $k \notin T_2$. By Lemma 3.7 we have

$$\|v_k\| \geq \xi_2 \|A_k^T c_k\|^2 \geq \xi_2 \xi^2,$$

and we may define

$$u_{\sup}^{\Xi} := \sup\{\|u_k\| : z_k \in \Xi\} < \infty,$$

so together we have

$$\|u_k\| \leq \left(u_{\sup}^{\Xi}/(\xi_2\xi^2)\right) \|v_k\|. \tag{3.17}$$

Lemmas 3.7 and 3.8 then imply that there exists a constant $\xi' > 0$ yielding

$$\begin{aligned} \Delta m_k(d_k;\mu_j,\pi_k) &\geq \xi_3 \pi_k \|A_k^T c_k\|^2 && (3.18)\\ &\geq \xi_3 \pi_k \frac{1}{\omega^2} \|v_k\|^2 \\ &\geq \xi' \pi_k \|d_k\|^2, && (3.19) \end{aligned}$$

where the last inequality follows from (3.17). The steplength coefficient α_k can now be bounded in the following manner. First, we note that the fraction-to-the-boundary rule (2.13) is satisfied for any $\alpha_k \leq \eta_1/\|d_k^s\| \leq \eta_1/d_{\sup}$, where $\|d_k^s\| \leq d_{\sup}$ for some constant $d_{\sup} > 0$ independent of k whose existence follows from Lemma 3.7. Moreover, if the line search condition (2.29) does not hold for some $\bar{\alpha} \in (0, \eta_1/d_{\sup})$, then

$$\phi(z_k + \bar{\alpha}\widetilde{d}_k;\mu_j,\pi_k) - \phi(z_k;\mu_j,\pi_k) > -\eta_2\bar{\alpha}\Delta m_k(d_k;\mu_j,\pi_k), \tag{3.20}$$

which, along with Lemma 3.5, yields

$$(1-\eta_2)\Delta m_k(d_k;\mu_j,\pi_k) < \xi_1 \bar{\alpha} \pi_k \|d_k\|^2. \tag{3.21}$$

This inequality and (3.19) implies

$$(1-\eta_2)\xi' \pi_k \|d_k\|^2 < \xi_1 \bar{\alpha} \pi_k \|d_k\|^2,$$

so $\alpha_k \geq \alpha_{\inf}^{\Xi} > 0$, where $\alpha_{\inf}^{\Xi} := (1/2)\min\{\eta_1/d_{\sup}, (1-\eta_2)\xi'/\xi_1\}$ is a constant independent of k.

Suppose that there is an infinite number of iterations with $z_k \in \Xi$. For those $z_k \in \Xi$, Lemma 3.6, (3.15), and (3.18) imply that for each such iterate we have

$$\begin{aligned} \hat{\phi}(z_{k+1};\mu_j,\pi_{k+1}) &\leq \hat{\phi}(z_k;\mu_j,\pi_k) - \tfrac{1}{\pi_k}\eta_2\alpha_k\Delta m_k(d_k;\mu_j,\pi_k) \\ &\leq \hat{\phi}(z_k;\mu_j,\pi_k) - \eta_2\alpha_{\inf}^{\Xi}\xi_3\|A_k^T c_k\|^2 \\ &\leq \hat{\phi}(z_k;\mu_j,\pi_k) - \eta_2\alpha_{\inf}^{\Xi}\xi_3\xi^2, && (3.22) \end{aligned}$$

so $\hat{\phi}$ is reduced by at least a positive constant amount. However, this contradicts the fact that by Lemma 3.6, $\hat{\phi}$ is bounded below and monotonically decreasing, which means that there can only be a finite number of iterates with $z_k \in \Xi$. ∎

The first statement in Theorem 3.4 inspires special consideration for situations where all limit points of the sequence $\{A_k\}$ produced by Algorithm 2.2 for a given μ_j have full row rank. This next result, proved as Corollary 3.14 and Lemma 3.16 in [16], highlights important instances when we can expect the penalty parameter to remain bounded. Again, the results in [16] carry over to our setting since the step computation in the two algorithms is identical and because the quantities in the primal-dual matrix and right-hand side vector in (2.7) are uniformly bounded under Assumption 3.3 and by Lemma 3.6.

LEMMA 3.10. *Suppose that there exists $k_A \geq 0$ such that the smallest singular values of $\{A_k\}_{k \geq k_A}$ are bounded away from zero. Then the sequence $\{z_k\}$ yields*

$$\lim_{k\to\infty} \|c_k\| = 0 \tag{3.23}$$

and $\pi_k = \bar{\pi}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq k_A$ and $\bar{\pi} < \infty$.

In fact, there may be other situations when the sequence of penalty parameter values remains constant after a given iteration, even if all limit points of $\{A_k\}$ do not have full row rank. Therefore, in the next result we state that for the general case when the penalty parameter becomes constant, the sequence of steplength coefficients will remain bounded away from zero.

LEMMA 3.11. *If $\pi_k = \bar{\pi}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq 0$ and $\bar{\pi} < \infty$, then the sequence $\{\alpha_k\}$ is bounded away from zero.*

Proof. For $k \in T_2$, Algorithm 2.2 yields $\alpha_k = 1$, so we need only consider cases where $k \notin T_2$.

