Math. Prog. Comp. (2012) 4:181–209
DOI 10.1007/s12532-012-0041-4

FULL LENGTH PAPER

A penalty-interior-point algorithm for nonlinear constrained optimization

Frank E. Curtis

Received: 26 April 2011 / Accepted: 11 April 2012 / Published online: 24 April 2012

Abstract Penalty and interior-point methods for nonlinear optimization problems have enjoyed great successes for decades. Penalty methods have proved to be effective for a variety of problem classes due to their regularization effects on the constraints. They have also been shown to allow for rapid infeasibility detection. Interior-point methods have become the workhorse in large-scale optimization due to their Newton-like qualities, both in terms of their scalability and convergence behavior. Each of these two strategies, however, have certain disadvantages that make their use either impractical or inefficient for certain classes of problems. The goal of this paper is to present a penalty-interior-point method that possesses the advantages of penalty and interior-point techniques, but does not suffer from their disadvantages. Numerous attempts have been made along these lines in recent years, each with varying degrees of success. The novel feature of the algorithm in this paper is that our focus is not only on the formulation of the penalty-interior-point subproblem itself, but on the design of updates for the penalty and interior-point parameters. The updates we propose are designed so that rapid convergence to a solution of the nonlinear optimization problem or an infeasible stationary point is attained. We motivate the convergence properties of our algorithm and illustrate its practical performance on large sets of problems, including sets of problems that exhibit degeneracy or are infeasible.

Keywords Nonlinear optimization · Large-scale optimization · Nonconvex optimization · Constrained optimization · Penalty methods · Interior-point methods · Penalty-interior-point methods

F. E. Curtis (✉)
Department of Industrial and Systems Engineering, Lehigh University,
Bethlehem, PA 18018, USA
e-mail: frank.e.curtis@gmail.com

Mathematics Subject Classification 49M05 · 49M15 · 49M20 · 49M29 · 49M37 · 65F05 · 65F50 · 65K05 · 65K10 · 90C06 · 90C26 · 90C30 · 90C51

1 Introduction

We present an algorithm for nonlinear optimization problems of the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} \ (\underset{x}{\text{min}}) & f(x) \\ \text{subject to (s.t.)} & c(x) \leq 0, \end{array} \tag{1.1}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$ and the constraints $c : \mathbb{R}^n \to \mathbb{R}^t$ are continuously differentiable. Due to regularization techniques discussed subsequently in this paper, the solution methods we describe for problem (1.1) can also be applied to problems with equality constraints $c^{\mathcal{E}}(x) = 0$, either by treating them as the inequalities $c^{\mathcal{E}}(x) \leq 0$ and $-c^{\mathcal{E}}(x) \leq 0$ or, better yet, by relaxing them with slack variables. For simplicity in our presentation, we focus on the inequality constrained case, but further details on handling equality constraints are presented in Sect. 4.

The main purpose of this paper is to describe a method for the solution of (1.1) that confronts two key challenges. First, we are interested in the solution of large-scale problems, and so solution techniques that require a computationally expensive procedure to be performed during each iteration (such as the solution of a linear or quadratic optimization subproblem) are inefficient for our purposes. Second, we want the method to be regularized so that it may efficiently handle problems with difficult constraint sets. By this, we mean problems with constraint redundancies and/or (near) singularities in the constraint Jacobian. We also aim for the method to be able to handle infeasible instances of (1.1). In such cases, a certificate of infeasibility can be returned to the user if the algorithm minimizes a feasibility violation measure as in

$$\min_x \ v(x), \quad \text{where} \quad v(x) := \sum_{i=1}^{t} \max\{c^i(x), 0\} \tag{1.2}$$

at a point that does not satisfy the constraints of (1.1). (Here, c^i denotes the ith element in the vector function c.) A number of nonlinear optimization algorithms provide convergence guarantees to solutions of (1.2) if problem (1.1) is infeasible, but in many situations the rate of convergence to such points is extremely slow. In contrast, our aim is to solve (1.1) or (1.2) *rapidly*.

One successful strategy for the solution of problem (1.1) is to employ a *penalty method*. Such an approach essentially removes the constraints of the problem and replaces them with cost terms in the objective so as to *penalize* violations in the original constraints. For example, if the cost terms correspond to an ℓ_1 norm (as in the feasibility violation measure $v(x)$ in (1.2)), then the penalty subproblem associated with (1.1) can be posed as

$$\begin{aligned} &\min_{x,s} \ \rho f(x) + \sum_{i=1}^{t} s^i \\ &\text{s.t. } c(x) - s \leq 0, \ s \geq 0, \end{aligned} \tag{1.3}$$

where $\rho \geq 0$ is a penalty parameter. A challenge in the implementation of such an approach is the design of a technique for decreasing ρ during the solution process so that (1.1), or (1.2) if (1.1) is infeasible, is solved efficiently. A second challenge is that many implementations of penalty methods require the solution of constrained linear or quadratic optimization subproblems numerous times throughout the solution process; e.g., see [5,7–9,20]. This fact makes such approaches intractable for many large-scale applications.

Interior-point methods present a more efficient alternative to penalty methods for the solution of large-scale problems. The methodology in such a strategy is to reduce the problem to one with only equality constraints through the addition of slack variables that are forced to remain *interior* to the non-negative orthant. Specifically, after formulating the logarithmic barrier subproblem

$$\begin{aligned} &\min_{x,r} \ f(x) - \mu \sum_{i=1}^{t} \ln r^i \\ &\text{s.t. } c(x) + r = 0 \ \ (\text{with } r > 0) \end{aligned} \tag{1.4}$$

problem (1.1) is solved by (approximately) solving (1.4) for decreasing values of $\mu > 0$. As for a penalty method, a challenge in the implementation of an interior-point method is the design of an effective strategy for updating μ, which must be driven to zero to yield a solution to (1.1). A second challenge is that interior-point methods often lack a sophisticated mechanism for regularizing the constraints that is present in, say, a penalty approach. As a result, they can perform poorly on problems with difficult constraint sets.

In this paper, we motivate and describe an algorithm that employs both penalty and interior-point strategies. The hope is that by combining penalty and interior-point techniques in an appropriate manner, we may successfully maintain their advantages, but not suffer from their disadvantages. We join components of (1.3) and (1.4) to form the *penalty-interior-point* subproblem

$$\begin{aligned} &\min_{z} \ \phi(z;\rho,\mu) \ \text{ s.t. } \ \theta(z) = 0, \\ &\text{where } \ \phi(z;\rho,\mu) \ := \ \rho f(x) - \mu \sum_{i=1}^{t} (\ln r^i + \ln s^i) + \sum_{i=1}^{t} s^i \\ &\text{and } \ \theta(z) \qquad := \ c(x) + r - s \ \ (\text{with } r > 0 \text{ and } s > 0) \end{aligned} \tag{1.5}$$

that is to be solved for the variables in $z := [x^T \ r^T \ s^T]^T$. This subproblem may be derived, for example, by applying an interior-point reformulation to the penalty subproblem (1.3) and then eliminating trivial equality constraints. We show in Sect. 2

that, as long as the sequences of values for ρ and μ are set appropriately, solving (1.5) can be used as a means for solving (1.1), or at least (1.2).

Our framework is not without significant challenges. Indeed, a daunting task in any approach that considers solving subproblem (1.5) in order to produce a solution to (1.1) or (1.2) is the design of an update strategy for *two* parameters: the penalty parameter ρ and the interior-point parameter μ. However, we believe that we have designed an approach that is both intuitively appealing and yields solid preliminary performance results. Rather than tackle updates for ρ and μ simultaneously, we have crafted a mechanism that updates their values sequentially during each iteration at the modest cost of only a few matrix-vector operations. Our numerical experiments illustrate that our updating scheme consistently results in fewer iterations than a more conservative strategy that only updates ρ and μ by monitoring progress over the course of numerous iterations.

A number of related penalty-interior-point-like algorithms have previously been proposed, analyzed, and tested. The first such methods that in some way resemble our approach were the *modified barrier* methods proposed in [24,30] (see also [31]). These algorithms, originally developed to eradicate the ill-conditioning effects caused by classical logarithmic-barrier functions applied to inequality constrained problems, essentially incorporate Lagrange multiplier estimates to play the role of penalty parameters within a logarithmic barrier term. See also [16] for an extension of such methods to handle equality constraints through the use of augmented Lagrangians. More recent methods that also resemble our techniques are the *penalty interior-point* algorithm in [1], the *penalty-barrier* methods in [3,4], the *interior-point* ℓ_2*-penalty* method in [10,11], the *interior-point* ℓ_1*-penalty* method proposed in [18], and the *exterior-point* approach discussed in [32]. These algorithms (and ours) differ in detail, though they are all inspired by the benefits of regularization through penalties and the prospect of search direction calculations through linear system solves as in many interior-point methods.

The outline of this paper is as follows. In Sect. 2 we present theoretical foundations for our algorithm by illustrating that solving the penalty-interior-point subproblem (1.5) is an appropriate means for solving problem (1.1) or, if a feasible solution is not found, problem (1.2). We then develop and present our algorithm in Sect. 3. Numerical results on large collections of nonlinear optimization test problems are provided in Sect. 4 and concluding remarks are presented in Sect. 5.

