Math. Program., Ser. B (2012) 136:209–227
DOI 10.1007/s10107-012-0557-4

FULL LENGTH PAPER

A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations

Frank E. Curtis · Johannes Huber · Olaf Schenk · Andreas Wächter

Received: 12 April 2011 / Accepted: 4 April 2012 / Published online: 24 May 2012

Abstract This paper describes an implementation of an interior-point algorithm for large-scale nonlinear optimization. It is based on the algorithm proposed by Curtis et al. (SIAM J Sci Comput 32:3447–3475, 2010), a method that possesses global convergence guarantees to first-order stationary points with the novel feature that inexact search direction calculations are allowed in order to save computational expense. The implementation follows the proposed algorithm, but includes many practical enhancements, such as functionality to avoid the computation of a normal step during every iteration. The implementation is included in the `IPOPT` software package paired with an iterative linear system solver and preconditioner provided in `PARDISO`. Numerical results on a large nonlinear optimization test set and two PDE-constrained optimization problems with control and state constraints are presented to illustrate that the implementation is robust and efficient for large-scale applications.

Frank E. Curtis was supported by National Science Foundation grant DMS-1016291.

F. E. Curtis
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA
e-mail: frank.e.curtis@lehigh.edu

J. Huber
Department of Mathematics and Computer Science, University of Basel, Basel, Switzerland
e-mail: johannes.huber@unibas.ch

O. Schenk
Institute of Computational Science, Universitá della Svizzera italiana, Lugano, Switzerland
e-mail: olaf.schenk@usi.ch

A. Wächter (✉)
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA
e-mail: andreas.waechter@northwestern.edu

Keywords Large-scale optimization · PDE-constrained optimization · Interior-point methods · Nonconvex programming · Line search · Trust regions · Inexact linear system solvers · Krylov subspace methods

Mathematics Subject Classification 49M05 · 49M15 · 49M37 · 65F10 · 65K05 · 65N22 · 90C06 · 90C26 · 90C30 · 90C51 · 90C90

1 Introduction

The techniques described in this paper are motivated by increased interests in the solution of large-scale nonlinear optimization problems. By large-scale, we refer to classes of problems for which contemporary optimization techniques, including most interior-point methods, have proved to be impractical due to large numbers of variables/constraints and significant fill-in during the factorization of derivative matrices. New computationally efficient strategies are needed if such large-scale problems are to be solved realistically in practical situations.

The main purpose of this paper is to describe a practical implementation, including enhanced algorithmic features, for the algorithm proposed and analyzed in [13]. This algorithm addresses the challenges posed in large-scale nonlinear optimization by employing iterative linear system solvers in place of direct factorization methods when solving the large-scale linear systems involved in an interior-point strategy. Moreover, computational flexibility is greatly increased as inexact search direction calculations are allowed, but controlled sufficiently so that theoretical convergence guarantees are maintained. Our experience has shown that the implementation described in this paper achieves these desirable characteristics.

A prime example of a class of problems for which our techniques may be applicable are those where the constraints involve discretized partial differential equations (PDEs) [4,5,20]. Typical methods for solving these types of problems generally fall into the categories of nonlinear elimination [1,3,23], reduced space [19,21], or full space [6,7,18] techniques. The algorithm discussed in this paper fits into the category of full-space methods, but is unique from many previously proposed approaches in its ability to attain strong theoretical convergence guarantees with great computational flexibility.

We describe our implementation in the context of the generic problem

$$\min_{x\in\mathbb{R}^n} \; f(x) \;\; \text{s.t.} \;\; c_{\mathcal{E}}(x) = 0 \;\; \text{and} \;\; c_{\mathcal{I}}(x) \geq 0, \tag{1.1}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$, equality constraints $c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^p$, and inequality constraints $c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^q$ are assumed to be sufficiently smooth (e.g., C^2). If problem (1.1) is infeasible, however, then our algorithm is designed to return a stationary point for the unconstrained problem

$$\min_{x\in\mathbb{R}^n} \tfrac{1}{2}\left\|c_{\mathcal{E}}(x)\right\|_2^2 + \tfrac{1}{2}\left\| \max\{-c_{\mathcal{I}}(x), 0\} \right\|_2^2 \tag{1.2}$$

as a certificate of infeasibility. A solution to (1.2) that does not satisfy the constraints of problem (1.1) is known as an infeasible stationary point.

While the algorithm described in this paper is similar to the one presented in [13], we present its implementation here in more detail. In addition, there are a number of notable differences. The primary difference is the strategy we describe for switching between two step computation methods. Our goal is to improve the overall efficiency of the algorithm while still ensuring the convergence guarantees provided in [13]. The other main differences include refinements of the termination tests for the iterative linear system solver and Hessian modification strategy, as well as a new adaptive refinement strategy in the preconditioner computation. Due to space limitations, some of the motivations behind our techniques are omitted. A more comprehensive development can be found in the full-length version [10].

The focus of our numerical experiments is the performance of the nonlinear optimization algorithm when an iterative solver is used for the inexact solution of the arising linear systems. We note, however, that in practice it is also extremely important to employ effective preconditioners. For example, in the context of PDE-constrained optimization, advantages can be gained by exploiting spectral properties of discrete differential operators. In our study, we obtain very encouraging results using the general-purpose preconditioner implemented in PARDISO; see Sect. 3.1.

Notation. All norms are considered ℓ_2. We drop function dependencies once they are defined and, when applicable, apply iteration number information to the function itself; i.e., by c_k, we mean $c(x_k)$.

2 Algorithm description

We motivate and describe our implemented algorithm in this section. The method is based on the series of inexact SQP, Newton, and interior-point algorithms that have been proposed and analyzed in [8,9,12,13], though the majority relates to the latest enhancements in [13]. We begin by describing the basic interior-point framework of the approach, and then discuss the major computational component of the algorithm, namely, the search direction calculation. Further specifications and details of our software implementation are provided in Sects. 3 and 4.

It is important to note that, in this section, we consider *scaled* derivatives corresponding to the slack variables for the inequality constraints; see $\gamma(z;\mu)$, $A(z)$, and $W(z,\lambda;\mu)$ throughout this section. This results in *scaled* sets of equations for computing the primal-dual step; see [10] for details. In particular, we denote $\Sigma \approx S$ as a diagonal scaling matrix that depends on the values of the slack variables s.

2.1 Interior-point framework

The framework of the algorithm is a classical interior-point strategy. Let $z = (x, s)$ be the primal variables, where $s \in \mathbb{R}^q$ is a vector of slack variables, and let

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

For a sequence of barrier parameters $\mu \downarrow 0$, problem (1.1) is solved through the solution of a sequence of barrier subproblems of the form

$$\min_{z} \ \varphi(z;\mu) \ \text{ s.t. } \ c(z) = 0. \tag{2.1}$$

If f, $c_{\mathcal{E}}$, and $c_{\mathcal{I}}$ are continuously differentiable, then first-order Karush–Kuhn–Tucker (KKT) optimality conditions for (2.1) are

$$\begin{bmatrix} \gamma(z;\mu) + A(z)^T \lambda \\ c(z) \end{bmatrix} = 0 \tag{2.2}$$

with $s \geq 0$, where $\lambda \in \mathbb{R}^{p+q}$ are Lagrange multipliers, $e \in \mathbb{R}^q$ is a vector of ones,

$$\gamma(z;\mu) := \begin{bmatrix} \nabla f(x) \\ -\mu S^{-1} \Sigma e \end{bmatrix}, \quad \text{and} \quad A(z) := \begin{bmatrix} \nabla c_{\mathcal{E}}(x)^T & 0 \\ \nabla c_{\mathcal{I}}(x)^T & -\Sigma \end{bmatrix}.$$

In situations where (1.1) is infeasible, (2.1) has no solution, so the algorithm transitions from solving (2.1) to solving (1.2). The latter problem has the KKT conditions $A(z)^T c(z) = 0$, $s \geq 0$, and $c_{\mathcal{I}}(x) - s \leq 0$. In fact, the algorithm maintains $s \geq 0$ and $c_{\mathcal{I}}(x) - s \leq 0$ during each iteration by increasing s when necessary. Thus, convergence to a solution of the barrier subproblem (2.1) or an infeasible stationary point of (1.1) is achieved once (2.2) or $A(z)^T c(z) = 0$, respectively, is satisfied.