For $k \notin T_2$, the fraction-to-the-boundary rule (2.13) is satisfied for all $\alpha_k \leq \eta_1/\|d_k^s\| \leq \eta_1/d_{\sup}$, where $\|d_k^s\| \leq d_{\sup}$ for some constant $d_{\sup} > 0$ independent of k whose existence follows from Lemma 3.7. Then, as in the proof of Lemma 3.9 (see (3.20)–(3.21)), we have that if (2.29) fails for $\bar{\alpha} \in (0, \eta_1/d_{\sup})$, then

$$(1-\eta_2)\Delta m_k(d_k; \mu_j, \pi_k) < \xi_1 \bar{\alpha} \pi_k \|d_k\|^2.$$

Lemma 3.8 then yields

$$(1-\eta_2)\xi_3 \left(\|u_k\|^2 + \pi_k \|A_k^T c_k\|^2\right) < \xi_1 \xi_4 \bar{\alpha} \pi_k \left(\|u_k\|^2 + \max\{1, \pi_k\} \|A_k^T c_k\|^2\right),$$

so

$$\bar{\alpha} > \frac{(1-\eta_2)\xi_3 \left(\|u_k\|^2 + \pi_k \|A_k^T c_k\|^2\right)}{\xi_1 \xi_4 \pi_k \left(\|u_k\|^2 + \max\{1, \pi_k\} \|A_k^T c_k\|^2\right)} \geq \alpha_{\inf},$$

where $\alpha_{\inf} > 0$ is some constant bounded away from zero whose existence follows from the fact that $0 < \pi_{-1} \leq \pi_k \leq \bar{\pi}$. Thus, if $\pi_k = \bar{\pi}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq 0$ and $\bar{\pi} < \infty$, then α_k for $k \notin T_2$ need never be set below $(1/2)\min\{\eta_1/d_{\sup}, \alpha_{\inf}\}$ for (2.29) to be satisfied. $\square$

The previous result ensures that sufficient progress toward a solution point will be made along each nonzero search direction d_k computed in the algorithm. As a consequence, the following lemma, proved as Lemmas 3.18 and 3.19 in [16], states that if the penalty function settles with a particular value of the penalty parameter, the primal step components vanish and dual feasibility is guaranteed. In terms of the equations and results provided in this paper, the proofs in [16] make extensive use of the dual residual condition (2.21), condition (2.30), and Lemma 3.8. Sufficient decreases in the penalty function in the line search, along with Assumption 3.3, produce vanishing primal steps, and sufficiently accurate solutions to (2.10) in the neighborhood of solution points provide dual feasibility in the limit.

Lemma 3.12. *If $\pi_k = \bar{\pi}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq 0$ and $\bar{\pi} < \infty$, then*

$$\lim_{k\to\infty} \|d_k\| = 0$$

and

$$\lim_{k\to\infty} \|\gamma_k + A_k^T \lambda_{k+1}\| = 0.$$

We are now ready to prove the main result stated at the beginning of this section.

Proof of Theorem 3.4. If all limit points of $\{A_k\}$ have full row rank, then there exists $k_A \geq 0$ such that the smallest singular values of $\{A_k\}_{k\geq k_A}$ are bounded away from zero. By Lemma 3.10 we then have that the penalty parameter is constant for all k sufficiently large, which means $\{\pi_k\}$ is bounded, and so by Lemmas 3.10 and 3.12 we have the limit (3.3). On the other hand, if a limit point of $\{A_k\}$ does not have full row rank, then we have the limit (3.4) by Lemma 3.9. Moreover, if $\{\pi_k\}$ is bounded, then the fact that if Algorithm 2.2 increases π_k, it does so by at least δ_π and implies that the penalty parameter is in fact constant for all k sufficiently large. This implies with Lemma 3.12 that we have the limit (3.5). ▯

We reiterate here some key features of our algorithm that have allowed us to extend so many of the results presented in [16] for equality constrained optimization in order to prove Theorem 3.4. First, the inclusion of a slack reset to ensure $s_k \geq 0$ and $c_{\mathcal{I}}(x_k) - s_k \leq 0$ have allowed us to require only $A_k^T c_k \to 0$ to state that the algorithm converges to a feasible point or at least to an infeasible stationary point of problem (1.1). Without such a slack reset, the algorithm may stall at a point with $s_k \geq 0$, $A_k^T c_k = 0$, and $c_{\mathcal{I}}^{(i)}(x_k) - s_k^{(i)} > 0$ for some i, which is not a stationary point for the feasibility measure in (1.2). The second critical component of our approach was the use of scaled derivatives in our definition of the primal-dual system (2.10). Naturally, in an implementation an equivalent system can be defined without such a scaling, but it is important to note that our termination tests *must* be enforced with this scaling in order for our global convergence results to apply.

3.2. Global convergence for the nonlinear program.

We close this section by proving a result related to the overall global convergence of Algorithm 2.2. In the following theorem, we suppose that for any j, the inner `for` loop terminates when

$$\|\nabla f(x_k) + \nabla c_{\mathcal{E}}(x_k)\lambda_{\mathcal{E},k+1} + \nabla c_{\mathcal{I}}(x_k)\lambda_{\mathcal{I},k+1}\|_\infty \leq \varepsilon\mu_j, \tag{3.24a}$$

$$\|S_k\lambda_{\mathcal{I},k+1} + \mu_j e\|_\infty \leq \varepsilon\mu_j, \tag{3.24b}$$

$$\text{and} \quad \|c_k\|_\infty \leq \varepsilon\mu_j \tag{3.24c}$$

for some constant $\varepsilon \in (0,1)$, yielding the outer iterate sequence $\{(z_j, \lambda_j)\}$. Note that if the algorithm terminates finitely with (3.2) satisfied, we set $\lambda_{k+1} \leftarrow \lambda_k$ so that (3.24) holds.

Theorem 3.13. *Suppose that Assumptions* 3.1 *and* 3.3 *hold and define $\{\mu_j\}$ to be a sequence of positive constants such that $\{\mu_j\} \to 0$. Algorithm* 2.2 *then yields one of the following outcomes:*

(i) *During outer iteration j,* (3.24c) *is never satisfied, in which case the stationarity condition* (2.4) *for the infeasibility problem* (1.2) *is satisfied in the limit.*

(ii) *During outer iteration j, there exists an infinite subsequence of inner iterates where* (3.24c) *is satisfied but* (3.24a) *or* (3.24b) *(or both) is not, in which case the stationarity condition* (2.4) *for the feasibility problem* (1.2) *is satisfied in the limit and $\{\pi_k\} \to \infty$.*