Notation We use superscripts to denote (sets of) elements of a vector and subscripts to denote iteration numbers of an algorithm. An exception is when we use the superscript(s) ρ and/or μ, in which case we are indicating a quantity's dependence (in some sense) on these parameters. Vectors comprised of stacked subvectors are written as ordered lists of subvectors; e.g., by $z = (x, r, s)$ we mean $z = [x^T \ r^T \ s^T]^T$. Displacements in variables are denoted by placing Δ before the variable name, and for the displacement corresponding to a particular iteration k, the subscript follows the variable name. For example, the step in x computed during iteration k is denoted as Δx_k. Scalars and vectors are denoted by lowercase letters, matrices are denoted by capital letters, and the capitalization of a vector name indicates the diagonal matrix formed by placing elements of that vector on the diagonal; e.g., $R := \text{diag}(r)$. I and e,

respectively, denote the identity matrix and vector of all ones of any size, the expression $M_1 \succ M_2$ is used to indicate that the matrix $M_1 - M_2$ is positive definite, and all norms are considered ℓ_2 unless otherwise indicated. Finally, we denote the Hadamard (or Schur) product of two vectors e and s of the same length as the vector $e \circ s$, which has entries $(e \circ s)^i = e^i s^i$.

2 Theoretical foundations

In this section, we discuss relationships between the nonlinear optimization problem (1.1), the feasibility problem (1.2), the penalty subproblem (1.3), and the penalty-interior-point subproblem (1.5). We argue that (1.3) and (1.5) have nice regularity properties no matter the properties of the vector function c and that solving (1.5) is an appropriate technique for solving (1.3), which in turn is appropriate for solving (1.1) or (1.2) depending on the selected value(s) for ρ. The results in this section are straightforward to prove and have appeared various times in the literature. Thus, formal proofs are not provided.

For now, we only make the following assumption.

Assumption 2.1 The functions f and c are continuously differentiable on $\mathbb{R}^n$.

Under Assumption 2.1 alone, we have the following result stating that, regardless of the properties of the constraints in (1.1), the penalty and penalty-interior-point subproblems are always regular in the sense that the Mangasarian-Fromovitz constraint qualification (MFCQ) is satisfied at *all* feasible points.

Theorem 2.2 *The MFCQ for* (1.3) *is satisfied at all feasible points for* (1.3). *Similarly, the MFCQ for* (1.5) *is satisfied at all feasible points for* (1.5).

Our goal now will be to argue that solving (1.5) for a sequence of interior-point parameters such that $\mu \to 0$ is an effective means for solving (1.3). Indeed, this should already be evident if one views (1.5) as the interior-point subproblem for (1.3), but our remarks highlight restrictions that should be placed on the slack variables and on the Lagrange multipliers during the optimization process. We require specifications of first-order optimality conditions for subproblems (1.3) and (1.5), which are given below. Note that first-order optimality conditions for (1.3) include those for the feasibility problem (1.2) as a special case.

Definition 2.3 A point $(x^\rho, s^\rho, \lambda^\rho)$ is first-order optimal for (1.3) if

$$\rho \nabla f(x^\rho) + \nabla c(x^\rho)\lambda^\rho = 0 \tag{2.1a}$$
$$\lambda^\rho \circ (c(x^\rho) - s^\rho) = 0 \tag{2.1b}$$
$$(e - \lambda^\rho) \circ s^\rho = 0 \tag{2.1c}$$
$$c(x^\rho) - s^\rho \leq 0 \tag{2.1d}$$
$$e \geq \lambda^\rho \geq 0 \tag{2.1e}$$
$$s^\rho \geq 0. \tag{2.1f}$$

If $\rho = 0$, then x^ρ is first-order optimal for (1.2) if (2.1) holds for some (s^ρ, λ^ρ).

Definition 2.4 A point $(z^{\rho,\mu}, \lambda^{\rho,\mu})$ is first-order optimal for (1.5) if

$$\rho \nabla f(x^{\rho,\mu}) + \nabla c(x^{\rho,\mu})\lambda^{\rho,\mu} = 0 \tag{2.2a}$$
$$R^{\rho,\mu}\lambda^{\rho,\mu} - \mu e = 0 \tag{2.2b}$$
$$S^{\rho,\mu}(e - \lambda^{\rho,\mu}) - \mu e = 0 \tag{2.2c}$$
$$\theta(z^{\rho,\mu}) = 0 \tag{2.2d}$$
$$(r^{\rho,\mu}, s^{\rho,\mu}) > 0. \tag{2.2e}$$

The following result links first-order optimal points of (1.5) and those of (1.3).

Theorem 2.5 *Fix $\rho \geq 0$ and, for $\mu \to 0$, let $\{(z^{\rho,\mu}, \lambda^{\rho,\mu})\}$ be a sequence of first-order optimal points for (1.5). Then, each cluster point $(x^\rho, s^\rho, \lambda^\rho)$ of $\{(x^{\rho,\mu}, s^{\rho,\mu}, \lambda^{\rho,\mu})\}$ has $s^\rho \geq 0$ and $e \geq \lambda^\rho \geq 0$, and so is a first-order optimal point for (1.3).*

Theorem 2.5 is enlightening as it indicates bounds that should be placed on the slack variables and Lagrange multipliers during the optimization process for (1.5). In particular, the slack variables should remain interior to the non-negative orthant and each multiplier should be restricted to $(0, 1)$ so that, as $\mu \to 0$, a sequence of solutions to (2.2) may produce a first-order optimal point for (1.3).

The hope in solving the penalty subproblem (1.3) is that, as long as the penalty parameter $\rho > 0$ is updated appropriately, a solution of (1.3) will correspond to a solution of the nonlinear optimization problem (1.1). (Note that it has already been established in Definition 2.3 that solutions of (1.3) for $\rho = 0$ correspond to solutions of the feasibility problem (1.2).) Thus, we are now ready to consider the relationship between (1.1) and (1.3), for which we require first-order optimality conditions for problem (1.1).

Definition 2.6 A point (x_*, λ_*) is first-order optimal for (1.1) if

$$\nabla f(x_*) + \nabla c(x_*)\lambda_* = 0 \tag{2.3a}$$
$$\lambda_* \circ c(x_*) = 0. \tag{2.3b}$$
$$c(x_*) \leq 0 \tag{2.3c}$$
$$\lambda_* \geq 0. \tag{2.3d}$$

The following result is well-known and the proof is straightforward.

Theorem 2.7 *If for $\rho > 0$ the point $(x^\rho, s^\rho, \lambda^\rho)$ is a first-order optimal point for (1.3) with $c(x^\rho) \leq 0$, then $(x_*, \lambda_*) = (x^\rho, \lambda^\rho/\rho)$ is a first-order optimal point for (1.1).*

This result confirms that solving the penalty problem (1.3) is an effective means for solving (1.1) or (1.2). That is, if $\rho > 0$ can be chosen small enough so that the solution of (1.3) provides a feasible solution to (1.1), then the slack variables will vanish and an optimal primal-dual pair for (1.1) is at hand. On the other hand, if no such ρ can be found, then the algorithm should reduce ρ so that a certificate of infeasibility can be returned in the form of a primal-dual pair for (1.2) that corresponds to an infeasible point for (1.1).

Our last result shows that as long as the Lagrange multipliers in (2.3) are finite, there exists a sufficiently small penalty parameter such that a particular first-order optimal point of (1.1) corresponds to a first-order optimal point for (1.3). Thus, if (1.1) is feasible and a constraint qualification is assumed that guarantees finiteness of the Lagrange multipliers at all first-order optimal points, then there exists an overall lower bound on ρ needed for solving (1.1).

Theorem 2.8 *If* (x_*, λ_*) *is a first-order optimal point for* (1.1)*, then for all penalty parameter values* $\rho \in (0, 1/\|\lambda_*\|_\infty]$*, the point* $(x^\rho, s^\rho, \lambda^\rho) = (x_*, 0, \rho\lambda_*)$ *is a first-order optimal point for* (1.3)*. Thus, if the set of Lagrange multipliers corresponding to solutions of* (2.3) *is bounded, then there exists* $\rho_* > 0$ *such that any first-order optimal point for* (1.1) *corresponds to a first-order optimal point for* (1.3) *for* $\rho \in (0, \rho_*]$.

One option for guaranteeing finiteness of the set of optimal Lagrange multipliers is to impose, for example, the MFCQ on (1.1) itself. An assumption such as this is common, but it is important to note that it cannot be imposed if equality constraints are handled inappropriately. Thus, although we consider the inequality constrained form (1.1) for the majority of our discussion, it is crucial to outline reasonable means for solving generally constrained problems. We do this in Sect. 4.

This concludes our discussion of the theoretical foundations for our proposed algorithm. We have seen that the penalty-interior-point subproblem (1.5) is sufficiently regular in that the MFCQ is satisfied at all feasible points. This same property holds for the penalty subproblem (1.3), to which (1.5) (in some sense) converges as $\mu \to 0$. Thus, we expect solving (1.5) to be a reasonable task, even if the constraints of problem (1.1) are difficult or infeasible. Moreover, we have seen that as long as the penalty and interior-point parameters are updated appropriately and the slack variables and Lagrange multipliers are restricted to remain in appropriate intervals, a first-order optimal point for (1.1), or at least (1.2), can be obtained by solving the first-order optimality conditions (2.2).