At an iterate (z_k, λ_k), the algorithm computes a primal-dual search direction (d_k, δ_k) satisfying appropriate conditions for guaranteeing global convergence; see Sect. 2.2. Given such a direction, we compute the scaled direction $\widetilde{d}_k := (d_k^x, \Sigma_k d_k^s)$ along which a line search is performed. The line search involves two conditions. First, to maintain positivity of the slacks, a stepsize $\alpha_k^{\max} \in (0, 1]$ satisfying

$$s_k + \alpha_k^{\max} \Sigma_k d_k^s \geq (1 - \eta_1) s_k \tag{2.3}$$

is determined for a constant $\eta_1 \in (0, 1)$. We use $\eta_1 = \max\{0.99, 1 - \mu\}$ in our implementation. The algorithm then backtracks from this value to compute $\alpha_k \in (0, \alpha_k^{\max}]$ yielding sufficient decrease in the penalty function

$$\phi(z;\mu,\pi) := \varphi(z;\mu) + \pi \|c(z)\|, \tag{2.4}$$

where $\pi > 0$ is a penalty parameter. The condition we enforce is

$$\phi(z_k + \alpha_k \widetilde{d}_k; \mu, \pi) \leq \phi(z_k; \mu, \pi) - \eta_2 \alpha_k \Delta m_k(d_k; \mu, \pi) \tag{2.5}$$

where $\eta_2 \in (0, 1)$ is a constant (we choose $\eta_2 = 10^{-8}$), and where $\Delta m_k(d_k; \mu, \pi)$ relates to the directional derivative of ϕ along the computed search direction.

We define $\Delta m_k(d_k;\mu,\pi)$ in equation (2.9) in Sect. 2.2. In the dual space, we update $\lambda_{k+1} \leftarrow \lambda_k + \beta_k \delta_k$ where β_k is the smallest value in $[\alpha_k, 1]$ satisfying

$$\left\| \gamma_k + A_k^T \lambda_{k+1} \right\| \leq \left\| \gamma_k + A_k^T (\lambda_k + \delta_k) \right\|. \tag{2.6}$$

Our complete interior-point framework is presented as Algorithm 1. In the algorithm, we refer to two search direction computation variants, Algorithms 2 and 3, presented in Sect. 2.2. These methods include mechanisms for computing (d_k, δ_k) and updating the penalty parameter π. The termination criteria for the original problem (1.1) and the barrier problem (2.1), the choice of the initial point, and the updating rule for the barrier parameter μ are identical with those in [28].

2.2 Search direction computation

The main computational component of Algorithm 1 is the primal-dual search direction calculation (step 4). We describe two approaches for computing this direction, presented as Algorithms 2 and 3. The first approach is simpler, but global convergence for Algorithm 1 is only guaranteed with this method if an infinite subsequence of iterations involve (scaled) constraint Jacobians $\{A_k\}$ that have full row rank and singular values bounded away from zero. Otherwise, the second approach must be employed to ensure convergence. The two algorithms have many common features, and we start by discussing the components present in both techniques. (In Sect. 3.1 we discuss our mechanism for having the algorithm dynamically choose between these two approaches during each iteration of Algorithm 1.)

Algorithm 1 Interior-Point Framework

```
1: (Initialization) Choose line search parameters $\eta_1, \eta_2 \in (0,1)$, an initial barrier parameter $\mu > 0$, and
   an initial penalty parameter $\pi > 0$. Initialize $(x_0, s_0, \lambda_0)$ so that the slack variables satisfy $s_0 > 0$ and
   $s_0 \geq c_{\mathcal{I}}(x_0)$. Set $k \leftarrow 0$.
2: (Tests for convergence) If convergence criteria for (1.1) are satisfied, then terminate and return $x_k$ as an
   optimal solution. Else, if convergence criteria for (1.2) are satisfied and $x_k$ is infeasible for (1.1), then
   terminate and return $x_k$ as an infeasible stationary point.
3: (Barrier parameter update) If convergence criteria for (2.1) are satisfied, then decrease the barrier param-
   eter $\mu > 0$, reset $\pi > 0$, and go to step 2.
4: (Search direction computation) Compute $(d_k, \delta_k)$ and update $\pi$ by Algorithm 2 or Algorithm 3. Set the
   search direction as $\widetilde{d}_k \leftarrow (d_k^x, \Sigma_k d_k^s)$.
5: (Line search) If $\widetilde{d}_k = 0$, then $\alpha_k \leftarrow 1$. Else, let $\alpha_k^{\max}$ be the largest value in $(0,1]$ satisfying (2.3) and
   let $l$ be the smallest value in $\mathbb{N}_0$ such that $\alpha_k \leftarrow 2^{-l}\alpha_k^{\max}$ satisfies (2.5).
6: (Iterate update) Set $z_{k+1} \leftarrow z_k + \alpha_k \widetilde{d}_k$, $s_{k+1} \leftarrow \max\{s_{k+1}, c_{\mathcal{I}}(x_{k+1})\}$, update $\lambda_{k+1}$ according to
   (2.6), set $k \leftarrow k+1$, and go to step 3.
```

The search direction computation is based on Newton's method applied to the KKT conditions of problem (2.1). Defining the scaled Hessian matrix

$$W(z,\lambda;\mu) := \begin{bmatrix} \nabla_{xx}^2 f & 0 \\ 0 & \Sigma\Xi\Sigma \end{bmatrix} + \sum_{i=1}^{p} \lambda_{\mathcal{E}}^{(i)} \begin{bmatrix} \nabla_{xx}^2 c_{\mathcal{E}}^{(i)} & 0 \\ 0 & 0 \end{bmatrix} + \sum_{i=1}^{q} \lambda_{\mathcal{I}}^{(i)} \begin{bmatrix} \nabla_{xx}^2 c_{\mathcal{I}}^{(i)} & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.7}$$

a Newton iteration for (2.2) is defined by the linear system

$$\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ c_k \end{bmatrix}. \tag{2.8}$$

In (2.7), Ξ is an approximate Hessian for the barrier term; see Sect. 3.2.

The central issue that must be confronted when applying Newton's method for large-scale applications is that exact solutions of (2.8) may be computationally expensive to obtain. Therefore, our major concern is how an iterative linear system solver can be employed for solving (2.8) in such a way that inexact solutions are allowed, yet global convergence of the algorithm is guaranteed. This issue was the inspiration for all of the algorithms proposed in [8,9,12,13].

Algorithms 2 and 3 each outline a series of termination tests for an iterative solver applied to (2.8) that state conditions under which an inexact solution (d_k, δ_k) can be considered an acceptable direction for step 4 in Algorithm 1. Of central importance in these termination tests are the residual vectors

$$\rho_k(d_k, \delta_k) := \gamma_k + W_k d_k + A_k^T(\lambda_k + \delta_k) \text{ and } r_k(d_k) := c_k + A_k d_k,$$

as well as the primal-dual relative residual

$$\Psi_k(d_k, \delta_k) := \left\| \begin{bmatrix} \rho_k(d_k, \delta_k) \\ r_k(d_k) \end{bmatrix} \right\| \Big/ \left\| \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \right\|.$$

To promote fast convergence, Ψ_k should be small [14]. Thus, our implementation aims to compute steps for which this relative residual is below a desired threshold, but appropriate fallbacks are in place in case this is difficult to achieve.