(iii) *Each outer iteration results in an iterate* $\{(z_j, \lambda_j)\}$ *satisfying* (3.24), *in which case all limit points of* $\{x_j\}$ *are feasible, and if a limit point* $\bar{x}$ *of* $\{x_j\}$ *satisfies the LICQ, the first-order optimality conditions of* (1.1) *hold at* $\bar{x}$*; i.e., there exists* $\bar{\lambda}$ *such that*

$$\nabla f(\bar{x}) + \nabla c_{\mathcal{E}}(\bar{x})\bar{\lambda}_{\mathcal{E}} + \nabla c_{\mathcal{I}}(\bar{x})\bar{\lambda}_{\mathcal{I}} = 0, \tag{3.25a}$$

$$c_{\mathcal{E}}(\bar{x}) = 0, \tag{3.25b}$$

$$c_{\mathcal{I}}(\bar{x}) \geq 0, \tag{3.25c}$$

$$\bar{\lambda}_{\mathcal{I}} \leq 0, \tag{3.25d}$$

$$\bar{\lambda}^{(i)} c_{\mathcal{I}}^{(i)}(\bar{x}) = 0, \quad i = 1, \ldots, q. \tag{3.25e}$$

Proof. If (3.24c) is never satisfied during outer iteration j, then we know by Theorem 3.4 that the limit (3.4) holds, which means that (2.4) is satisfied in the limit. This corresponds to situation (i). Further, by Theorem 3.4, we have that if (3.24a) or (3.24b) (or both) is not satisfied for an infinite subsequence of iterates, then $\{\pi_k\} \to \infty$, which along with (3.4) completes the proof of situation (ii).

The remaining possibility is that (3.24) is eventually satisfied during each outer iteration $j \geq 0$. Define an infinite subsequence of indices j_l such that $x_{j_l} \to \bar{x}$ as $l \to \infty$. Since $s_{j_l} \geq 0$ and $c(z_{j_l}) = (c_{\mathcal{E}}(x_{j_l}), c_{\mathcal{I}}(x_{j_l}) - s_{j_l}) \to 0$, we have $c_{\mathcal{E}}(\bar{x}) = 0$, $s_{j_l} \to \bar{s} = c_{\mathcal{I}}(\bar{x})$ as $l \to \infty$, and $c_{\mathcal{I}}(\bar{x}) \geq 0$. Therefore, all limit points of $\{x_j\}$ are feasible, and in particular (3.25b) and (3.25c) hold.

Now define $\widehat{\mathcal{I}} = \{i : c_{\mathcal{I}}^{(i)}(\bar{x}) = 0\}$. By (3.24b), we have $S_j \lambda_{\mathcal{I},j} \to 0$, which means that $\lambda_{\mathcal{I},j_l}^{(i)} \to 0$ for $i \notin \widehat{\mathcal{I}}$. Along with (3.24a), this implies

$$\nabla f(x_{j_l}) + \nabla c_{\mathcal{E}}(x_{j_l})\lambda_{\mathcal{E},j_l} + \sum_{i \in \widehat{\mathcal{I}}} \lambda_{\mathcal{I},j_l}^{(i)} \nabla c_{\mathcal{I}}^{(i)}(x_{j_l}) \to 0. \tag{3.26}$$

The inequality (3.24b) also yields

$$s_j^{(i)} \lambda_{\mathcal{I},j}^{(i)} \leq (\varepsilon - 1)\mu_j < 0 \tag{3.27}$$

for $i = 1, \ldots, q$, which along with $s_j \geq 0$ implies that $\lambda_{\mathcal{I},j} \leq 0$ for all j. By LICQ, the columns of $\nabla c_{\mathcal{E}}(\bar{x})$ and the vectors $\nabla c_{\mathcal{I}}^{(i)}(\bar{x})$ for $i \in \widehat{\mathcal{I}}$ form a linearly independent set, so (3.26) and (3.27) imply that the sequence $\{\lambda_{j_l}\}$ converges to some value $\bar{\lambda}$ satisfying (3.25a) and (3.25d). ∎

4. Numerical experiments.

We have implemented our algorithm in the `IPOPT` optimization package [47] version 3.7.0,[1] tied with the iterative linear system solver and preconditioners implemented in the `PARDISO` software package [41] version 3.4.[2] In this section, we describe some details of our code, illustrate that it is robust on large nonlinear optimization test sets, and illustrate its efficiency on a pair of PDE-constrained model problems.

4.1. Implementation details.

The default `IPOPT` code implements the algorithm described in [47]. In order to test the method described in this paper, a few modifications were made.

We augmented the `IPOPT` code to include the option to obtain the search direction by means of inexact normal and primal-dual step computations. Rather than scale

[1] http://www.coin-or.org/Ipopt/.
[2] http://www.pardiso-project.org/.

the derivative matrices in order to set up the primal-dual system (2.10), however, we solve an equivalent system with unscaled derivatives (as already implemented in IPOPT) and scale the search direction for the slack variables in order to apply our termination tests. We also replace the scaling matrix S_k with $\hat{S}_k = \text{diag}(\hat{s}_k)$ in (2.8), where $\hat{s}_k^{(i)} = \min\{100, s_k\}$. That is, we scale only elements in d_k^s corresponding to small slacks when forming $\widetilde{d}_k^s$. This change does not effect our theoretical results but has the advantage that inefficiencies or numerical problems due to large slack variables can be avoided.

In an unscaled system, the (2,2)-block in (2.6) appears as $S_k^{-1}\Sigma_k S_k^{-1}$. In our implementation, we initially choose Σ_k so that the resulting matrix is $S_k^{-1}Y_k$, where y_k are dual variables corresponding to the slack bounds. However, since this choice may mean that Assumption 3.3 is not satisfied, we perturb y_k so that

$$\overline{\upsilon}\mu S_k^{-2} \succeq S_k^{-1}Y_k \succeq \underline{\upsilon}\mu S_k^{-2}$$

for constants $\overline{\upsilon} \geq 1 \geq \underline{\upsilon} > 0$. This ensures that the matrix does not deviate too much from μS_k^{-2} (i.e., the choice $\Sigma_k = I$), which is consistent with the original IPOPT implementation. For the multipliers, we initialize $\lambda_0 \leftarrow 0$ and $y_0 \leftarrow 10^{-4}$.