3 Algorithm development

We develop the details of our algorithm in a couple stages so as to highlight the important aspects of each component. For given ρ and μ, the algorithm essentially amounts to a line-search Newton method for solving (1.5) with additional functionalities for maintaining the slack variables and Lagrange multipliers in appropriate intervals. We begin with the basics behind our step computation and line search, and then discuss strategies for updating the penalty and interior-point parameters during each iteration. Complete descriptions of a conservative and an adaptive algorithm are presented during the course of this section.

3.1 Search direction computation and line search

Our search direction computation is derived from a Newton iteration applied to the optimality conditions for problem (1.5) (i.e., (2.2)) for given values of the penalty

parameter ρ and the interior-point parameter μ. At a given iterate (z_k, λ_k), applying a Newton iteration to (2.2) involves a linear system of the form

$$\begin{bmatrix} H_k & 0 & 0 & \nabla c(x_k) \\ 0 & \Omega_k & 0 & I \\ 0 & 0 & \Gamma_k & -I \\ \nabla c(x_k)^T & I & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta r_k \\ \Delta s_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \rho \nabla f(x_k) + \nabla c(x_k) \lambda_k \\ \lambda_k - \mu R_k^{-1} e \\ e - \lambda_k - \mu S_k^{-1} e \\ \theta(z_k) \end{bmatrix}. \tag{3.1}$$

Defining the Lagrangian of (1.5) to be

$$L(z, \lambda; \rho, \mu) := \phi(z; \rho, \mu) + \lambda^T \theta(z),$$

it is appropriate to choose

$$H_k \approx \nabla^2_{xx} L(z_k, \lambda_k; \rho, \mu) = \rho \nabla^2_{xx} f(x_k) + \sum_{i=1}^{t} \lambda_k^i \nabla^2_{xx} c^i(x_k) \tag{3.2}$$

so that the Newton matrix in (3.1) has the desired inertia $(n+2t, t, 0)$; see, e.g., [14] and Assumption 3.1 below. For the (diagonal) matrices Ω_k and Γ_k, there are two choices depending on whether the second and third blocks in (2.2) are scaled—with R and S, respectively—before or after derivatives are taken to derive (3.1). If the equations are scaled beforehand, then this leads to a *primal* step computation with $\Omega_k := \mu R_k^{-2}$ and $\Gamma_k := \mu S_k^{-2}$. If the equations are scaled afterward, then we have a *primal-dual* step computation with $\Omega_k := R_k^{-1} \Lambda_k$ and $\Gamma_k := S_k^{-1}(I - \Lambda_k)$. It is generally accepted that computational experience supports the primal-dual option over the primal option, so we make this choice in our algorithm.

As stated, the linear system (3.1) forms the basis of our search direction computation. Prior to solving (3.1), however, we incorporate an additional procedure to adjust the slack variables. The procedure, which we call a *slack reset* (see also the "magical steps" of [18, Sect. 6.1]), effectively eliminates the slack variables from much of the iteration. It proceeds in the following manner. Suppose that x_k is given so that $c(x_k)$ is known. Then, set r_k and s_k to solve

$$\begin{aligned} \min_{r,s} \; & -\mu \sum_{i=1}^{t} (\ln r^i + \ln s^i) + \sum_{i=1}^{t} s^i \\ \text{s.t. } & c(x_k) + r - s = 0 \;\; (\text{with } (r, s) > 0) \end{aligned} \tag{3.3}$$

for the current $\mu > 0$. (Note that, with x fixed at x_k, (1.5) reduces to (3.3).) This problem is convex and separable, and it has the unique solution given by

$$\begin{aligned} r_k^i &= r^i(x_k; \mu) := \mu - \tfrac{1}{2} c^i(x_k) + \tfrac{1}{2} \sqrt{c^i(x_k)^2 + 4\mu^2} \\ \text{and} \quad s_k^i &= s^i(x_k; \mu) := \mu + \tfrac{1}{2} c^i(x_k) + \tfrac{1}{2} \sqrt{c^i(x_k)^2 + 4\mu^2} \end{aligned} \tag{3.4}$$

for all $i = 1, \ldots, t$. Thus, at any iterate x_k and for any $\mu > 0$, the slack variables r_k and s_k can trivially be set to solve (3.3).

Applied at the start (or, equivalently, at the end) of every iteration, the important effects of this slack reset are two-fold. First, the obvious effect is that the constraints of (1.5) are satisfied throughout the solution process. This means that, with respect to the optimality conditions (2.2), the rest of the algorithm need only focus on satisfying the components (2.2a) and (2.2b)–(2.2c) pertaining to dual feasibility and complementarity, respectively. The more significant effect, however, relates to the fact that we can avoid having to define a *third* parameter (along with ρ and μ) that might otherwise be needed in a merit function to balance the priorities of minimizing the objective and satisfying the constraints of (1.5). That is, with $r_k = r(x_k; \mu)$ and $s_k = s(x_k; \mu)$ defined by (3.4), the component Δx_k obtained by solving (3.1) is a descent direction at $x = x_k$ for the merit function

$$\widetilde{\phi}(x; \rho, \mu) := \rho f(x) - \mu \sum_{i=1}^{t} (\ln r^i(x; \mu) + \ln s^i(x; \mu)) + \sum_{i=1}^{t} s^i(x; \mu).$$

We formalize this claim with Theorem 3.2 below. For the lemma, we make the following assumption. Such an assumption is standard for line search methods with search direction computations similar to (3.1).

Assumption 3.1 The matrix H_k in (3.1) yields

$$\begin{aligned} &\Delta x_k^T H_k \Delta x_k + \Delta r_k^T \Omega_k \Delta r_k + \Delta s_k^T \Gamma_k \Delta s_k > 0 \\ \text{for all } \Delta z_k \neq 0 \text{ such that } &\nabla c(x_k)^T \Delta x_k + \Delta r_k - \Delta s_k = 0, \end{aligned} \tag{3.5}$$

which, in particular, implies

$$\Delta x_k^T H_k \Delta x_k > 0 \text{ for all } \Delta x_k \neq 0 \text{ such that } \nabla c(x_k)^T \Delta x_k = 0.$$

Theorem 3.2 *Let $r_k = r(x_k; \mu)$ and $s_k = s(x_k; \mu)$ be set as in (3.4). Then, the search direction Δx_k yielded by (3.1) satisfies*

$$\begin{aligned} \nabla \widetilde{\phi}(x_k; \rho, \mu)^T \Delta x_k &= \nabla \phi(z_k; \rho, \mu)^T \Delta z_k \\ &= -\Delta x_k^T H_k \Delta x_k - \Delta r_k^T \Omega_k \Delta r_k - \Delta s_k^T \Gamma_k \Delta s_k < 0 \end{aligned} \tag{3.6}$$

and so is a descent direction for $\widetilde{\phi}(x; \rho, \mu)$ at $x = x_k$.

Proof The result follows from the representations

$$\nabla \widetilde{\phi}(x_k; \rho, \mu)^T \Delta x_k = \rho \nabla f(x_k)^T \Delta x_k + \sum_{i=1}^{t} \frac{1}{2} \left(1 + \frac{\mu}{r_k^i} - \frac{\mu}{s_k^i}\right) \nabla c^i(x_k)^T \Delta x_k$$

$$\text{and } \nabla \phi(z_k; \rho, \mu)^T \Delta z_k = \rho \nabla f(x_k)^T \Delta x_k + \sum_{i=1}^{t} \left[-\frac{\mu}{r_k^i} \Delta r_k^i + \left(1 - \frac{\mu}{s_k^i}\right) \Delta s_k^i \right]$$

and the fact that, for $i = 1, \ldots, t$, (3.1) implies

$$-\frac{\mu}{r_k^i}\Delta r_k^i + \left(1 - \frac{\mu}{s_k^i}\right)\Delta s_k^i = \frac{1}{2}\left(1 + \frac{\mu}{r_k^i} - \frac{\mu}{s_k^i}\right)\nabla c^i(x_k)^T \Delta x_k.$$

□

It is also prudent to state the relationship between minimizers of $\widetilde{\phi}(\cdot; \rho, \mu)$ and first-order optimal points of (1.5) when the slack variables are set as in (3.4) and steps in the Lagrange multipliers are computed via the Newton system (3.1). We do so with the following theorem, the proof of which is straightforward.

Theorem 3.3 *Suppose that for fixed ρ and μ the sequence of iterates $\{x_k\}$ yields*

$$\nabla\widetilde{\phi}(x_k; \rho, \mu) \to 0, \tag{3.7}$$

r_k and s_k are set by (3.4) for all k, and the Lagrange multipliers $\{\lambda_k\}$ yield

$$\left\| \begin{bmatrix} R_k\lambda_k - \mu e \\ S_k(e - \lambda_k) - \mu e \end{bmatrix} \right\| \to 0. \tag{3.8}$$

Then, any limit point of $\{(z_k, \lambda_k)\}$ is first-order optimal for (1.5).