For convex problems, the algorithm can focus exclusively on Ψ_k, terminating the calculation of (d_k, δ_k) whenever this value is below a threshold [14]. For nonconvex problems, however, the priority is to find solutions to (2.2) that correspond to minimizers, not saddle points or maximizers. The methods developed in [8,9,12,13] therefore include additional conditions and procedures that aid the algorithms in converging toward minimizers of (2.1). These additional conditions involve a local model of the penalty function $\phi(z; \mu, \pi)$ at z_k, denoted as

$$m_k(d; \mu, \pi) := \varphi_k + \gamma_k^T d + \pi \|c_k + A_k d\|,$$

and the reduction in this model yielded by d_k, which is defined as

$$\Delta m_k(d_k; \mu, \pi) := m_k(0; \mu, \pi) - m_k(d_k; \mu, \pi). \tag{2.9}$$

It can be shown that $D\phi_k(\widetilde{d}_k; \mu, \pi) \leq -\Delta m_k(d_k; \mu, \pi)$, where $D\phi_k(\widetilde{d}_k; \mu, \pi)$ is the directional derivative of $\phi(\cdot; \mu, \pi)$ at z_k along $\widetilde{d}_k$; see [13]. To ensure that $\widetilde{d}_k$ is always a descent direction for $\phi(\cdot; \mu, \pi)$ at z_k, the termination tests require that $\Delta m_k(d_k; \mu, \pi)$ is sufficiently large, potentially after an increase of π.

We now present our first step computation approach, Algorithm 2. The algorithm contains three termination tests that are similar in form to those contained in Algorithm 3 later on. In [8,9,12,13], these tests are called sufficient merit function approximation reduction termination tests (SMART tests, for short) due to the significant role that $\Delta m_k(d_k;\mu,\pi)$ plays in the theoretical behavior the algorithm. For our numerical experiments, we choose $J=100$, $\kappa_{\text{des}}=10^{-3}$, $\kappa=10^{-2}$, $\epsilon_1=0.09$, $\theta=10^{-12}\mu$ (where μ is the current value of the barrier parameter), $\zeta=10^{-4}$, $\tau=0.1$, $\kappa_3=10^{-3}$, $\epsilon_3=10^{-8}$, and $\kappa_W=10^{-2}$. In particular, we typically require the relative residual to be only less than $\kappa=10^{-2}$ or $\kappa_{\text{des}}=\kappa_3=10^{-3}$, but the actual tolerance is dictated by the entirety of the termination criteria. The value for ξ is chosen as described in Sect. 3.2.

Algorithm 2 Inexact Newton Iteration with SMART Tests

```
1: (Initialization) Choose parameters $J \in \mathbb{N}_0$, $\kappa_{\text{des}} \in (0,1)$, $\kappa \in (0,1)$, $\epsilon_1 \in (0,1)$, $\theta > 0$, $\zeta > 0$,
   $\tau \in (0,1)$, $\kappa_3 \in (0,1)$, $\epsilon_3 > 0$, and $\kappa_W > 0$. Choose a diagonal matrix $D_k \succ 0$. Initialize $j \leftarrow 1$ and
   $(d_k,\delta_k) \leftarrow (0,0)$.
2: (Residual test) If $j \le J$ and $\Psi_k > \kappa_{\text{des}}$, then go to step 7.
3: (Termination test 1) If $\Psi_k \le \kappa$ and the model reduction condition
   $$\Delta m_k(d_k;\mu,\pi) \ge \max\left\{\tfrac{1}{2}d_k^TW_kd_k, \theta\|d_k\|^2\right\} + \epsilon_1\pi\max\{\|c_k\|, \|r(d_k)\|-\|c_k\|\} \quad (2.10)$$
   holds, then terminate by returning $(d_k,\delta_k)$ and the current $\pi$.
4: (Termination test 2) If the residual conditions
   $$\|\rho_k(d_k,\delta_k)\| \le \kappa\|c_k\| \text{ and } \|r_k(d_k)\| \le \kappa\|c_k\|$$
   are satisfied and the curvature condition $\frac{1}{2}d_k^TW_kd_k \ge \theta\|d_k\|^2$ holds, then terminate by returning $(d_k,\delta_k)$
   and $\pi \leftarrow \max\{\pi, \pi^t+\zeta\}$ where
   $$\pi^t \leftarrow \left(\gamma_k^Td_k + \tfrac{1}{2}d_k^TW_kd_k\right) \Big/ \left((1-\tau)(\|c_k\|-\|r_k(d_k)\|)\right).$$
5: (Termination test 3) If the dual displacement and feasibility measures satisfy
   $$\|\rho_k(0,\delta_k)\| \le \kappa_3\left\|\gamma_k + A_k^T\lambda_k\right\| \text{ and } \|c_k\| \le \epsilon_3\left\|\gamma_k + A_k^T\lambda_k\right\|,$$
   then terminate by returning $(0,\delta_k)$ (i.e., reset $d_k \leftarrow 0$) and the current $\pi$.
6: (Hessian modification) If $\Psi_k \le \kappa_W$ and $\frac{1}{2}d_k^TW_kd_k < \theta\|d_k\|^2$, then modify $W_k \leftarrow W_k + \xi D_k$, reset
   $j \leftarrow 1$ and $(d_k,\delta_k) \leftarrow (0,0)$, and go to step 2.
7: (Search direction update) Perform one iteration of an iterative solver on (2.8) to compute an improved
   (approximate) solution $(d_k,\delta_k)$. Increment $j \leftarrow j+1$ and go to step 2.
```

Our second approach, Algorithm 3, is a replacement for Algorithm 2 when the scaled constraint Jacobian may be ill-conditioned or rank-deficient. In such cases, it is necessary to regularize the step computation since otherwise the calculation may not be well-defined, or at best may lead to long, unproductive search directions.

Algorithm 3 performs this regularization by decomposing the search direction as $d_k \leftarrow v_k + u_k$, where the *normal component* v_k represents a direction toward linearized feasibility and the *tangential component* u_k represents a direction toward optimality.

Algorithm 3 Regularized Inexact Newton Iteration with SMART Tests

```
1: (Initialization) Choose parameters $J \in \mathbb{N}_0$, $\kappa_{\rm des} \in (0,1)$, $\psi > 0$, $\theta > 0$, $\kappa \in (0,1)$, $\epsilon_1 \in (0,1)$,
   $\epsilon_2 \in (0,1)$, $\zeta > 0$, $\tau \in (0,1)$, $\kappa_3 \in (0,1)$, $\epsilon_3 \in (0,1)$, and $\kappa_W > 0$. Choose a diagonal matrix $D_k \succ 0$.
   Compute $v_k$ by Algorithm 3 in [10]. Initialize $j \leftarrow 1$ and $(d_k, \delta_k) \leftarrow (0,0)$.
2: (Residual test) If $j \le J$ and $\Psi_k > \kappa_{\rm des}$, then go to step 10.
3: (Direction decomposition) Set $u_k \leftarrow d_k - v_k$.
4: (Tangential component test) If
       $$\|u_k\| \le \psi \|v_k\| \quad (2.11)$$
   or if the inequalities
       $$\tfrac{1}{2} u_k^T W_k u_k \ge \theta \|u_k\|^2 \quad (2.12a)$$
       $$(\gamma_k + W_k v_k)^T u_k + \tfrac{1}{2} u_k^T W_k u_k \le 0 \quad (2.12b)$$
   are satisfied, then continue to step 5; otherwise, go to step 8.
5: (Dual residual test) If the dual residual condition
       $$\|\rho_k(d_k,\delta_k)\| \le \kappa \min\left\{ \left\| \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ A_k v_k \end{bmatrix} \right\|, \left\| \begin{bmatrix} \gamma_{k-1} + A_{k-1}^T \lambda_k \\ A_{k-1} v_{k-1} \end{bmatrix} \right\| \right\} \quad (2.13)$$
   is satisfied, then continue to step 6; otherwise, go to step 8.
6: (Termination test 1) If the model reduction condition
       $$\Delta m_k(d_k;\mu,\pi) \ge \max\left\{ \tfrac{1}{2} u_k^T W_k u_k, \theta\|u_k\|^2 \right\} + \epsilon_1 \pi (\|c_k\| - \|r_k(v_k)\|) \quad (2.14)$$
   is satisfied, then terminate by returning $(d_k,\delta_k)$ and the current $\pi$.
7: (Termination test 2) If $\|c_k\| - \|r_k(d_k)\| \ge \epsilon_2(\|c_k\| - \|r_k(v_k)\|) > 0$, then terminate by returning $(d_k,\delta_k)$
   and $\pi \leftarrow \max\{\pi, \pi^t + \zeta\}$ where
       $$\pi^t \leftarrow \left(\gamma_k^T d_k + \tfrac{1}{2} u_k^T W_k u_k\right) \Big/ \left((1-\tau)(\|c_k\| - \|r_k(d_k)\|)\right).$$
8: (Termination test 3) If the dual displacement $\delta_k$ yields
       $$\|\rho_k(0,\delta_k)\| \le \kappa_3 \min\left\{ \left\| \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ A_k v_k \end{bmatrix} \right\|, \left\| \begin{bmatrix} \gamma_{k-1} + A_{k-1}^T \lambda_k \\ A_{k-1} v_{k-1} \end{bmatrix} \right\| \right\}$$
   and the stationarity and dual feasibility measures satisfy $\|A_k^T c_k\| \le \epsilon_3 \|\gamma_k + A_k^T \lambda_k\|$, then terminate by
   returning $(0,\delta_k)$ (i.e., reset $d_k \leftarrow 0$) and the current $\pi$.
9: (Hessian modification) If $\Psi_k \le \kappa_W$, but both (2.11) and (2.12a) do not hold, then modify $W_k \leftarrow$
   $W_k + \xi D_k$, reset $j \leftarrow 1$ and $(d_k,\delta_k) \leftarrow (0,0)$, and go to step 2.
10: (Search direction update) Perform one iteration of an iterative solver on (2.16) to compute an improved
   (approximate) solution $(d_k,\delta_k)$. Increment $j \leftarrow j+1$ and go to step 2.
```