Finally, IPOPT's default filter line-search procedure is replaced by the backtracking line-search described in Algorithm 2.2 using the exact penalty function (2.14), including the slack reset (2.16). Moreover, since we observed in some cases that the penalty parameter was set to a high value in the early stages of the optimization that hampered progress later on, we employ the flexible penalty method described in [15] since it proved to be more efficient in our tests. This mechanism has a similar effect as a standard penalty function, so we claim that the use of this tool does not affect our convergence theory.

Next we describe some of the details of the normal and primal-dual step computations of Algorithm 2.2. Each is performed by applying the symmetric quasi-minimum residual (SQMR) method [21, 22] as implemented in PARDISO to a large symmetric indefinite linear system.

As previously mentioned, a conjugate gradient or the LSQR method can be used to compute the normal component v_k. However, we found this approach to perform poorly for ill-conditioned Jacobians. Furthermore, it is not clear how to precondition the underdetermined problem (2.9). Therefore, we chose to apply an inexact dogleg approach; see [36, 37]. We begin by computing the Cauchy point $v_k^C = \bar{\alpha}_k \bar{v}_k$ (see the normal component condition (2.18)) and then (approximately) solve the *augmented system* (e.g., see [14])

$$\begin{bmatrix} I & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} v_k^N \\ \delta_k^N \end{bmatrix} = - \begin{bmatrix} 0 \\ c_k \end{bmatrix}, \tag{4.1}$$

which for an inexact solution yields the residual vector

$$\begin{bmatrix} \rho_k^N \\ r_k^N \end{bmatrix} = \begin{bmatrix} I & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} v_k^N \\ \delta_k^N \end{bmatrix} + \begin{bmatrix} 0 \\ c_k \end{bmatrix}. \tag{4.2}$$

Note that an exact solution of (4.1) gives the least-norm solution of (2.9) with $\omega = \infty$. The inexact dogleg step is then defined as a point along the line segment between the Cauchy point and v_k^N that is feasible for problem (2.9) and satisfies the Cauchy decrease condition (2.18); i.e., it satisfies our normal component condition. We tailored this approach into an implementation that has worked well in our tests. For example, we consider the fraction-to-the-boundary rule (2.13) when choosing between the

Cauchy and inexact dogleg steps. A detailed description of the normal step computation is provided as Algorithm 4.1.

Algorithm 4.1 Normal Step Computation.

```
Given parameters $0 < \tilde{\epsilon}_v, \kappa_v < 1$ and $l^{\rm n}_{\max}, \omega > 0$.
Compute the Cauchy point $v_k^C = \bar{\alpha}_k \bar{v}_k$, where $\bar{v}_k = -A_k^T c_k$ and $\bar{\alpha}_k$ solves (2.19)
Initialize $v_k \leftarrow v_k^C$
for $l = 0, 1, 2, \ldots, l^{\rm n}_{\max}$ do
    Perform an SQMR iteration on (4.1) to compute $(v^{N,l}, \delta^{N,l})$
    if $\|(\rho^{N,l}, r^{N,l}\|) \leq \kappa_v \|c_k\|$ and $\|c_k + A_k v^{N,l}\| \leq \|c_k + A_k v_k^C\|$ then
        break
    end if
end for
Set $v_k^N \leftarrow v^{N,l}$
Set $v_k^D = (1-\alpha) v_k^C + \alpha v_k^N$, where $\alpha \in [0,1]$ is the largest value such that $v_k^D$ is
feasible for (2.9)
Set $\alpha_k^C$ and $\alpha_k^D$ as the largest values in $[0,1]$ satisfying (2.13) along $v_k^C$ and $v_k^D$,
respectively
if $(\|c_k\| - \|c_k + \alpha_k^D A_k v_k^D\|) \geq \tilde{\epsilon}_v (\|c_k\| - \|c_k + \alpha_k^C A_k v_k^C\|)$ then
    Set $v_k \leftarrow v_k^D$
end if
```

Algorithm 4.2 provides details of the primal-dual step computation. We run the SQMR algorithm until either an acceptable search direction has been computed (i.e., one satisfying at least one of our termination tests) or the Hessian modification strategy indicates that a perturbation to W_k is appropriate. In the latter case, we restart SQMR with the zero vector and continue this process until an acceptable search direction is computed. The form of the Hessian modification is the same as in the default IPOPT algorithm described in [47], Algorithm IC. That is, if a modification is requested in the algorithm, a scalar multiple of the identity matrix is added to the Hessian, where the magnitude of the scaling factor depends on the scalar that was used for the most recent modification. We write the algorithm with an iteration limit $l^{\rm pd}_{\max}$ to indicate that due to numerical errors a solver may fail to satisfy our tests in a finite number

Algorithm 4.2 Primal-dual Step Computation.

```
Choose $\kappa', \tilde{l}^{\rm pd}_{\max}, l^{\rm pd}_{\max} > 0$
Set $l \leftarrow 0$, $(d^0, \delta^0) \leftarrow 0$, and $W_k$ via (2.6)
for $l = 1, 2, \ldots, l^{\rm pd}_{\max}$ do
    Perform a SQMR iteration on (2.10) to compute $(d^l, \delta^l)$
    if $l \geq \tilde{l}^{\rm pd}_{\max}$ or (4.3) holds then
        if Termination Tests 1 or 3 is satisfied then
            break
        else if Termination Test 2 is satisfied then
            Set $d^l \leftarrow 0$ and break
        else if (2.22) and (2.23a) do not hold for $d^l$ then
            Set $l \leftarrow 0$, $(d^0, \delta^0) \leftarrow 0$, and modify $W_k$ to increase its smallest eigenvalue
        end if
    end if
end for
Set $(d_k, \delta_k) \leftarrow (d^l, \delta^l)$
```

Table 4.1
Parameter values used for Algorithm 2.2.

Parameter	Value	Parameter	Value	Parameter	Value
ψ	10^{-1}	κ	0.1	ϵ_2	1
ζ	0.1	ϵ_3	0.99	δ_π	10^{-4}
η_1	$\max\{0.99, 1-\mu_j\}$	τ	0.1	σ	$\tau\epsilon_3$
η_2	10^{-8}	ω	100	μ_0	0.1
ϵ_v	1	θ	$\mu_j 10^{-12}$	π_{-1}	10^{-6}
$\tilde{\epsilon}_v$	0.1	$l^{\rm n}_{\max}$	200	κ_v	10^{-3}
κ'	10^{-3}	$\tilde{l}^{\rm pd}_{\max}$	100	$l^{\rm pd}_{\max}$	500

of iterations, in which case one may simply accept the last inexact solution that is computed. Note, however, that this upper limit was rarely reached in our experiments due to the effectiveness of the preconditioner.