The backtracking line search procedure in our algorithm proceeds in the following manner. First, as in the implementation of many other interior-point methods, we restrict the search by observing fraction-to-the-boundary rules for the slack and dual variables; see Definition 2.4 and Theorem 2.5. An appropriate feature in our case, however, is that these rules take into account our slack reset procedure. That is, we require that the primal steplength $\alpha_k \in (0, 1]$ satisfies

$$r(x_k + \alpha_k\Delta x_k; \mu) \geq \min\{\tau, \mu\}r_k \quad \text{and} \quad s(x_k + \alpha_k\Delta x_k; \mu) \geq \min\{\tau, \mu\}s_k \tag{3.9}$$

for a given $\tau \in (0, 1)$. We also require that the steplength satisfies

$$\widetilde{\phi}(x_k + \alpha_k\Delta x_k; \rho, \mu) \leq \widetilde{\phi}(x_k; \rho, \mu) + \alpha_k\eta\nabla\widetilde{\phi}(x_k; \rho, \mu)^T\Delta x_k \tag{3.10}$$

for some $\eta \in (0, 1)$. Once $\alpha_k \in (0, 1]$ is set satisfying (3.9) and (3.10), we update

$$x_{k+1} \leftarrow x_k + \alpha_k\Delta x_k, \quad r_{k+1} \leftarrow r(x_{k+1}; \mu), \quad \text{and} \quad s_{k+1} \leftarrow s(x_{k+1}; \mu). \tag{3.11}$$

In the dual space, we impose a fraction-to-the-boundary rule to compute

$$\beta_k = \max\{\beta \in (0, 1] : \lambda_k + \beta\Delta\lambda_k \in [\min\{\tau, \mu\}\lambda_k, e - \min\{\tau, \mu\}(e - \lambda_k)]\} \tag{3.12}$$

and then update

$$\lambda_{k+1} \leftarrow \lambda_k + \beta_k \Delta\lambda_k. \tag{3.13}$$

The rule (3.12) is imposed to ensure $\lambda_k \in (0, 1)$ for all k. This, along with the restriction that the slack variables remain in the non-negative orthant, guarantees that the Hessian matrices Ω_k and Γ_k are positive definite for all k.

Note that if the interior-point parameter μ is changed after the update (3.11), but before the start of the next iteration, then the slack variables should again be reset by (3.4). This computation requires only a few vector operations.

3.2 Updating the penalty and interior-point parameters

We are now ready to present our techniques for updating the penalty and interior-point parameters during the optimization process. In the pursuit of appropriate updates for these quantities, there are at least two significant challenges. The first is the design of updating techniques that result in rapid convergence to a first-order optimal point of the nonlinear optimization problem (1.1), or at least of the feasibility problem (1.2). The second is that the user (and so the algorithm) does not know beforehand which of these problems should be solved. The algorithm must determine dynamically the emphasis that should be placed on minimizing violations in feasibility and minimizing the objective of (1.1), bearing in mind that there should be a preference for solving (1.1) instead of only solving (1.2). We present two updating schemes—one conservative and one adaptive—the latter of which we believe successfully addresses these challenges.

Our conservative strategy is detailed as Algorithm 1 below. At the heart of the algorithm is the conservative strategy for interior-point methods known as the Fiacco–McCormick approach [13]. In our context, this approach corresponds to setting fixed values for ρ and μ until (2.2) is satisfied to a sufficient accuracy, at which point μ is decreased and the procedure is repeated. The update we employ initially multiplies μ by a fraction, but then raises it to a power greater than 1 to yield superlinear convergence of μ to 0; e.g., see [37]. Despite the fact that it suffers from important limitations, the Fiacco–McCormick approach has been employed in various interior-point algorithms due to the foundation it provides for ensuring global convergence; e.g., see [6,36]. As for updating the penalty parameter ρ, Algorithm 1 uses a simple approach that only decreases ρ when a (sufficiently accurate) solution of (1.5) does not correspond to a (sufficiently accurate) solution to (1.1). We also include a safeguard to ensure that the iterates do not diverge from the feasible region. (Specific conditions corresponding to statements such as "satisfied to a sufficient accuracy" will be presented in Sect. 4.)

The conservative updating strategy in Algorithm 1 is representative of the updating schemes that have been implemented in other penalty-interior-point-like algorithms. Indeed, the algorithm in [1] essentially follows this same approach, except that it defines a unique penalty parameter for each slack variable. The algorithms in [10,18] update the penalty parameter by monitoring a feasibility violation measure over the course of all iterations—another type of conservative technique—though their method for updating the interior-point parameter is similar to the Fiacco–McCormick approach

Algorithm 1 Penalty-interior-point algorithm with conservative updates

1: (Initialization): Choose a fraction-to-the-boundary parameter $\tau \in (0, 1)$, sufficient decrease parameter $\eta \in (0, 1)$, backtracking parameter $\gamma \in (0, 1)$, reduction parameters $\kappa_\rho \in (0, 1)$, $\kappa_\mu \in (0, 1)$, and $\xi^\mu \in (1, 2)$, and infeasibility warning level $\omega > 0$. Choose an initial penalty parameter $\rho > 0$ and interior-point parameter $\mu > 0$. Choose an initial primal-dual iterate (x_0, λ_0) and set the slack variables (r_0, s_0) by (3.4). Set $k \leftarrow 0$.
2: (Optimality check): If (2.2) with the current ρ and $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is sufficiently small, then terminate; x_k is a first-order optimal point for (1.1).
3: (Infeasibility check): If (2.2) with $(\rho, \mu) = (0, 0)$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, then terminate; x_k is an infeasible stationary point for (1.1).
4: (Hessian computation): Set $\Omega_k := R_k^{-1} \Lambda_k$, $\Gamma_k := S_k^{-1}(I - \Lambda_k)$, and H_k so that (3.5) holds.
5: (Search direction computation): Compute $(\Delta z_k, \Delta \lambda_k)$ by (3.1).
6: (Line search): Set $\bar{\alpha}_k$ as the largest value in $(0, 1]$ such that (3.9) holds, then set α_k as the largest value in $\{\bar{\alpha}_k \gamma^0, \bar{\alpha}_k \gamma^1, \bar{\alpha}_k \gamma^2, \dots\}$ such that (3.10) holds. Set β_k by (3.12).
7: (Iterate update): Set z_{k+1} and λ_{k+1} by (3.11) and (3.13), respectively.
8: (Interior-point parameter update): If (2.2) with the current ρ and μ is satisfied to a sufficient accuracy, then set $\mu \leftarrow \min\{\kappa_\mu \mu, \mu^{\xi_\mu}\}$ and reset r_{k+1} and s_{k+1} by (3.4).
9: (Penalty parameter update): If (2.2) with the current ρ and $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, or if the new constraint violation satisfies $v(x_{k+1}) > \max\{v(x_0), v(x_k), \omega\}$, then set $\rho \leftarrow \kappa_\rho \rho$.
10: (k increment): Set $k \leftarrow k + 1$ and go to step 2.

described above. Finally, the methods in [4,32] either update their parameters by a fixed factor during every iteration or by monitoring problem function values over the course of the optimization process.

These conservative approaches will perform well in situations when problem (1.1) is feasible and the initial penalty parameter value is sufficiently small so that obtaining a first-order optimal point for (1.3) immediately yields a first-order optimal point for (1.1). However, in other cases—such as when the initial penalty parameter is too large, the nonlinear optimization problem (1.1) is infeasible, or if a conservative update for μ is slow—they may lead to poor performance. After all, in the context of Algorithm 1, each distinct value for the penalty parameter ρ potentially requires the computational efforts of a complete run of a standard interior-point algorithm. This can be exceedingly expensive if ρ must eventually be reduced to (near) zero.

Our approach is a more adaptive strategy as it considers changes in ρ and μ *during every iteration*. Indeed, this is the only significant change that we suggest from the conservative framework. We apply Algorithm 1, but replace step 5 with a subroutine that (potentially) updates ρ and μ before computing the search direction. In such a context, it is reasonable to leave the primal-dual matrix on the left-hand side of (3.1) as a fixed quantity throughout a given iteration. The values for ρ and μ normally influence the values for H_k, Ω_k, and Γ_k, but since we will be interested in comparing the solutions of (3.1) for various pairs (ρ, μ), it is beneficial to fix these matrices so that only one matrix factorization is required per iteration. (See Sect. 4 for further details on how this can be done.) In this manner, changes in ρ and μ during the search direction computation will only effect the right-hand side of (3.1). There are two key drawbacks to performing the operations in this way, both of which are accounted for in our algorithm. The first drawback is that not having Ω_k and Γ_k depend on μ means that we are in danger of losing the descent property for the merit function $\widetilde{\phi}(x; \rho, \mu)$

proved in Theorem 3.2. We rectify this by enforcing an explicit condition on the directional derivative of this function; see (3.17). The second drawback is that not having H_k depend on ρ may hinder the algorithm from converging quickly on infeasible problems. We rectify this through a condition that may decrease ρ quite rapidly; see (3.18).

During iteration k, we define a set of potential penalty parameter values $\mathcal{R}_k$ and a set of potential interior-point parameters $\mathcal{M}_k$. In $\mathcal{R}_k$ and $\mathcal{M}_k$, respectively, we include the current ρ and μ along with lesser values. (Our approach could include in $\mathcal{R}_k$ and $\mathcal{M}_k$ greater values of each parameter as long as appropriate safeguards are in place; e.g., see [28]. However, since such safeguards are not discussed in this paper, we assume that $\mathcal{R}_k$ and $\mathcal{M}_k$, respectively, do not include values of the parameters greater than the current ρ and μ.) Ideally, the set $\mathcal{R}_k$ would include potential penalty parameter values that are arbitrarily close to zero. We say this as the conditions we outline below may only be satisfied for an arbitrarily small penalty parameter value. For practical purposes, however, it may be detrimental to consider an unlimited number of values, as this may increase the per-iteration computational cost. Overall, for ease of exposition, we assume in the following discussion that $\mathcal{R}_k$ includes arbitrarily small values, but in the statement of our algorithm we include a safeguard so that $\mathcal{R}_k$ need only be a small finite set.