The normal component v_k is defined as an approximate solution to

$$\min_v \ \tfrac{1}{2}\|r_k(v)\|^2 \quad \text{s.t.} \quad \|v\| \le \omega \|A_k^T c_k\| \tag{2.15}$$

for some $\omega > 0$. The trust region constraint regularizes the computation of the normal step and controls the size of this component even when A_k loses rank. We initialize $\omega \leftarrow 100$, but have found it practically beneficial to set $\omega \leftarrow \min\{10\omega, 10^{20}\}$ if $\|v_k\| = \omega\|A_k^T c_k\|$ and $\alpha_k = 1$ at the end of iteration k; see [12].

As with the linear system (2.8), the exact solution of (2.15) is expensive. However, global convergence is guaranteed as long as v_k is feasible for problem (2.15) and satisfies Cauchy decrease, i.e., $\|c_k\| - \|r_k(v_k)\| \geq \epsilon_v(\|c_k\| - \|r_k(\bar{\alpha}_k \bar{v}_k)\|)$ for some constant $\epsilon_v \in (0, 1)$ (we choose $\epsilon_v = 0.1$); see [12,13]. Here, the vector $\bar{v}_k := -A_k^T c_k$ is the steepest descent direction for problem (2.15) at $v = 0$ and the steplength $\bar{\alpha}_k$ is the minimizer of $\frac{1}{2}\|c_k + \bar{\alpha} A_k \bar{v}_k\|^2$ over all $\bar{\alpha} \leq \omega$.

A number of techniques have been developed for the inexact solution of large-scale instances of problem (2.15) with solutions satisfying Cauchy decrease; e.g., see the CG method described in [26]. In our software, we have implemented an inexact dog-leg approach [24] similar to that in [13]; see [10] for details. It requires the inexact solution of a system similar to (2.8) with W_k replaced by a positive definite diagonal matrix.

We now present Algorithm 3. Since v_k is computed separately from u_k, we now apply an iterative linear system solver to the reformulated system

$$\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ -A_k v_k \end{bmatrix}. \tag{2.16}$$

The relative residual Ψ_k is redefined accordingly as

$$\Psi_k(d_k, \delta_k) := \left\| \begin{bmatrix} \rho_k(d_k, \delta_k) \\ -A_k v_k + A_k d_k \end{bmatrix} \right\| \Bigg/ \left\| \begin{bmatrix} \gamma_k + A_k^T \lambda_k \\ -A_k v_k \end{bmatrix} \right\|.$$

Since (2.16) stipulates $A_k d_k = A_k v_k$, the system is consistent for suitable W_k. We choose the inputs for Algorithm 3 to be the same as those used in Algorithm 2. The values for the new constants are chosen to be $\epsilon_2 = 0.9$ and $\psi = 0.1$.

3 Algorithm details

3.1 Switching between search direction calculations and preconditioning

Algorithm 1 paired with Algorithm 3 constitutes an approach that is theoretically globally convergent to first-order stationary points under common assumptions [13]. However, as Algorithm 2 will produce viable search directions in most practical situations and, in contrast to Algorithm 3, it only requires the inexact solution of a single linear system, it is generally advantageous to pair Algorithm 1 with Algorithm 2 rather than with Algorithm 3. Thus, our implementation computes search directions with Algorithm 2 and only switches to Algorithm 3 when there is evidence that Algorithm 2 may be unable to produce a productive search direction.

Our trigger for switching between the two search direction algorithms is based on the steplength obtained as a result of the line search. If during a given iteration of Algorithm 1, Algorithm 2 has been employed for computing the search direction and the line search produces a steplength below a given threshold $\bar{\alpha}_1$, then this may be an indication that A_k is losing rank, causing the steps to become too large. (Of course, the short steplength may simply be due to the nonlinearity of the problem functions themselves, but even in that case the algorithm may benefit by employing Algorithm 3.) In such cases, we decide to employ Algorithm 3 in the following iteration of Algorithm 1 and continue to employ it until an iteration yields a steplength above $\bar{\alpha}_1$. The motivation for this scheme is that Algorithm 1 paired with Algorithm 2 is guaranteed to converge for an equality-constrained problem (e.g., a given barrier subproblem) under common assumptions, as long as the constraint Jacobians have full row rank and their smallest singular values are bounded away from zero. Specifically, for the analysis of Algorithm 2 in [9], the latter requirement is used to show that the steplength α_k is bounded away from zero. Thus, we use a small steplength α_k as an indicator to switch to Algorithm 3. In our implementation, we choose the threshold value to be $\bar{\alpha}_1 = 10^{-3}$.

When an iterative solver is used to solve the linear systems (2.8) and (2.16), an effective preconditioner is essential to keep the number of iterations low. For best performance, the preconditioner should be tailored to specific problems. However, for our numerical experiments, we use a general-purpose preconditioning approach. The details of our preconditioning method can be found in [10,25]. In short, the approach is based on an algebraic multi-level preconditioner designed for symmetric highly indefinite systems. It uses symmetric maximum weight matchings to improve the block diagonal dominance of the system, followed by an inverse-based pivoting strategy to compute an inexact factorization. In order to bound the norm of the inverse, the factorization of some rows and columns might be postponed to the end. This leaves a Schur complement to which the procedure is applied recursively within a multilevel preconditioning framework.

For the iterative solution of linear systems we use the symmetric quasi-minimum residual (SQMR) method [16] which has been found to work well with this preconditioner. Here, we allow a depth up to 30 in the multi-level approach [25], the constant bounding the norm of the inverse of the factor is chosen to be $\kappa_L = 2$, and the drop tolerances for the factor and the Schur complement are set to be $\epsilon_L = 10^{-2}$ and $\epsilon_S = 10^{-3}$, respectively. The SQMR method is allowed a maximum number of 1,500 iterations. If this number is exceeded, then the preconditioner is recomputed with tightened drop tolerances (both divided by 3) and the iteration counter is reset. If necessary, the tolerances are tightened repeatedly. If an acceptable solution for Algorithm 2 has not been computed after 4 such attempts, then the method reverts to Algorithm 3. If an acceptable solution for Algorithm 3 has not been computed after 4 such attempts, then the last computed inexact solution is used (without guarantees for a successful line search). In either of these latter two cases, before a new linear system is solved, the drop tolerances are multiplied by 3, though they are never set higher than the default values given above.