To avoid an unnecessarily large number of outer iterations and to promote fast local convergence, Algorithm 4.2 aims to produce reasonably accurate solutions that satisfy the residual bound

$$\left\| \begin{bmatrix} \rho_k \\ r_k \end{bmatrix} \right\| \leq \kappa' \left\| \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ A_k v_k \end{bmatrix} \right\| \tag{4.3}$$

for a given $\kappa' \in (0,1)$ before consideration of our termination tests. However, if after a fixed number of iterations, $\tilde{l}^{\rm pd}_{\max}$, this condition is not satisfied, then it is dropped and the algorithm requires only that one of Termination Tests 1, 2, or 3 is satisfied.

As preconditioner, both for the augmented system (4.1) and the primal-dual system (2.7), we use the incomplete multilevel factorization solver in `PARDISO` with inverse-based pivoting [9], stabilized by symmetric-weighted matchings; see [40, 42]. For the results in section 4.2, we used a maximum inverse norm factor of 5×10^6 and a dropping tolerance of 0.01 for the Schur complement and the factor. For the results in section 4.3, we used a maximum inverse norm factor of 5×10^6 and a dropping tolerance of 0.5 for the Schur complement and the factor.

As for the input parameters used in our implementation, these quantities are summarized in Table 4.1. We note that we update $\lambda_{k+1} = \lambda_k + \beta_k\delta_k$ by choosing β_k as the smallest value in $[\alpha_k, 1]$ satisfying (2.30), as this choice yields good practical performance in our tests.

All of the results in the following subsections were obtained on a 2GHz Xeon machine running RedHat Linux.

4.2. Numerical results on the CUTEr and COPS collections.

Our first set of numerical experiments illustrates the robustness of our algorithm on a broad range of test problems. This will serve to show that inexact step computations do not significantly impede convergence of the algorithm as long as inexactness is controlled appropriately through our termination tests. It will also provide evidence that our trust-region approach for handling ill-conditioning and our inertia-correction strategy for handling nonconvexity are effective in practice.

We applied our implementation to problems from the CUTEr [24, 25] and COPS [18] collections for which `AMPL` (a modeling language for mathematical programming) [20, 23] models were readily available.[3] Problem (1.1) is a slightly different formulation than the one considered in the original `IPOPT` algorithm, so our implementation does

[3]http://www.orfe.princeton.edu/~rvdb/ampl/nlmodels/cute/index.html, http://www.mcs.anl.gov/~more/cops/.

Table 4.2
Solver results on the CUTEr and COPS collections.

Solver result	Alg. 2.2	IPOPT
Optimal solution found	564	629
Solved to acceptable level	13	11
Infeasible stationary point found	3	4
Maximum iteration limit	35	15
Maximum CPU time limit	38	7
Restoration failure	17	10
Search direction too small	6	1
Iterates diverging	2	2
Error in step computation	0	3
Error in line search	6	0
Invalid function call	0	2

not currently support constraint functions with both lower and upper bounds, as this would require significant alterations to the software. Therefore, problems of this type in the test sets were removed from our list. We also removed problems that are square or contain too few degrees of freedom; i.e., we do not attempt to solve problems with $n \leq p$. The remaining set includes 684 problems, many of which are highly nonlinear, ill-conditioned, and/or otherwise difficult to solve.

Table 4.2 provides the number of problems that terminate with different solver results for both Algorithm 2.2 and the original IPOPT algorithm run in conjunction with, respectively, the iterative and direct linear solvers provided in PARDISO. "Optimal solution found" indicates that an iterate was found satisfying the first-order optimality conditions within the termination tolerance of 10^{-8}. "Solved to acceptable level" means that the code determines that the current iterate is near optimal, yet the termination tolerance was not completely satisfied. "Infeasible stationary point found" means that a stationary point for (1.2) was located within the termination tolerance of 10^{-8}, but the constraint violation was not small. "Maximum iteration limit" and "Maximum CPU time limit" indicate that the iteration limit of 3000 or the CPU time limit of 3 hours was reached, respectively. The algorithm returns "Restoration failure" in Algorithm 2.2 if PARDISO was unable to solve the linear system to a sufficient accuracy to compute an acceptable step, whereas in the original IPOPT algorithm this result means that the feasibility restoration phase failed. "Search direction too small" means that the algorithm computed a series of significantly short search directions, so further progress was not likely. "Iterates diverging" indicates that the primal iterates became extremely large in norm. Finally, "Error in step computation," "Error in line search," and "Invalid function call" mean, respectively, that the Hessian modification became exceedingly large before an acceptable step was computed, there was a failure in the line search, or the algorithm computed a point at which AMPL was not able to return a valid function value.

The results illustrate that the implementation of Algorithm 2.2 is quite robust. Indeed, if we refer to an algorithm as successful if the solver result was "Optimal solution found," "Solved to acceptable level," or "Infeasible stationary point found," then IPOPT was successful 94% and Algorithm 2.2 was successful 85% of the time. In cases where both algorithms were successful, the final objective function values obtained were identical within 1% (or 0.01 if the optimal value was less than one in absolute value), except for 8 cases where Algorithm 2.2 converged to a point with a smaller objective value and 23 cases where IPOPT obtained a better value. This illustrates that, despite the fact that our inertia-correction strategy is only a heuristic and may not compute a step based on a fully convex model, our code is still able

Table 4.3
Problem sizes for instances of Example 4.1.