Our strategy involves a series of conditions that must be satisfied by the search direction resulting from a particular choice of $\rho \in \mathcal{R}_k$ and $\mu \in \mathcal{M}_k$. If a certain choice of these parameters results in a search direction satisfying these conditions, then we call such a (ρ, μ) an *admissible pair*. (The definition of an admissible pair changes depending on whether or not the current iterate is sufficiently feasible.) We first find the largest $\rho \in \mathcal{R}_k$ such that, for some $\mu \in \mathcal{M}_k$, the pair (ρ, μ) is admissible. Once this ρ has been found, the penalty parameter is fixed at this value, and we then choose $\mu \in \mathcal{M}_k$ so that the resulting pair (ρ, μ) is still admissible and the resulting search direction satisfies an additional condition.

The cornerstones of our conditions can be summarized as two main ideas: (1) For updating the penalty parameter, *define a quantity with known desirable properties related to the pursuit of primal feasibility* on which a series of updating conditions can be based, and (2) for updating the interior-point parameter, *define a measure related to the pursuit of dual feasibility and complementarity* through which the algorithm can promote long steps and fast convergence. These types of ideas have appeared separately before in the literature, namely in the steering rules for updating a penalty parameter described in [9] and the adaptive barrier rules for updating an interior-point parameter described in [28]. However, to the best of our knowledge, this is the first work in which these distinct strategies have been combined.

For first updating ρ, we define the type of quantity mentioned in the previous paragraph by considering the following linear model of the penalty-interior-point function $\phi(z; \rho, \mu)$ about a given point z:

$$l(\Delta z; \rho, \mu, z) := \phi(z; \rho, \mu) + \nabla\phi(z; \rho, \mu)^T \Delta z.$$

At an iterate z_k, let $\Delta z_k^{\rho,\mu}$ denote the search direction computed by (3.1) for given values of ρ and μ. (Similar notation is also used below for the search direction component

$\Delta x_k^{\rho,\mu}$ and the step in the multipliers $\Delta\lambda_k^{\rho,\mu}$.) The reduction in $l(\Delta z; \rho, \mu, z)$ at z_k for such a computed search direction is then

$$\begin{aligned}\Delta l(\Delta z_k^{\rho,\mu}; \rho, \mu, z_k) &:= l(0; \rho, \mu, z_k) - l(\Delta z_k^{\rho,\mu}; \rho, \mu, z_k) \\ &= -\nabla\phi(z_k; \rho, \mu)^T \Delta z_k^{\rho,\mu}.\end{aligned}$$

This reduction is positive for any nonzero step since (3.1), $\theta(z_k) = 0$, and (3.5) together yield (3.6). In particular, a nonzero solution to (3.1) for $\rho = 0$ yields

$$\Delta l(\Delta z_k^{0,\mu}; 0, \mu, z_k) > 0. \tag{3.14}$$

The fact that we know this quantity is strictly positive is of central importance since we can use it as a measure of the potential progress toward primal feasibility that is possible from the current iterate.

If a given iterate is not sufficiently feasible, then we define an admissible pair (ρ, μ) as any satisfying the conditions outlined in the following bullets.

– We aim to ensure that the progress toward attaining (linearized) primal feasibility is proportional to the level that would be obtained with $\rho = 0$. To this end, along with the linear model of $\phi(z; \rho, \mu)$ defined above, we also define the linear model of the merit function $\widetilde{\phi}(x; \rho, \mu)$ given by

$$\widetilde{l}(\Delta x; \rho, \mu, x) := \widetilde{\phi}(x; \rho, \mu) + \nabla\widetilde{\phi}(x; \rho, \mu)^T \Delta x.$$

A search direction $\Delta x_k^{\rho,\mu}$ computed by (3.1) yields the reduction

$$\begin{aligned}\Delta\widetilde{l}(\Delta x_k^{\rho,\mu}; \rho, \mu, x_k) &:= \widetilde{l}(0; \rho, \mu, x_k) - \widetilde{l}(\Delta x_k^{\rho,\mu}; \rho, \mu, x_k) \\ &= -\nabla\widetilde{\phi}(x_k; \rho, \mu)^T \Delta x_k^{\rho,\mu}.\end{aligned} \tag{3.15}$$

For the value of μ used to compute r_k and s_k, Theorem 3.2 reveals that the model reductions $\Delta l(\Delta z_k^{\rho,\mu}; \rho, \mu, z_k)$ and $\Delta\widetilde{l}(\Delta x_k^{\rho,\mu}; \rho, \mu, x_k)$ coincide. However, for varying μ, these reductions may differ. We can always be sure that (3.14) holds when $\rho = 0$, but it is in fact a reduction in $\widetilde{l}(\Delta x; 0, \mu, x_k)$ that is our best indication of progress toward primal feasibility. Thus, we say an admissible pair (ρ, μ) must be one such that the resulting search direction satisfies

$$\Delta\widetilde{l}(\Delta x_k^{\rho,\mu}; 0, \mu, x_k) \geq \epsilon_1 \Delta l(\Delta z_k^{0,\mu}; 0, \mu, z_k) > 0 \tag{3.16}$$

for some constant $\epsilon_1 \in (0, 1)$. (By Theorem 3.2 and since $\Delta\widetilde{l}(\Delta x_k^{\rho,\mu}; 0, \mu, x_k)$ varies continuously with ρ, (3.16) will be satisfied for μ set as the value used to compute r_k and s_k and for ρ sufficiently small. Thus, (3.16) is satisfiable as long as $\mathcal{M}_k$ includes μ and $\mathcal{R}_k$ includes sufficiently small values.)

– Our conditions should ensure that the search direction is one of sufficient descent for our merit function. This is commonly done by ensuring that the search direction yields a sufficient reduction in a quadratic model of the merit function

(e.g., see [9]), and we adopt such a strategy in our definition of an admissible pair. Applying block elimination to (3.1), we find that an appropriate quadratic model of the merit function $\widetilde{\phi}(x;\rho,\mu)$ is given by

$$\begin{aligned}\widetilde{q}(\Delta x;\rho,\mu,x,\lambda) &:= \widetilde{l}(\Delta x;\rho,\mu,x)\\ &\quad + \tfrac{1}{2}\Delta x^T(H+\nabla c(x)(\Omega^{-1}+\Gamma^{-1})^{-1}\nabla c(x)^T)\Delta x,\end{aligned}$$

and the reduction in this quantity yielded by Δx_k at (x_k,λ_k) is given by

$$\begin{aligned}\Delta\widetilde{q}(\Delta x_k;\rho,\mu,x_k,\lambda_k) &:= \widetilde{q}(0;\rho,\mu,x_k,\lambda_k)-\widetilde{q}(\Delta x_k;\rho,\mu,x_k,\lambda_k)\\ &= \Delta\widetilde{l}(\Delta x_k;\rho,\mu,x_k)\\ &\quad -\tfrac{1}{2}\Delta x_k^T(H_k+\nabla c(x_k)(\Omega_k^{-1}+\Gamma_k^{-1})^{-1}\nabla c(x_k)^T)\Delta x_k.\end{aligned}$$

For the value of μ used to compute r_k and s_k, this reduction will be positive, but for varying μ, this is not a guarantee. Thus, we enforce descent in the merit function $\widetilde{\phi}(z;\rho,\mu)$ by stating that an admissible pair must yield

$$\Delta\widetilde{q}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k,\lambda_k)\geq\epsilon_2\Delta l(\Delta z_k^{0,\mu};0,\mu,z_k)>0 \tag{3.17}$$

where $\epsilon_2\in(0,1)$ is a given constant. (Note that (3.17) will be satisfied for μ set as the value used to compute r_k and s_k and for ρ sufficiently small. Thus, as in the case of (3.16), we have that (3.17) is satisfiable as long as $\mathcal{M}_k$ includes μ and $\mathcal{R}_k$ includes sufficiently small values.)

– If the current iterate satisfies (2.2) sufficiently accurately for $(\rho,\mu)=(0,0)$, then the algorithm is likely converging to an infeasible stationary point. In such cases, we note that in [5], in the context of a penalty-sequential-quadratic-programming method, it is possible to attain superlinear convergence to an infeasible stationary point $(x_\times,s_\times,\lambda_\times)$ if the penalty parameter is driven to zero according to $o(\|(x_k,s_k,\lambda_k)-(x_\times,s_\times,\lambda_\times)\|)$. Thus, with the hope of achieving this behavior, we require that an admissible pair (ρ,μ) has

$$\rho\leq\left\|\begin{bmatrix}\nabla c(x_k)\lambda_k\\ R_k\lambda_k\\ S_k(e-\lambda_k)\end{bmatrix}\right\|^2. \tag{3.18}$$

If the current iterate is not in the neighborhood of an infeasible stationary point, then the right-hand side of (3.18) will be relatively large and the satisfaction of this condition will not impose a decrease in ρ.