3.2 Hessian modification strategy and flexible penalty function

In the definition of the Hessian matrix W_k in (2.7), the choice $\Xi_k = \mu S_k^{-2}$ corresponds to the so-called *primal* interior-point iteration. This was considered for the global convergence analysis in [13]. However, our implementation follows the more efficient *primal-dual* interior-point strategy $\Xi_k = S_k^{-1} Y_k$, where $Y_k = \text{diag}(y_k)$ with dual variables y_k corresponding to the slack bounds (i.e. $s \geq 0$). It is easy to see that the analysis in [13] still holds as long as $\overline{\nu}\mu S_k^{-2} \succeq \Xi_k \succeq \underline{\nu}\mu S_k^{-2}$ for some constants $\overline{\nu} \geq 1 \geq \underline{\nu} > 0$; we choose $\overline{\nu} = 10^{10}$ and $\underline{\nu} = 10^{-10}$ in our experiments. This is achieved by adjusting y_k, if necessary, after each iteration; see [28].

Our strategy for modifying the Hessian in Algorithms 2 and 3 is analogous to the one described in [28], where a multiple of the identity is added to the *unscaled* Hessian matrix. This corresponds to using D_k with a (1,1)-block being I and a (2,2)-block being Σ_k^2. Furthermore, we choose ξ according to the strategy for choosing δ_w in Algorithm IC in [28]. However, in contrast to Algorithm IC, our trigger for a modification is not the inertia of the primal-dual system. Rather, we trigger a modification based on the conditions described in step 6 of Algorithm 2 and step 9 of Algorithm 3. We have also found it beneficial to trigger a modification at the start of the search direction computation if in the previous iteration the line search reduced α_k due to the sufficient decrease condition (2.5). This leads to somewhat shorter search directions and makes the acceptance of larger steplengths more likely, often leading to a reduction in iteration count.

An important algorithmic feature of our code is the use of a flexible penalty function [11]. This mechanism is designed to avoid a pitfall of penalty functions, namely the potential for the algorithm to set an unnecessarily large value of the penalty parameter and thus restrict the iterates to remain close to the feasible region. This can lead to small steplengths and slow convergence.

The effect of the flexible penalty function on our line search is that, instead of requiring α_k to satisfy the sufficient decrease condition (2.5) for a *fixed* π, we only require that α_k satisfies a sufficient decrease condition for *some* π in an interval $[\pi^l, \pi^u]$. In particular, given $\pi^l \leq \pi^m \leq \pi^u$, $\alpha_k \in (0, \alpha_k^{\max}]$ is acceptable as long as

$$\phi(z_k + \alpha_k \widetilde{d}_k; \mu, \pi) \leq \phi(z_k; \mu, \pi) - \eta_2 \alpha_k \Delta m_k(d_k; \mu, \pi^m) \text{ for some } \pi \in [\pi^l, \pi^u]. \tag{3.1}$$

We have adapted the strategy proposed in [11] for updating π^l and π^u and for setting π^m during each iteration. In particular, our updates for π^l and π^u are essentially those in [11]; the details can be found in [10]. As for the choice of π^m, we consider two cases. If in the current iteration Termination test 2 was satisfied, but Termination test 1 was not, then we follow [11] and set π^m to be the maximum of π^l and π^t, where π^t is computed during Termination test 2. This choice guarantees that the model reduction $\Delta m_k(d_k; \mu, \pi^m)$ is positive. Otherwise, if Termination test 1 was satisfied, then we set π^m to be π^l since, based on our decision to use π^l as the penalty parameter value in (2.10) and (2.14), this choice also guarantees that the model reduction $\Delta m_k(d_k; \mu, \pi^m)$ is positive. Overall, we guarantee that (3.1) is a sufficient decrease condition for ϕ in either case.

4 Numerical experiments

The algorithm described in the previous sections was implemented in the `IPOPT` open-source package (http://www.coin-or.org/Ipopt/); for our experiments we use revision 1954 of the `branches/parallel` development branch. The linear systems are solved using the iterative linear system solvers and preconditioners implemented in PARDISO (http://www.pardiso-project.org/) version 4.1.1. The finite-element discretization of the PDEs in Sects. 4.1–4.2 was implemented using the open-source libmesh library [22], revision 3881 in its trunk branch, together with the PETSc library [2] version 3.1-p3. The three-dimensional meshes for the example in Sect. 4.2 are generated with the tetgen software (http://tetgen.berlios.de/).

In IPOPT, we use default parameter settings with a termination tolerance 10^{-6}, together with the parameter choices given in the previous sections. The iterative linear solver in PARDISO uses SQMR [16] with the preconditioner described in Sect. 3.1.

To assess the robustness of the algorithm we compare its performance with the default method in IPOPT on AMPL [15] versions of problems from the CUTE set [17]; details of the experiment can be found in [10]. Algorithm 1 is able to solve 89 % of the 617 problems, and is therefore almost as robust as the default algorithm in IPOPT that solves 96 %. Note that for 142 problems, the algorithm switches to Algorithm 3 at some point, indicating that the switching strategy in Sect. 3.1 was utilized by the algorithm. In contrast, the method proposed in [13], which always uses Algorithm 3, has a success rate of only 84 %, meaning that we have achieved an increase in robustness of 5 % by utilizing Algorithm 2.

The numerical experiments in the rest of this section illustrate the performance of our implementation on two PDE-constrained problems. We show that our method provides improved computation times compared to the default IPOPT algorithm. The results were obtained on 8-core Intel Xeon machines with 2.33 GHz clock speed and 32 GB RAM, running Ubuntu Linux with GNU 4.4.1 compilers.

4.1 Optimal boundary control

Our first PDE-constrained optimization problem is an optimal control problem motivated by the "Heating with radiation boundary conditions" example in Section 1.3.1 of [27]. The PDE is to satisfy $-\Delta T = 0$, where T denotes temperature, in a domain $\Omega \subseteq \mathbb{R}^3$. A suitable boundary condition is to satisfy $\frac{\partial T}{\partial n} = \chi(u - T^4)$ on the boundary Γ of Ω. This condition expresses the radiation heat loss according to the Stefan–Boltzmann law with a Stefan's constant $\chi > 0$, where the control $u \geq 0$ dictates heat that can be resupplied on Γ. The goal is to minimize the amount of heat supplied, i.e., $\int_\Gamma u\, da$, while attaining a temperature of $T_j^{\min}$ or more within N_S subregions $\Omega_j \subseteq \Omega$. We multiply $-\Delta T = 0$ with a test function $v \in H^1(\Omega)$ and apply Green's formula together with the boundary condition. The weak formulation of the PDE is then to find $T \in H^1(\Omega)$ such that

$$0 = -\int_\Omega \Delta T v\, dx = \int_\Omega \nabla T \cdot \nabla v\, dx - \chi \int_\Gamma (T^4 - u)\, v\, da \quad \forall v \in H^1(\Omega). \tag{4.1}$$

We generate a regular mesh of tetrahedrons, each with volume $h^3/24$ for $h > 0$, and use the standard linear finite element basis functions $\{\varphi_i\}_{i=1,\ldots,n_h}$. Projecting (4.1) onto the generated finite dimensional subspace V^h by approximating T with $T^h = \sum_i T^{(i)}\varphi_i$ and u by $u^h = \sum_i u^{(i)}\varphi_i$ (the latter requires discretized values $u^{(i)}$

Table 1 Problem sizes for instances of the example in Sect. 4.1

h	#var	#bds	#eq	#ineq
0.04	89,453	9,183	81,951	0
0.03	199,389	16,498	186,319	0
0.02	670,153	42,313	640,151	0