N	n	p	q	H	J
20	8000	5832	4336	54737	40824
30	27000	21952	10096	186207	153664
40	64000	54872	18256	443077	384104
50	125000	110592	28816	867347	774144
60	216000	195112	41776	1501017	1365784
70	343000	314432	57136	2386087	2201024
80	512000	474552	74896	3564557	3221864

to solve nonconvex problems well. Our approach fails more often, but this is to be expected, as our code is not as mature as `IPOPT` and an increase in iteration count often follows from the use of inexact Newton steps. Furthermore, for some problems in these test sets, the use of an iterative linear solver can actually lead to an increase in computation time, causing the algorithm to run out of the allowed CPU time. Overall, we find these results to be very encouraging.

4.3. Numerical results on PDE-constrained model problems. We also applied our implementation to two PDE-constrained optimal control problems to illustrate its performance on large-scale models. The models were implemented in `MATLAB`. Each problem is solved on the three-dimensional domain $\Omega = [0,1]\times[0,1]\times[0,1]$ over which we use equidistant Cartesian grids with N grid points in each spacial direction and a standard 7-point stencil for discretizing the differential operators. The goal is to find optimal values of the control $u(x)$ and state variables $y(x)$ to fit a target $y_t(x)$; i.e., the objective of each problem is to minimize

$$f(y(x)) = \frac{1}{2}\int_\Omega (y(x) - y_t(x))^2\, dx. \tag{4.4}$$

Our first model is a boundary control problem inspired by Example 5.1 in [32].

Example 4.1. Let the control $u(x)$ be defined on the boundary $\partial\Omega$. Then minimize (4.4) with

$$y_t(x) = 3 + 10x^{(1)}(x^{(1)}-1)x^{(2)}(x^{(2)}-1)\sin(2\pi x^{(3)})$$

subject to $y(x) = u(x)$ on $\partial\Omega$, the differential equation

$$-\nabla(e^{y(x)}\cdot\nabla y(x)) = 20 \quad \text{in } \Omega,$$

and the bounds

$$2.5 \le u(x) \le 3.5 \quad \text{on } \partial\Omega.$$

Table 4.3 provides the problem sizes for instances of Example 4.1 that we solved with Algorithm 2.2. The header N refers to the number of grid points per spacial dimension, which is followed by the number of variables n, equality constraints p, and inequality constraints q. The headers H and J refer to the number of nonzero elements in the Hessian and Jacobian matrices, respectively.

Using $y_0(x) = 3$ and $u_0(x) = 3$ as a starting point, the results for these instances are provided in Table 4.4. We provide the number of iterations required k^*, the total CPU time required (rounded to the nearest second), the average number of iterative linear solver iterations per normal step computation l^{n*} (rounded to the

TABLE 4.4
Numerical results for Algorithm 2.2 *and* IPOPT *applied to Example* 4.1.

	Algorithm 2.2					IPOPT		
N	k^*	CPU	l^{n*}	l^{pd*}	f^*	k^*	CPU	f^*
20	15	11	25	24	1.3368e-2	15	54	1.3368e-2
30	14	74	48	68	1.3039e-2	14	795	1.3039e-2
40	13	518	51	70	1.2924e-2	13	6346	1.2924e-2
50	13	3402	61	164	1.2871e-2	13	33704	1.2871e-2
60	13	10124	83	132	1.2843e-2	---	---	---
70	14	21832	150	210	1.2826e-2	---	---	---
80	14	41804	109	301	1.2815e-2	---	---	---

nearest integer), the average number of iterative linear solver iterations per primal-dual step computation l^{pd*} (rounded to the nearest integer), and the optimal objective value f^*. These results can be compared to some results provided for the original IPOPT algorithm with the direct linear solver in PARDISO, which are also provided in the table. We find that our implementation of Algorithm 2.2 solves all instances of Example 4.1 in a small number of iterations. The gain in efficiency compared to the original IPOPT algorithm is apparent, particularly in the largest case we solved with IPOPT ($N = 50$). Even though IPOPT solves only one linear system per iteration instead of both (4.1) and (2.10), the computation time is significantly higher. This discrepancy will most likely be more pronounced for larger N.

Example 4.1 is nonlinear and nonconvex, but it was solved in few iterations without ever requiring modifications to the Hessian during the primal-dual step computation. In contrast, our second example is a simplified hyperthermia cancer treatment model [33, 42], and it is more challenging. Regional hyperthermia is a cancer therapy that aims at heating large and deeply seated tumors by means of radio wave adsorption, as heating tumors above a temperature of about 41°C results in preferential killing of tumor cells and makes them more susceptible to an accompanying radio or chemotherapy. For designing an optimal therapy, amplitudes and phases of the antennas have to be selected such that the tumor temperature is maximized up to a target therapeutical temperature y_t of 43°C.

Example 4.2. Let the control $u^{(j)} = a^{(j)}e^{i\phi^{(j)}}$ be a complex vector of amplitudes $a \in \mathbb{R}^{10}$ and phases $\phi \in \mathbb{R}^{10}$ of 10 antennas, let $M(x)$ be a 10×10 matrix with $M^{(j,k)}(x) = \langle E^{(j)}(x), E^{(k)}(x)\rangle$, where $E^{(j)}(x) = \sin\left(j \cdot \pi \cdot x^{(1)}x^{(2)}x^{(3)}\right)$, and define the "tumor" to be the central region $\Omega_0 = [3/8, 5/8] \times [3/8, 5/8] \times [3/8, 5/8]$. Then minimize (4.4) with

$$y_t(x) = \begin{cases} 37 & \text{in } \Omega \backslash \Omega_0, \\ 43 & \text{in } \Omega_0 \end{cases}$$

subject to the differential equation

$$-\Delta y(x) - 10(y(x) - 37) - u^* M(x) u = 0 \quad \text{in } \Omega, \tag{4.5}$$

the state variable bounds

$$\begin{aligned} 37.0 \le y(x) \le 37.5 &\quad \text{on } \partial\Omega, \\ 42.0 \le y(x) \le 44.0 &\quad \text{in } \Omega_0, \end{aligned}$$

and the control variable bounds

$$\begin{aligned} -10 \le a \le 10, \\ -4\pi \le \phi \le 4\pi. \end{aligned}$$

Table 4.5
Problem sizes for instances of Example 4.2.