In summary, if the current iterate is not sufficiently feasible, then (3.16) ensures progress in linearized feasibility, (3.17) ensures that the computed search direction is one of sufficient descent for our merit function, and (3.18) promotes fast convergence to infeasible stationary points. If, on the other hand, the current iterate is already sufficiently feasible, then we do not enforce any of (3.16), (3.17), or (3.18), but simply require

$$\Delta\widetilde{q}(\Delta x_k^{\rho,\mu};\, \rho, \mu, x_k, \lambda_k) > 0 \tag{3.19}$$

so that the computed direction is again one of descent for $\widetilde{\phi}(x;\, \rho, \mu)$ at x_k.

Let ρ be fixed as the largest value in $\mathcal{R}_k$ such that, for some $\mu \in \mathcal{M}_k$, the pair (ρ, μ) is admissible according to the strategy outlined above. Letting $\mathcal{A}_k$ denote the set of all admissible pairs corresponding to this value of ρ, we then choose μ based on the strategy in the next, and final, bullet.

– In addition to its effect on primal feasibility, a given search direction $(\Delta z, \Delta\lambda)$ will move toward or away from satisfying the dual feasibility and complementarity conditions in (2.2). In particular, the chosen value for μ will effect the progress obtained toward both of these goals. A measure of the effect that a given search direction has on these quantities is the quality function

$$m(\Delta z, \Delta\lambda;\, \rho, \mu, z) := \left\| \begin{bmatrix} \rho\nabla f(x) + \nabla c(x)(\lambda + \Delta\lambda) \\ (R + \Delta R)(\lambda + \Delta\lambda) \\ (S + \Delta S)(e - \lambda - \Delta\lambda) \end{bmatrix} \right\|_\infty . \tag{3.20}$$

We choose μ as the largest value in $\mathcal{M}_k$ such that the computed search direction approximately minimizes this function. Specifically, if $\bar{\mu}$ is the value such that the computed direction minimizes (3.20) over all $\mu \in \mathcal{M}_k$ with $(\rho, \mu) \in \mathcal{A}_k$, then we choose μ as the largest value such that $(\rho, \mu) \in \mathcal{A}_k$ and

$$m(\Delta z_k^{\rho,\mu}, \Delta\lambda_k^{\rho,\mu};\, \rho, \mu, z_k) \leq \epsilon_3 m(\Delta z_k^{\rho,\bar{\mu}}, \Delta\lambda_k^{\rho,\bar{\mu}};\, \rho, \bar{\mu}, z_k), \tag{3.21}$$

where $\epsilon_3 > 1$ is a given constant. Here, it is important that ρ is fixed during the selection of μ, since otherwise the approximate minimization of (3.20) may drive ρ to zero unnecessarily. (If ρ was not fixed, then the quality function may be reduced simply by decreasing ρ and computing smaller multipliers, but such steps do not necessarily indicate progress toward a solution.) We also remark that the strategy in [28] is to choose μ so as to minimize a quality function, whereas our approach is to find a *large* value that only approximately minimizes it. (We found this latter strategy to be more effective in our numerical experiments. If the interior-point parameter was set to $\bar{\mu}$, then we often found that the interior-point parameter was driven to zero too quickly due to only marginal gains in the quality function.)

Our approach can be implemented as Algorithm 1 with step 5 replaced with Algorithm 2 below. The latter algorithm could be fine-tuned and enhanced so that, e.g., search directions for certain pairs of ρ and μ are not calculated if they will not be used. However, for simplicity, we present a straightforward approach that is easier to follow. We also note that, in practice, one should perhaps follow the algorithm in [28] and cut trial search directions by fraction-to-the-boundary rules before evaluating the quantities in (3.16), (3.17), (3.19), and (3.20). For clarity in our presentation, we suppress this detail here, but it is done in our implementation.

Algorithm 2 Search direction computation with adaptive updates

1: (Initialization): Choose updating parameters $\epsilon_1, \epsilon_2 \in (0, 1)$ and $\epsilon_3 > 1$.
2: (Parameter sets selection): Choose $\mathcal{R}_k$ as a finite set of points in $(0, \rho]$ (that includes ρ) and $\mathcal{M}_k$ as a finite set of points in $(0, \mu]$ (that includes μ).
3: (Feasibility direction computations): For all $\mu \in \mathcal{M}_k$, compute $\Delta z_k^{0,\mu}$ by (3.1) with $\rho = 0$.
4: (Feasibility check): If $v(x_k)$ is sufficiently feasible, then go to step 5; else, go to step 6.
5: (Penalty parameter update, feasible case): Let $\mathcal{A}_k$ be the set of admissible pairs, i.e., the pairs (ρ, μ) such that the corresponding solutions $(\Delta z_k^{\rho,\mu}, \Delta\lambda_k^{\rho,\mu})$ to (3.1) satisfy (3.19). If $\mathcal{A}_k = \emptyset$, then maintain the current (ρ, μ) and go to step 8. Otherwise, choose the largest ρ such that $(\rho, \mu) \in \mathcal{A}_k$ for some $\mu \in \mathcal{M}_k$. Go to step 7.
6: (Penalty parameter update, infeasible case): Let $\mathcal{A}_k$ be the set of admissible pairs, i.e., the pairs (ρ, μ) such that (3.18) holds and the corresponding solutions $(\Delta z_k^{\rho,\mu}, \Delta\lambda_k^{\rho,\mu})$ to (3.1) satisfy (3.16) and (3.17). If $\mathcal{A}_k = \emptyset$, then maintain the current (ρ, μ) and go to step 8. Otherwise, choose the largest ρ such that $(\rho, \mu) \in \mathcal{A}_k$ for some $\mu \in \mathcal{M}_k$.
7: (Interior-point parameter update): Set $\bar{\mu}$ as the value in $\mathcal{M}_k$ such that with $\mu = \bar{\mu}$, we have $(\rho, \mu) \in \mathcal{A}_k$ and (3.20) is minimized. Then, set μ as the largest value in $\mathcal{M}_k$ such that $(\rho, \mu) \in \mathcal{A}_k$ and the corresponding solution $(\Delta z_k^{\rho,\mu}, \Delta\lambda_k^{\rho,\mu})$ to (3.1) satisfies (3.21).
8: (Search direction computation): Set $(\Delta z_k, \Delta\lambda_k)$ as the solution to (3.1).

4 Numerical results

Algorithm 1, with and without step 5 replaced by Algorithm 2, has been implemented in Matlab. Henceforth, we refer to our implementation of Algorithm 1 as `PIPAL-c` and our implementation of Algorithm 1 with step 5 replaced by Algorithm 2 as `PIPAL-a`. These acronyms stand for Penalty-Interior-Point ALgorithm,[1] conservative and adaptive. In this section, we discuss details of our implementation and present results for AMPL [15] versions of CUTEr [17,19] problems. We compare the results obtained by `PIPAL-c` and `PIPAL-a` with those obtained by the interior-point algorithm implemented in the well-established `IPOPT` software [37].[2]

4.1 Implementation details

This subsection includes pertinent implementation details, including those related to our termination conditions, technique for computing search directions, incorporation of a second-order correction, method for handling equality constraints, and choice of input parameters. They also include features that have been adopted from the algorithm implemented in `IPOPT` for the purpose of providing a fairer comparison in our numerical experiments.

4.1.1 Termination conditions

`PIPAL-c` and `PIPAL-a` both require checks of the feasibility violation measure $v(x_k)$ and of the first-order optimality conditions (2.2). We define these checks in our implementation in the following manner. First, we observe the condition

[1] In Hindi, pipal (pēˊpəl) is the Sacred Fig, or Bo-Tree, native to southern Asia, and sacred to Buddhists. It is a symbol of happiness, prosperity, longevity, and good luck.

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

$$v(x_k) \leq \epsilon v(x_0) \tag{4.1}$$

where $\epsilon \in (0, 1)$ is a small constant. If this condition is satisfied, then $v(x_k)$ is deemed sufficiently small, since then the level of infeasibility has decreased significantly with respect to the starting point. As for the first-order optimality conditions, we also define a relative tolerance, in this case with respect to the gradient of the penalty subproblem objective at the current iterate:

$$\left\| \begin{bmatrix} \rho \nabla f(x_k) + \nabla c(x_k) \lambda_k \\ R_k \lambda_k - \mu e \\ S_k (e - \lambda_k) - \mu e \end{bmatrix} \right\|_\infty \leq \epsilon \max\{1, \|\rho \nabla f(x_k)\|_\infty\}. \tag{4.2}$$

Equations (2.2) are said to be satisfied sufficiently accurately if (4.2) holds. Note that this condition takes into account the fact that $\theta(z_k) = 0$ for all k.

In summary, conditions (4.1) and (4.2) represent those that are checked for overall termination; see steps 2 and 3 of Algorithm 1. If both conditions hold for the current ρ and $\mu = 0$, then the current point is deemed first-order optimal for (1.1). If (4.1) does not hold, but (4.2) holds for $(\rho, \mu) = (0, 0)$, then the current point is deemed an infeasible stationary point for (1.1). If no point satisfying either of these two sets of conditions has been found, then we terminate after an iteration limit $k_{\max} > 0$. Both (4.1) and (4.2) are also checked in steps 8–9 of Algorithm 1 when considering parameter decreases, and (4.1) is checked in step 4 of Algorithm 2 when determining if the current point is sufficiently feasible.