Table 2 Performance measures for the example in Sect. 4.1

h	alg	it	$f(x_*)$	CPUs	CPUs/it	Speedup
0.04	`IPOPT`	33	39.9458	2,806.16	85.04	
0.04	Alg. 1	33	39.9458	646.77	19.60	4.34
0.03	`IPOPT`	34	37.7909	16,330.56	480.31	
0.03	Alg. 1	37	37.7909	4,495.83	121.51	3.63
0.02	`IPOPT`	46	40.9115	304,780.45	6,625.66	
0.02	Alg. 1	47	40.9115	38,824.33	826.05	7.85

only corresponding to the boundary Γ), we solve the finite-dimensional problem

$$\min_{u^{(i)},T^{(i)}} \sum_i u^{(i)} \int_\Gamma \varphi_i \, da$$

subject to $u^{(i)} \geq 0$ and

$$0 = \int_\Omega \sum_i T^{(i)} \nabla\varphi_i \cdot \nabla\varphi_j \, dx - \chi \int_\Gamma \left(\left(\sum_i T^{(i)} \varphi_i \right)^4 - \sum_i u^{(i)} \varphi_i \right) \varphi_j \, da \quad \forall j \tag{4.2a}$$

$$T^{(i)} \geq T_j^{\min} \quad \text{for } j \in \{1, \ldots, N_S\} \text{ and } i \in \left\{ \hat{i} \mid \exists x \in \Omega_j : \varphi_{\hat{i}}(x) = 1 \right\}. \tag{4.2b}$$

We choose $\chi = 1$ and $\Omega = (0,1)^3$ and define two regions to be heated, $\Omega_1 = [0.1, 0.2] \times [0.05, 0.3] \times [0, 0.1]$ and $\Omega_2 = [0.8, 1] \times [0.75, 1] \times [0.7, 1]$, with associated threshold temperatures of $T_1^{\min} = 2.5$ and $T_2^{\min} = 2$. In (4.2b), we used the fact that a nodal finite element basis was chosen, so that $\max_{x\in\Omega} \varphi_i(x) = 1$, and for all $x \in \Omega$ we have $\sum_i \varphi_i(x) = 1$. Since $\nabla\varphi_i \cdot \nabla\varphi_j = \mathcal{O}(1/h^2)$ and $\int_E dx = \mathcal{O}(h^3)$ for a tetrahedron E, we multiply (4.2a) by $10^{-2}/h$ in our implementation to ensure that the gradients of these constraints do not vanish as $h \to 0$. Similarly, the objective function was scaled internally by the factor $10^{-2}/(h^2)$.

We executed the optimization algorithm for three choices of the discretization level. As initial point, we chose $T = T_{\text{init}}$ with $T_{\text{init}} = 1.1(T_1^{\min} + T_2^{\min})$, and $u = (T_{\text{init}})^4$. Table 1 shows the discretization parameter (h), number of optimization variables (#var), number of simple bound constraints (#bds), number of equality constraints (#eq), and number of inequality constraints (#ineq) for various instances of this example. Table 2 provides performance measures in the form of number of iterations (it),

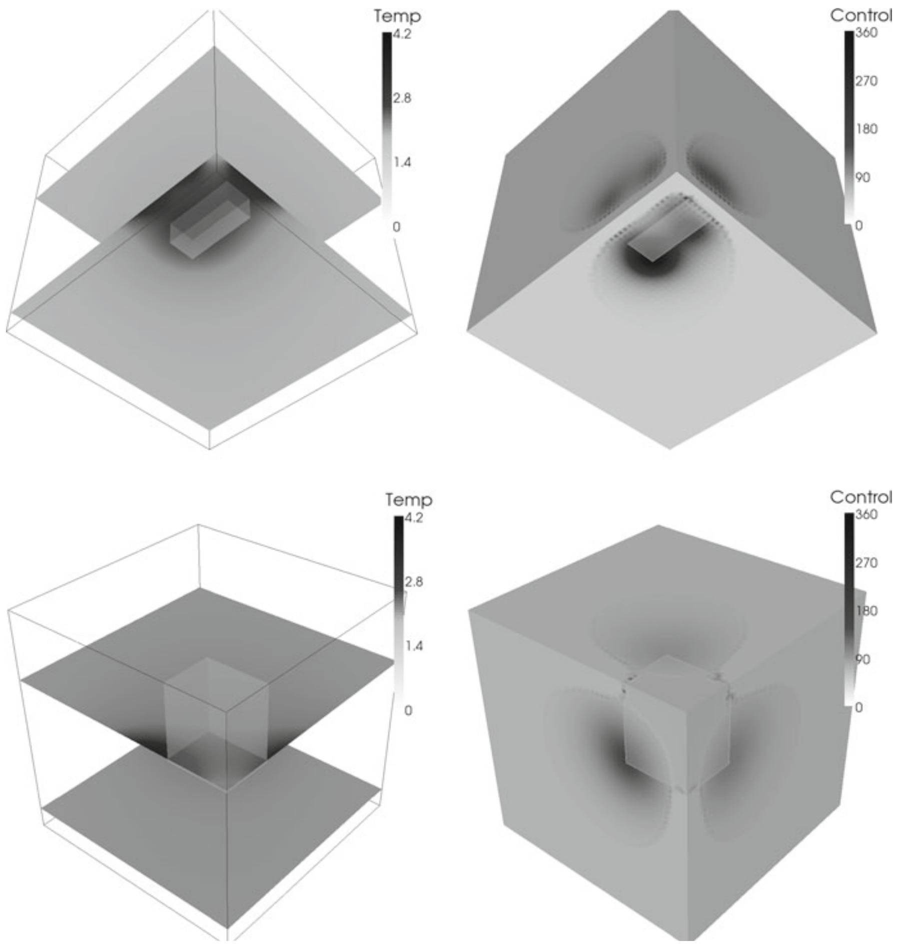

Fig. 1 Optimal state (*left*) and control (*right*) for the example in Sect. 4.1. The regions Ω_1 (*top*) and Ω_2 (*bottom*) are visualized as a box. It is interesting to note that the corners of the regions Ω_1 and Ω_2 are heated most, instead of the inner part of its surface

final objective value $f(x_*)$, CPU seconds (CPUs), and CPU seconds per iteration (CPUs/it) for the default `IPOPT` algorithm and for Algorithm 1. The last column shows the overall CPU time speedup obtained by the inexact algorithm. We see a significant gain in computation speed that becomes more pronounced as the problem size increases. For the largest problem, the speedup is a factor of 7.85. The corresponding solution is depicted in Fig. 1.

Table 2 lists the average CPU time per iteration, but it should be noted that the step computation requires considerably more time towards the end of the optimization procedure. Considering the case $h = 0.02$, in the first 22 `IPOPT` iterations the preconditioners (each computed in less than 1 min) have fill-in factors of at most 3 and SQMR requires only between 35 and 200 iterations, leading to less than 4 min for each step computation. However, in the final `IPOPT` iterations, the dropping tolerances have to be tightened. Then, the preconditioner (computed in up to 9 min) has a fill-in factor of almost 10 and SQMR requires more than 1,000 iterations, leading to times up to 35 min per step computation.

4.2 Server room cooling

Our second example is motivated by the problem of cooling computer equipment. In our simplified model, we assume that cold air is blown into a room from air conditioners (ACs), and that air leaves the room at exhausts. Inside the domain lies equipment with hot surfaces that need to be cooled by air passing alongside.