N	n	p	q	H	J
20	8020	5832	4626	54947	157464
30	27020	21952	10822	186417	592704
40	64020	54872	20958	443287	1481544
50	125020	110592	33250	867557	2985984

Table 4.6
Numerical results for Algorithm 2.2 *and* `IPOPT` *applied to Example* 4.1.

	Algorithm 2.2					IPOPT		
N	k^*	CPU	l^{n*}	l^{pd*}	f^*	k^*	CPU	f^*
20	30	87	85	80	2.4073	63	529	2.3571
30	111	3702	179	145	2.3763	81	11683	2.3719
40	75	12963	193	185	2.6459	131	135356	2.6419
50	80	105382	199	186	2.6072	---	---	---

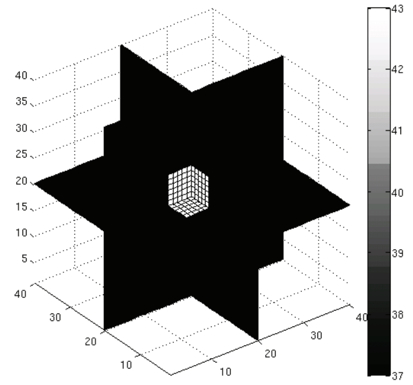

Fig. 4.1. *Target function* $y_t(x)$ *for Example* 4.2 *with* $N = 40$.

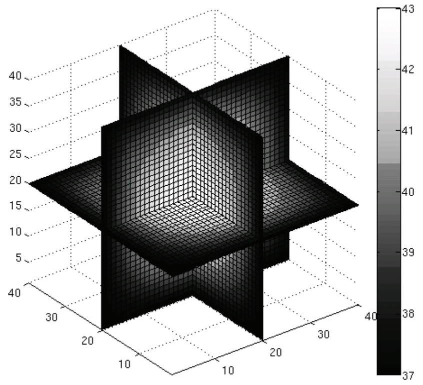

Fig. 4.2. *Solution* $y(x)$ *obtained by Algorithm* 2.2 *for Example* 4.2 *with* $N = 40$.

The PDE (4.5) describes heat dissipation in the tissue, removal of heat via 37°C blood flow, and absorbed energy produced by the 10 microwave antennas. Note that the algorithm uses a and ϕ as control variables (that form the complex vector u) and that the term $u^*M(x)u$ in (4.5) is real-valued. As starting point we selected $y(x) = 40$ for $x \in \Omega$, $a = (1, 0, \ldots, 0)^T$, and $\phi = 0$.

The problem sizes for instances of Example 4.2 that we solved with Algorithm 2.2 are provided in Table 4.5, and the corresponding results are presented in Table 4.6. Again, our algorithm is able to solve the problem in a number of iterations comparable to the original algorithm, while savings in computation time become apparent as N increases. The higher and more variable iteration count shows that this problem is more difficult to solve than Example 4.1, and both algorithms encountered a significant number of iterations in which the Hessian had to be modified. Also, the problem has several local solutions, which explains the differences in optimal values.

For illustrative purposes, we provide plots of the target function y_t and the solution y obtained by Algorithm 2.2 for $N = 40$; see Figures 4.1 and 4.2.

5. Conclusion and final remarks. We have presented and analyzed an interior-point algorithm for large-scale nonlinear nonconvex optimization. By enhancing the algorithm in [16] and extending the corresponding convergence results, we were able to show that the proposed method is globally convergent for generally constrained problems under loose assumptions.

The novel aspects of the approach include our definition of termination tests for iterative linear system solvers. These conditions allow for inexact step computations and therefore offer flexibility to the user in terms of computational expense while providing a theoretical convergence guarantee. Furthermore, the use of iterative linear solvers avoids the requirement to store large derivative matrices explicitly.

The normal step computation, by incorporating a trust region, allows the algorithm to deal with situations where the active constraint gradients are not linearly independent at infeasible limit points. As a consequence, this line-search algorithm can be shown to avoid the convergence problem discussed in [46] and to generate limit points that are stationary points for an infeasibility measure if feasibility is not achieved asymptotically.

An implementation of our algorithm was described and numerical results were presented on two optimization test sets and two large-scale PDE-constrained problems. These experiments illustrate the robustness of our algorithm and the computational advantages it provides as the problem size increases. We showed that our method is comparable with respect to iteration count and solution quality with a state-of-the-art continuous optimization algorithm, and it outperforms the conventional approach in terms of storage and CPU time for the larger problem instances in our tests.

Acknowledgments. We thank Johannes Huber (University of Basel) for writing `MATLAB` problem formulations for our numerical experiments and Peter Carbonetto (University of British Columbia) for contributing his `MATLAB` interface to the `IPOPT` open source project. We also thank the referees, as their comments and suggestions significantly improved the paper.

REFERENCES

[1] V. Arnautu and P. Neittaanmaki, *Optimal Control from Theory to Computer Programs*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

[2] J. T. Betts, *Practical Methods for Optimal Control using Nonlinear Programming*, Adv. Des. Control, SIAM, Philadelphia, 2001.

[3] L. T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. Van Bloemen Waanders, *Real-time PDE-constrained Optimization*, Comput. Sci. Eng., SIAM, Philadelphia, 2007.

[4] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. Van Bloemen Waanders, eds., *Large-scale PDE-constrained Optimization*, Lect. Notes Comput. Sci. Eng. 30, Springer, New York, 2003.

[5] G. Biros and O. Ghattas, *Inexactness issues in the Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization*, in Large-Scale PDE-Constrained Optimization, Lect. Notes Comput. Sci. Eng. 30, L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. Van Bloemen Waanders, eds., Springer, New York, 2003, pp. 93–114.

[6] G. Biros and O. Ghattas, *Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* I: *The Krylov–Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687–713.

[7] G. Biros and O. 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*, SIAM J. Sci. Comput., 27 (2005), pp. 714–739.

[8] M. Bollhöfer, M. J. Grote, and O. Schenk, *Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media*, SIAM J. Sci. Comput., 31 (2009), pp. 3781–3805.

[9] M. Bollhöfer and Y. Saad, *Multilevel preconditioners constructed from inverse-based ILUs*, SIAM J. Sci. Comput., 27 (2006), pp. 1627–1650.