4.1.2 Computing search directions

The feasibility and search direction computations in `PIPAL-a` are facilitated by noting that from only one matrix factorization and at most three sets of forward and backward solves, the solution of (3.1) for any given pair (ρ, μ) can be obtained. Indeed, if we define $\Delta^{\rho,\mu}$ as the solution to (3.1) for certain values of ρ and μ, then

$$\Delta^{\rho,\mu} = \left(\frac{\rho}{\bar{\rho}} + \frac{\mu}{\bar{\mu}} - 1\right) \Delta^{\bar{\rho},\bar{\mu}} + \left(1 - \frac{\mu}{\bar{\mu}}\right) \Delta^{\bar{\rho},0} + \left(1 - \frac{\rho}{\bar{\rho}}\right) \Delta^{0,\bar{\mu}}. \tag{4.3}$$

Thus, with $\bar{\rho}$ and $\bar{\mu}$ set, respectively, as the values for the penalty and interior-point parameters at the start of the iteration, the construction of $\Delta^{\rho,\mu}$ for any pair (ρ, μ) is performed easily by taking linear combinations of $\Delta^{\bar{\rho},\bar{\mu}}$, $\Delta^{\bar{\rho},0}$, and $\Delta^{0,\bar{\mu}}$. In `PIPAL-a`, for given constants $\kappa_\rho \in (0, 1)$ and $\kappa_\mu \in (0, 1)$ (recall the reduction parameters already specified in Algorithm 1), we form the linear combinations necessary to compute the search directions corresponding to

$$\begin{aligned} \mathcal{R}_k &= \{\kappa_\rho^4 \bar{\rho}, \kappa_\rho^3 \bar{\rho}, \kappa_\rho^2 \bar{\rho}, \kappa_\rho \bar{\rho}, \bar{\rho}\} \\ \text{and } \mathcal{M}_k &= \{\kappa_\mu^4 \bar{\mu}, \kappa_\mu^3 \bar{\mu}, \kappa_\mu^2 \bar{\mu}, \kappa_\mu \bar{\mu}, \bar{\mu}\}. \end{aligned}$$

Recall that if there exists no $(\rho, \mu) \in \mathcal{R}_k \times \mathcal{M}_k$ that is admissible according to the procedure in Algorithm 2, then `PIPAL-a` chooses $(\rho, \mu) = (\bar{\rho}, \bar{\mu})$ for computing

the search direction. In such cases, the search direction computed in PIPAL-a is the one that would have been computed in PIPAL-c.

Along with the comments in the preceding paragraph, we note if the Newton system matrix during iteration k—with H_k set to the Hessian of the Lagrangian as in (3.2)—satisfies Assumption 3.1, then indeed only one matrix factorization is required per iteration. However, if Assumption 3.1 is not satisfied, then we shift the eigenvalues of H_k by adding a multiple of the identity matrix to it so that the resulting system has the correct inertia. The shift is performed with an iterative scheme that first tries a small multiple of the identity, and then increases it exponentially, if necessary, until the Newton matrix in (3.1) has n positive eigenvalues; see [35] and note that a similar strategy has been adopted in IPOPT. This strategy is employed in both PIPAL-c and PIPAL-a.

4.1.3 Second-order corrections

One of the most successful practical enhancements to nonlinear optimization algorithms has been to incorporate a second-order correction step to promote constraint violation decrease; e.g., see [29]. Such a correction is designed to reduce constraint violation by computing a Newton-like step for the constraints at a trial point that fails to satisfy the conditions of the line search. If applied appropriately, second-order corrections can help avoid poor behavior of the algorithm, such as the Maratos effect [26].

Second-order corrections are implemented during the line search in PIPAL-c and PIPAL-a in the following manner. Let Δx_k be the current trial step and let $\bar{\alpha}_k$ be the largest value in $(0, 1]$ such that the fraction-to-the-boundary rules (3.9) are satisfied. If $\bar{\alpha}_k$ does not also satisfy the sufficient decrease condition (3.10) and $v(x_k + \bar{\alpha}_k \Delta x_k) \geq v(x_k)$, then prior to backtracking, we compute $\bar{r}_k$ and $\bar{s}_k$ according to (3.4) with $c(x_k)$ replaced by $c(x_k + \bar{\alpha}_k \Delta x_k)$. A second-order correction step is then obtained by solving

$$\begin{bmatrix} H_k & 0 & 0 & \nabla c(x_k) \\ 0 & \Omega_k & 0 & I \\ 0 & 0 & \Gamma_k & -I \\ \nabla c(x_k)^T & I & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k^{soc} \\ \Delta r_k^{soc} \\ \Delta s_k^{soc} \\ \Delta \lambda_k^{soc} \end{bmatrix} = - \begin{bmatrix} \rho \nabla f(x_k) + \nabla c(x_k) \lambda_k \\ \lambda_k - \mu \bar{R}_k^{-1} e \\ e - \lambda_k - \mu \bar{S}_k^{-1} e \\ 0 \end{bmatrix}. \quad (4.4)$$

Since the algorithm maintains $\theta(z_k) = 0$ for all k, (4.4) is exactly (3.1) with R_k and S_k replaced by $\bar{R}_k$ and $\bar{S}_k$, respectively, on the right-hand side. This means that no additional matrix factorization is required; the cost of computing the second-order correction step is simply the evaluation of $c(x_k + \bar{\alpha}_k \Delta x_k)$ and an additional forward and backward solve with the already computed factors of the Newton system matrix. If the trial point $x_k + \bar{\alpha}_k \Delta x_k + \Delta x_k^{soc}$ satisfies

$$\widetilde{\phi}(x_k + \bar{\alpha}_k \Delta x_k + \Delta x_k^{soc}; \rho, \mu) \leq \widetilde{\phi}(x_k; \rho, \mu) + \alpha_k \eta \nabla \widetilde{\phi}(x_k; \rho, \mu)^T \Delta x_k,$$

then step 6 ends and x_{k+1} in (3.11) is replaced by $x_k + \bar{\alpha}_k \Delta x_k + \Delta x_k^{soc}$. Otherwise, the second-order correction step is discarded and step 6 continues by backtracking along Δx_k from x_k.

A similar second-order correction strategy is implemented in IPOPT. Note, however, that while IPOPT may consider a series of second-order correction steps in a single iteration, PIPAL-c and PIPAL-a consider at most one second-order correction before commencing the backtracking line search.

4.1.4 Handling equality constraints

This paper has described an algorithm for the inequality constrained problem (1.1), but our methods can be applied to problems with equality constraints as well. Two options for incorporating an equality constraint $c^i(x) = 0$ are to split it into the pair of inequalities $c^i(x) \leq 0$ and $c^i(x) \geq 0$, or to introduce a slack variable $a^i \geq 0$, impose the inequalities $-a^i \leq c^i(x) \leq a^i$, and then apply the techniques above for inequality constrained optimization. However, there are drawbacks to both of these approaches. In the former case, the MFCQ fails to hold, meaning that the theoretical foundations in Sect. 2 do not apply (specifically, recall the discussion surrounding Theorem 2.8), and in the latter case, the penalty-interior-point subproblem necessarily involves the definition of even more slack variables, resulting in more degrees of freedom. An alternative approach that involves fewer slack variables is to replace $c^i(x) = 0$ with

$$c^i(x) + a^i - b^i = 0, \quad (a^i, b^i) \geq 0,$$

and then penalize constraint violation through the term $a^i + b^i$. Including these constraints and objective term in the penalty subproblem (1.3) is reasonable as the subproblem again satisfies the MFCQ, but in this case it is not necessary to add more slack variables when deriving the corresponding penalty-interior-point subproblem (since trivial equality constraints can be eliminated). This is the strategy we employ for handling equality constraints.

Following this strategy, we observe that for a given x_k, the subproblem for the slack reset (recall (3.3)) is again convex and separable. In particular, the subproblem corresponding to equality constraint i is

$$\begin{aligned} &\min_{a^i, b^i} \ -\mu(\ln a^i + \ln b^i) + (a^i + b^i) \\ &\text{s.t. } c^i(x_k) + a^i - b^i = 0 \ \ (\text{with } (a^i, b^i) > 0). \end{aligned} \tag{4.5}$$

Thus, in PIPAL-c and PIPAL-a, given x_k and for any $\mu > 0$, the slack variables a^i_k and b^i_k can trivially be set to solve (4.5); i.e., they are set as

$$\begin{aligned} a^i_k &= a^i(x_k; \mu) := \tfrac{1}{2}(\mu - c^i(x_k) + \sqrt{c^i(x_k)^2 + \mu^2}) \\ \text{and} \quad b^i_k &= b^i(x_k; \mu) := \tfrac{1}{2}(\mu + c^i(x_k) + \sqrt{c^i(x_k)^2 + \mu^2}). \end{aligned} \tag{4.6}$$

4.1.5 Features adopted from IPOPT

As our numerical experiments involve comparisons with the interior-point method in IPOPT, we have found it appropriate to make certain implementation choices in

PIPAL-c and PIPAL-a to promote consistency between the three algorithms. The first such features, and perhaps the most important, relate to manipulations of the input data and the initial point given by the user. Since it is an interior-point method, the algorithm in IPOPT may struggle on a problem that does not possess a strict relative interior of the feasible region. To avoid such behavior, IPOPT relaxes constraint bounds to ensure that a strict relative interior exists. The details of this can be found in [37, Sect. 3.5]. Though this type of relaxation is not necessary for PIPAL-c or PIPAL-a, we have adopted this modification of the initial data to have a fairer comparison with IPOPT. Similarly, IPOPT modifies the initial primal point x_k given by the user if it does not lie sufficiently interior to the feasible region. See [37, Sect. 3.6]. We have also adopted this strategy for comparison purposes, even though again it is not necessary for PIPAL-c or PIPAL-a.