For simplicity, we suppose that air is incompressible, has no internal friction, and that all velocities are far below the speed of sound. Under these assumptions, we can model air velocity $y(x)$ as the gradient of a potential $\Phi(x)$ satisfying the Laplace equation $-\Delta\Phi = 0$ in $\Omega \subseteq \mathbb{R}^3$. Appropriate boundary conditions for the walls (including non-heat producing surfaces of the equipment) Γ_W and for the heat producing surfaces Γ_T of the equipment are $\frac{\partial\Phi}{\partial n} = 0$ on $\Gamma_W \cup \Gamma_T$. Similarly, boundary conditions for the cold air inlets are $\frac{\partial\Phi}{\partial n} = -u_{AC_i}\Psi_{\Gamma_{AC_i}}$ on Γ_{AC_i}, and for the exhausts are $\frac{\partial\Phi}{\partial n} = u_{Ex_i}\Psi_{\Gamma_{Ex_i}}$ on Γ_{Ex_i}. Here, $\Psi_{\Gamma_{AC_i}}(x)$ $(i = 1, \ldots, N_{AC})$ and $\Psi_{\Gamma_{Ex_i}}(x)$ $(i = 1, \ldots, N_{Ex})$ define velocity profiles on the surfaces of the ACs and exhausts, respectively. Similarly, $u_{AC_i} \in \mathbb{R}$ and $u_{Ex_i} \in \mathbb{R}$ denote control parameters for the maximal flow rates at these air inlets and outlets. The weak formulation of the PDE is to find $\Phi \in H^1(\Omega)$ such that, for all $v \in H^1(\Omega)$, we have

$$0 = \int_{\Omega} \nabla\Phi \cdot \nabla v \, dx + \sum_{i=1}^{N_{AC}} \int_{\Gamma_{AC_i}} u_{AC_i}\Psi_{\Gamma_{AC_i}} v \, da - \sum_{i=1}^{N_{Ex}} \int_{\Gamma_{Ex_i}} u_{Ex_i}\Psi_{\Gamma_{Ex_i}} v \, da. \tag{4.3}$$

It is important to note that (4.3) has a solution only if the controls satisfy the mass balance equation

$$\sum_{i=1}^{N_{AC}} \int_{\Gamma_{AC_i}} u_{AC_i}\Psi_{\Gamma_{AC_i}} da - \sum_{i=1}^{N_{Ex}} \int_{\Gamma_{Ex_i}} u_{Ex_i}\Psi_{\Gamma_{Ex_i}} da = 0, \tag{4.4}$$

and in that case (4.3) only determines the potential $\Phi \in H^1(\Omega)$ up to an additive constant. Therefore, a normalization condition will be introduced below.

As a constraint, we require that the air speed at the heat-producing surfaces has a minimum velocity so that heat is carried away. More precisely, recalling that the velocity is the gradient of the potential function Φ, we impose the point-wise state constraint $\|\nabla\Phi(x)\|_2^2 \geq y_{\min}^2$ for all $x \in \Gamma_T$ with a constant $y_{\min} > 0$.

To obtain the discretized problem, we generate an irregular mesh of tetrahedrons, each with maximal volume h^3, again choose a finite-dimensional subset $V^h \subseteq H^1(\Omega)$ with a basis $\{\varphi_i\}_{i=1,\ldots,n_h}$, and express the finite-dimensional approximation Φ_h of $\Phi = \sum_i \phi^{(i)}\varphi_i$ with coefficients $\phi \in \mathbb{R}^{n_h}$. Defining $u = (u_{AC}, u_{Ex})$ as the vector consisting of all control parameters, the discretized PDE (4.3) then becomes $A\phi - Bu = 0$, where A denotes the stiffness matrix $A^{(i,j)} = \int_{\Omega} \nabla\varphi_i \cdot \nabla\varphi_j dx$, and

$B = [B_{AC}\ B_{Ex}]$ implements the boundary conditions with $B_{AC}^{(i,j)} = -\int_{\Gamma_{AC_j}} \Psi_{\Gamma_{AC_j}} \varphi_i \, da$ and $B_{Ex}^{(i,j)} = \int_{\Gamma_{Ex_j}} \Psi_{\Gamma_{Ex_j}} \varphi_i \, da$.

The finite-dimensional optimization problem is

$$\min_{\phi_i, u_i, \bar{u}} \sum \beta_j u_{AC_j}$$

subject to $u \geq 0$ and

$$A\phi - Bu + \gamma e \bar{u} = 0 \tag{4.5a}$$

$$\gamma e^T \phi - \bar{\gamma} \bar{u} = 0 \tag{4.5b}$$

$$e^T Bu = 0 \tag{4.5c}$$

$$\int_{\Gamma_e} \nabla\phi(x) \cdot \nabla\phi(x) \, da - y_{\min}^2 \left(\int_{\Gamma_e} da \right) \geq 0 \text{ for } \Gamma_e \subseteq \Gamma_T \tag{4.5d}$$

with weights $\beta_i > 0$ in the objective function, and $e = (1, \ldots, 1)^T \in \mathbb{R}^{n_h}$. Here, (4.5c) is a compact way of writing (4.4), and (4.5d) is the discretized version of the point-wise state constraint, which is posed for all element faces Γ_e contained in a heat producing surface Γ_T. Note that the constraint (4.5d) is nonlinear and nonconvex. Again, in our implementation of the above problem, we scaled the constraints (4.5a) and (4.5d) by factors $10^{-2}/h$ and $10^{-1}/h$, respectively, to ensure that the gradients of those functions do not vanish as $h \to 0$.

To overcome the ill-posedness of the PDE, an auxiliary variable $\bar{u} \in \mathbb{R}$ has been added. Equation (4.5a) includes the discretized PDE, where $\gamma e \bar{u}$ acts as a virtual source or sink over Ω. Since we impose mass conservation in (4.5c), this term yields $\bar{u} = 0$. Furthermore, an integral-type equation is imposed in (4.5b). Indeed, $e^T \phi$ is a discretization of $\int_\Omega \Phi d\mu$ for some measure μ depending on the finite-element discretization and is set to zero in (4.5b) since $\bar{u} = 0$, therefore normalizing Φ.

For our experiments, we choose $\beta_i = 1$, $\gamma = 1$, $\bar{\gamma} = 10^8$, $y_{\min} = 1$, $\Gamma_{AC_1} = \{0\} \times [0.4, 0.6] \times [0.2, 0.4]$, $\Gamma_{AC_2} = [0.4, 0.6] \times \{0\} \times [0.2, 0.4]$, $\Gamma_{AC_3} = [0.4, 0.6] \times \{1\} \times [0.2, 0.4]$, and $\Gamma_{Ex_1} = \{1\} \times [0.4, 0.6] \times [0.6, 0.8]$. The equipment is placed so that $\Omega_{Eq_1} = [0.2, 0.7] \times [0.2, 0.4] \times [0, 0.8]$ and $\Omega_{Eq_2} = [0.2, 0.6] \times [0.6, 0.8] \times [0, 0.8]$ with the remaining boundaries Γ_T and Γ_W defined as illustrated in Fig. 2. The airflows at the inlets and outlets are defined to have quadratic profiles.

Due to the nooks in Ω created by the equipment, numerical experiments with linear finite elements showed only linear L^2-convergence of the PDE solution as $h \to 0$. However, since the point-wise state constraint involves the gradient of the state variable, superlinear convergence is crucial. Thus, we have chosen quadratic finite-elements and observed quadratic convergence for the PDE solution; see [10].

Table 3 shows sizes for various instances of this problem. As a starting point, we calculated the solution of (4.5a)–(4.5c) for $u_{AC_i} = 20$. Table 4 provides performance measures for the default IPOPT algorithm and for our implementation. Also here we see a clear reduction in computation time achieved by using the inexact algorithm,

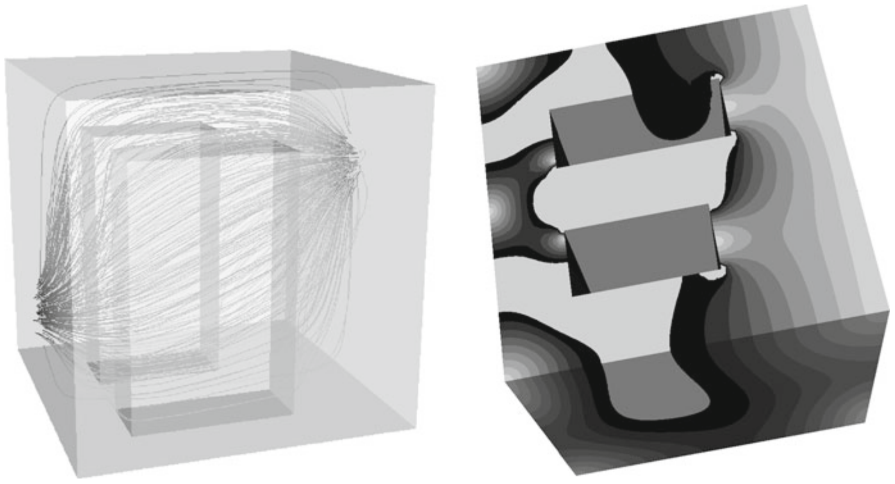

Fig. 2 Optimal solution of the server room cooling optimization example. *On the left*, we see the streamlines of the airflow, going from the main AC on the left to the exhaust on the right. *On the right*, we have a bottom view of the domain Ω, where the colors have been chosen to be dark if the air velocity is close to the threshold $y_{\min} = 1$. One can clearly see a region at the wall of the larger piece of equipment, at which the velocity is close to critical, indicating the location of the active constraints (4.5d) in Γ_T