[10] A. Borzi and V. Schulz, *Multigrid methods for PDE optimization*, SIAM Rev., 51 (2009), pp. 361–395.

[11] R. H. Byrd, F. E. Curtis, and J. Nocedal, *An inexact SQP method for equality constrained optimization*, SIAM J. Optim., 19 (2008), pp. 351–369.

[12] R. H. Byrd, F. E. Curtis, and J. Nocedal, *An inexact Newton method for nonconvex equality constrained optimization*, Math. Program., 122 (2010), pp. 273–299.

[13] R. H. Byrd, J.-Ch. Gilbert, and J. Nocedal, *A trust region method based on interior point techniques for nonlinear programming*, Math. Program., 89 (2000), pp. 149–185.

[14] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.

[15] F. E. Curtis and J. Nocedal, *Flexible penalty functions for nonlinear constrained optimization*, IMA J. Numer. Anal., 28 (2008), pp. 749–769.

[16] F. E. Curtis, J. Nocedal, and A. Wächter, *A matrix-free algorithm for equality constrained optimization problems with rank-deficient Jacobians*, SIAM J. Optim., 20 (2009), pp. 1224–1249.

[17] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[18] E. D. Dolan, J. J. Moré, and T. S. Munson, *Benchmarking Optimization Software with COPS* 3.0, Argonne National Laboratory Research report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2004.

[19] M. Fisher, J. Nocedal, Y. Trémolet, and S. J. Wright, *Data assimilation in weather forecasting: A case study in PDE-constrained optimization*, Optim. Eng., 10 (2009), pp. 409–426.

[20] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole, Florence, KY, 2002.

[21] R. W. Freund, *Preconditioning of symmetric, but highly indefinite linear systems*, in 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, A. Sydow, ed., Wissenschaft and Technik, Berlin, 1997, pp. 551–556.

[22] R. W. Freund and F. Jarre, *A QMR-based interior-point algorithm for solving linear programs*, Math. Program., 76 (1997), pp. 183–210.

[23] D. M. Gay, *Hooking Your Solver to AMPL*, Technical report, Computing Sciences Research Center, Bell Laboratories, Murray Hill, NJ, 1997.

[24] N. I. M. Gould, I. Bongartz, A. R. Conn, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[25] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[26] E. Haber and U. M. Ascher, *Preconditioned all-at-once methods for large, sparse parameter estimation problems*, Inverse Problems, 17 (2001), pp. 1847–1864.

[27] M. Heinkenschloss and D. Ridzal, *An inexact trust-region SQP method with applications to PDE-constrained optimization*, in Numerical Mathematics and Advanced Applications: Proceedings of ENUMATH 2007, the 7th European Conference on Numerical Mathematics and Advanced Applications, Graz, Austria, K. Kunisch, G. Of, and O. Steinbach, eds., Springer, 2008, pp. 613–620.

[28] M. Heinkenschloss and L. N. Vicente, *Analysis of inexact trust-region SQP algorithms*, SIAM J. Optim., 12 (2002), pp. 283–302.

[29] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Math. Model. Theory Appl. 23, Springer, Dordrecht, Netherlands, 2009.

[30] H. Jäger and E. W. Sachs, *Global convergence of inexact reduced SQP methods*, Optim. Methods Softw., 7 (1997), pp. 83–110.

[31] D. E. Kirk, *Optimal Control Theory: An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[32] H. Maurer and H. D. Mittelmann, *Optimization techniques for solving elliptic control problems with control and state constraints. Part 1: Boundary control*, Comput. Optim. Appl., 16 (2000), pp. 29–55.

[33] E. Neufeld, *High Resolution Hyperthermia Treatment Planning*, Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, Hartung-Gorre Verlag, NR. 17947, 2008, DOI 10.3929/ethz-a-005743425.

[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., 2nd ed., Springer, New York, 2006.

[35] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[36] R. P. Pawlowski, J. P. Simonis, H. F. Walker, and J. N. Shadid, *Inexact Newton dogleg methods*, SIAM J. Numer. Anal., 46 (2008), pp. 2112–2132.

[37] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London, 1970, pp. 87–114.

[38] E. E. Prudencio, R. H. Byrd, and X.-C. Cai, *Parallel full space SQP Lagrange–Newton–Krylov–Schwarz algorithms for PDE-constrained optimization problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1305–1328.

[39] D. Ridzal, *Trust-region SQP Methods with Inexact Linear System Solves for Large-scale Optimization*, Ph.D. thesis, Rice University, Houston, TX, 2006.

[40] O. Schenk, M. Bollhöfer, and R. A. Römer, *On large-scale diagonalization techniques for the Anderson model of localization*, SIAM Rev., 50 (2008), pp. 91–112.

[41] O. Schenk and K. Gärtner, *On fast factorization pivoting methods for sparse symmetric indefinite systems*, Electron. Trans. Numer. Anal., 23 (2006), pp. 158–179.

[42] O. Schenk, A. Wächter, and M. Weiser, *Inertia-revealing preconditioning for large-scale nonconvex constrained optimization*, SIAM J. Sci. Comput., 31 (2008), pp. 939–960.

[43] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[44] P. L. Toint, *Nonlinear Stepsize Control, Trust Regions and Regularizations for Unconstrained Optimization*, Technical report, Department of Mathematics, FUNDP, University of Namur, Namur, Belgium, 2008.

[45] S. Ulbrich, *Generalized SQP-methods with "Parareal" time-domain decomposition for time-dependent PDE-constrained optimization*, in Real-Time PDE-Constrained Optimization, L. T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, eds., SIAM, Philadelphia, 2007, pp. 145–168.

[46] A. Wächter and L. T. Biegler, *Failure of global convergence for a class of interior point methods for nonlinear programming*, Math. Program., 88 (2000), pp. 565–574.

[47] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2005), pp. 25–57.

[48] D. P. Young, W. P. Huffman, R. G. Melvin, C. L. Hilmes, and F. T. Johnson, *Nonlinear elimination in aerodynamic analysis and design optimization*, in Large-Scale PDE-Constrained Optimization, L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. Van Bloemen Waanders, eds., Springer, New York, 2003, pp. 17–44.