Table 3 Problem sizes for instances of the example in Sect. 4.2

h	#var	#bds	#eq	#ineq
0.03	88,398	4	88,395	1,528
0.02	285,510	4	285,507	3,409
0.015	663,886	4	663,883	6,110

Table 4 Performance measures for the example in Sect. 4.2

h	alg	it	$f(x_*)$	CPUs	CPUs/it	Speedup
0.03	`IPOPT`	21	15.5283	4,511.48	214.83	
0.03	Alg. 1	24	15.5283	1,710.30	71.26	2.64
0.02	`IPOPT`	33	15.5694	69,427.33	2,103.86	
0.02	Alg. 1	28	15.5694	10,008.50	357.45	6.94
0.015	`IPOPT`	32	15.6509	528,320.22	16,510.01	
0.015	Alg. 1	27	15.6509	29,526.53	1,093.58	17.89

without a loss in solution accuracy. Specifically, the computation time for the largest instance with more than 600,000 variables was reduced from more than 6 days to 8.2 h, a speedup by a factor of 17.89. Figure 2 shows the optimal solution for $h = 0.015$.

In this example, the default settings for the preconditioner thresholds were sufficient in each iteration, so no tightening occurred. For the $h = 0.015$ case, the computation time for the preconditioner ranged from 164 to 234 s (with an average of 204 s), the number of SQMR iterations was 204–1,129 with an average of 396, and the time spent in SQMR ranged from 365 to 2,018 s (with an average of 719 s). While there was some variation, we did not observe a degeneration of computation time per step computation as in Sect. 4.1.

5 Conclusion and final remarks

We have presented a detailed description of an implementation of a primal-dual interior-point method for large-scale nonconvex optimization where the search directions are computed inexactly by means of an iterative linear system solver. Ideally, the algorithm computes a search direction through the inexact solution of a single linear system (as in [9]). However, when appropriate, it falls back on the step decomposition strategy proposed in [13] so that, overall, the strong global convergence properties presented in [13] are attained. Numerical experiments on a large set of test problems and on two PDE-constrained optimization problems have also been presented. These results demonstrate the robustness of the approach and illustrate the significant speedup our algorithm attains when compared to an algorithm based on direct factorizations of the primal-dual system matrices.

As mentioned at the end of Sect. 4.1, we have observed a decrease in effectiveness of the preconditioner as the barrier parameter approaches zero. It remains a subject of future research to explore whether the specific structure of the ill-conditioning caused by the log-barrier terms can be handled efficiently within the incomplete-factorization-based preconditioner described in Sect. 3.1.

Acknowledgments The authors thank Michael Henderson, Vanessa Lopez, and Ulisses Mello for suggesting the server room cooling example, and Madam Sathe for discussions regarding the preconditioner. They also thank the referees whose suggestions led to significant improvements. Johannes Huber gratefully acknowledges financial support from IBM during his summer internship at the IBM Watson Research Center.

References

1. Arnautu, V., Neittaanmaki, P.: Optimal Control from Theory to Computer Programs. Kluwer, Dordrecht (2003)
2. Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc users manual. Technical Report ANL-95/11—Revision 3.1, Argonne National Laboratory (2010)
3. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming Advances in Design and Control. SIAM, Philadelphia (2001)
4. Biegler, L.T., Ghattas, O., Heinkenschloss, M., Keyes, D., Van Bloemen Waanders, B.: Real-Time PDE-Constrained Optimization. Computational Science and Engineering. SIAM, Philadelphia (2007)
5. Biegler, L.T., Ghattas, O., Heinkenschloss, M., Van Bloemen Waanders, B. (eds.): Large-Scale PDE-Constrained Optimization. Lecture Notes in Computational Science and Engineering. Springer, Berlin (2003)
6. Biros, G., Ghattas, O.: Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part I: the Krylov–Schur solver. SIAM J. Sci. Comput. **27**(2), 687–713 (2005)
7. Biros, G., Ghattas, O.: 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**(2), 714–739 (2005)
8. Byrd, R.H., Curtis, F.E., Nocedal, J.: An inexact SQP method for equality constrained optimization. SIAM J. Optim. **19**(1), 351–369 (2008)

9. Byrd, R.H., Curtis, F.E., Nocedal, J.: An inexact Newton method for nonconvex equality constrained optimization. Math. Program. **122**(2), 273–299 (2010)
10. Curtis, F.E., Huber, J., Schenk, O., Wächter, A.: On the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations. Technical report, Lehigh ISE 12T-006, Optimization Online ID: 2011-04-2992 (2012)
11. Curtis, F.E., Nocedal, J.: Flexible penalty functions for nonlinear constrained optimization. IMA J. Numer. Anal. **28**(4), 749–769 (2008)
12. Curtis, F.E., Nocedal, J., Wächter, A.: A matrix-free algorithm for equality constrained optimization problems with rank deficient Jacobians. SIAM J. Optim. **20**(3), 1224–1249 (2009)
13. Curtis, F.E., Schenk, O., Wächter, A.: An interior-point algorithm for nonlinear optimization with inexact step computations. SIAM J. Sci. Comput. **32**(6), 3447–3475 (2010)
14. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. SIAM J. Numer. Anal. **19**(2), 400–408 (1982)
15. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming. Brooks/Cole, Belmont (2002)
16. Freund, R.W.: Preconditioning of symmetric, but highly indefinite linear systems. In: Sydow, A. (ed.) 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, pp. 551–556. Wissenschaft & Technik, Berlin (1997)
17. Gould, N.I.M., Bongartz, I., Conn, A.R., Toint, Ph.L.: CUTE: constrained and unconstrained testing environment. ACM Trans. Math. Softw. **21**(1), 123–160 (1995)
18. Haber, E., Ascher, U.M.: Preconditioned all-at-once methods for large, sparse parameter estimation problems. Inverse Probl. **17**, 1847–1864 (2001)
19. Heinkenschloss, M., Vicente, L.N.: Analysis of inexact trust-region SQP algorithms. SIAM J. Optim. **12**(2), 283–302 (2002)
20. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE Constraints, Volume 23 of Mathematical Modeling: Theory and Applications. Springer, Dordrecht (2009)
21. Jäger, H., Sachs, E.W.: Global convergence of inexact reduced SQP methods. Optim. Methods Softw. **7**(2), 83–110 (1997)
22. Kirk, B.S., Peterson, J.W., Stogner, R.H., Carey, G.F.: libMesh: A C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng. Comput. **22**(3–4), 237–254 (2006)
23. Kirk, D.E.: Optimal Control Theory: An Introduction. Prentice-Hall, Englewood Cliffs (1970)
24. Powell, M.J.D: A hybrid method for nonlinear equations. In: Rabinowitz, P. (ed.) Numerical methods for nonlinear algebraic equations, pp. 87–114. Gordon and Breach, London (1970)
25. Schenk, O., Bollhöfer, M., Roemer, R.A.: On large scale diagonalization techniques for the Anderson model of localization. SIAM J. Sci. Comput. **28**(3), 963–983 (2006)
26. Steihaug, T.: The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal. **20**(3), 626–637 (1983)
27. Tröltzsch, F.: Optimal control of partial differential equations: theory, methods, and applications, Volume 112. American Mathematical Society, Providence (2010)
28. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)