One difference between IPOPT and our software in terms of the initial conditions of the algorithm is the choice of the initial dual iterate. IPOPT solves a linear system to compute least-square multipliers at the initial primal point, but in PIPAL-c and PIPAL-a we set the initial multipliers for equality constraints to zero (as they are required to lie in $(-1, 1)$ throughout the solution process) and the initial multipliers for inequality constraints to $\frac{1}{2}$ (as they are required to lie in $(0, 1)$).

The remaining features adopted from IPOPT relate to implementation choices made to avoid numerical issues. First, it has widely been observed that nonlinear optimization codes perform very differently for different scalings of the problem functions, despite the scale invariance of Newton iterations. This is due, for example, by scaling affecting the globalization mechanisms of a given algorithm. In PIPAL-c and PIPAL-a, we aim to avoid these issues by automatically scaling a given problem based on gradient information of the problem functions when evaluated at the initial point. The strategy that we use is exactly that described in [37, Sect. 3.8], i.e., we choose scaling factors so that we never multiply a function by a number larger than one, and ensure that all objective and constraint gradient components are at most a given constant $g_{\max} > 0$ at the initial point. We also enforce that the penalty parameter, interior-point parameter, and all slack variables are not set below given minimum values $\rho_{\min}$, $\mu_{\min}$, and $s_{\min}$, respectively, throughout the optimization process. See [37, eq. (7)] for a similar strategy used in IPOPT when updating μ. This is to avoid numerical errors when solving the Newton systems (3.1) and (4.4), and during the slack resets (3.4) and (4.6). Finally, if a function evaluation error is detected during the line search, then the trial steplength is immediately rejected and the backtracking continues; see [37, pg. 34].

4.1.6 Input parameters

The input parameters for our implementation have been set as those values that we found to yield the best results in our numerical experiments, or, in the cases of ξ^{μ} and $g_{\max}$, have been set as those values used in IPOPT. We chose the initial values for the dynamic parameters ρ and μ both to be 1e-01, which we believe to be standard. As for the remaining input parameters, they are given in Table 1.

Table 1 Input parameter values for PIPAL-c and PIPAL-a

Par.	Val.	Par.	Val.	Par.	Val.	Par.	Val.
τ	1e-02	κ_μ	1e-01	ϵ_2	1e-02	$g_{\max}$	1e+02
η	1e-08	ξ_μ	1+5e-01	ϵ_3	1+1e-02	$\rho_{\min}$	1e-12
γ	5e-01	ω	1e+02	ϵ	1e-06	$\mu_{\min}$	1e-12
κ_ρ	1e-01	ϵ_1	1e-02	$k_{\max}$	1e+03	$s_{\min}$	1e-20

4.2 Experiments with the CUTEr collection

We ran three sets of numerical experiments, each involving the application of PIPAL-c, PIPAL-a, and IPOPT to AMPL versions of CUTEr problems. The set of problems we first considered is freely available.[3] However, due to memory limitations in Matlab, we removed large problems for which the initial Newton system matrix (see (3.1)) contained twenty thousand or more nonzero entries. We also removed unconstrained problems as our software is designed specifically for constrained optimization. All problems were run with AMPL's presolver turned off so as to test the algorithms on difficult constraint sets.

Our first set of experiments involved unadulterated versions of the 438 AMPL models satisfying the criteria in the previous paragraph. The second and third sets of experiments involved degenerate and infeasible variants, respectively, of 125 of these problems, specifically those originally from the Hock–Schittkowski collection [22]. In each set of experiments, we have removed problems for which none of the algorithms was able to terminate successfully.

All experiments were run on an 8-core AMD Opteron 6128 machine with 2.0GHz clock speed running Debian GNU/Linux and Matlab 7.11 (R2010b). The IPOPT results were obtained with version 3.9stable and the linear solver MA27.

4.2.1 Experiments with a set of CUTEr problems

This set of experiments involved 417 problems for which at least one algorithm terminated successfully. The models in this set vary in size from only a few to thousands of variables and constraints. The largest problems solved in terms of the numbers of variables were artif, bdvalue, and morebv, each with $n = 5{,}002$, and the largest problem solved in terms of the numbers of constraints was clnlbeam with $t = 1{,}996$ inequality constraints and 1,000 equality constraints.

The results for this set of experiments are summarized in Fig. 1. In the figure, we provide four performance profiles. The profiles on the top and the bottom-left compare iteration counts for each pair of algorithms; see [27]. For example, in the profile on the top-left, PIPAL-c is "positive" and PIPAL-a is "negative". If, for a particular problem, the "positive" algorithm requires fewer iterations to locate a first-order optimal or infeasible stationary point, then this is represented by a positive-valued bar with value equal to the ratio of iteration counts for the two algorithms. A similarly defined

[3] http://orfe.princeton.edu/~rvdb/ampl/nlmodels/cute/.

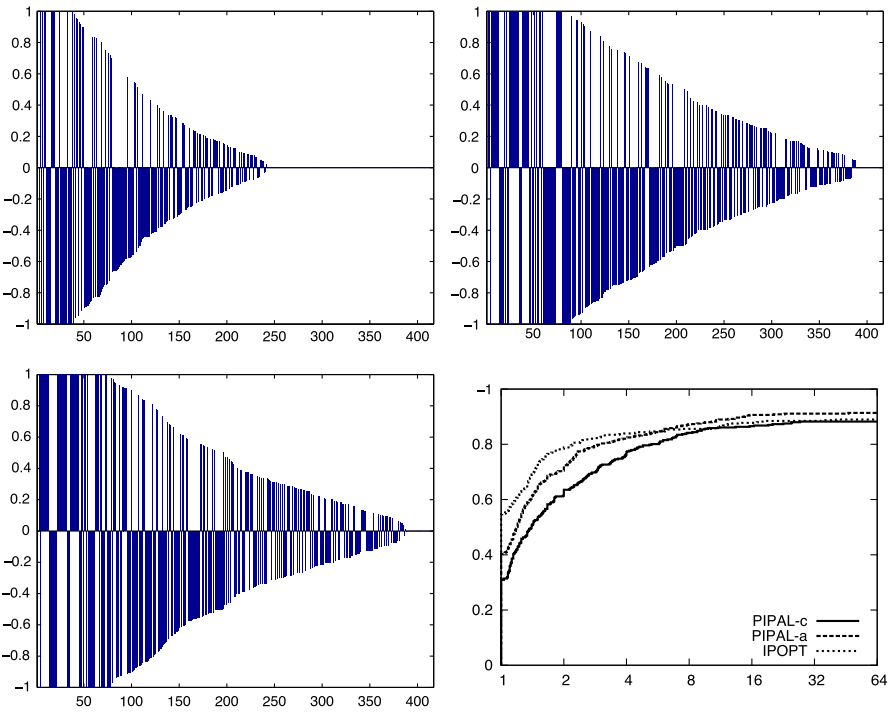

Fig. 1 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on problems from the CUTEr collection

Table 2 "Positive" and "negative" orientations of algorithms in the performance profiles on the top and bottom-left of Figs. 1, 2, 3, and 4

Plot location	"Positive"	"Negative"
Top-left	`PIPAL-c`	`PIPAL-a`
Top-right	`PIPAL-c`	`IPOPT`
Bottom-left	`PIPAL-a`	`IPOPT`

negative-valued bar is given if the "negative" algorithm requires fewer iterations. If one algorithm solved a problem and the other did not, then this is represented by a bar of height 1 or -1. In this representation, one algorithm dominates another by having more (large) bars in its direction, where the bars have been ordered from largest-to-smallest in magnitude for the purpose of clarity in the comparison. Corresponding to Fig. 1, Table 2 lists the characterizations of "positive" and "negative" algorithms in each of the profiles.

The plot provided on the bottom-right of Fig. 1 considers all three algorithms together and has the form of a logarithmic performance profile as proposed in [12]. Here, the leftmost values indicate the proportion of times each algorithm solves a given problem using the least number of iterations. The sum of these values exceeds one as ties are present. The right-most values represent the robustness of each approach; i.e., it provides the percentage of times that the problem is solved. In the overall plot, one

Table 3 Numbers of problems for which PIPAL-c and PIPAL-a yielded each of twelve possible final penalty parameter values (after rounding to the nearest power of 10)

Final ρ	1e-01	1e-02	1e-03	1e-04	1e-05	1e-06
PIPAL-c	316	29	28	24	8	5
PIPAL-a	172	61	45	57	22	45
Final ρ	1e-07	1e-08	1e-09	1e-10	1e-11	1e-12
PIPAL-c	7	9	1	1	1	7
PIPAL-a	9	8	13	0	3	1

algorithm dominates another by having its corresponding line above and to the left of those corresponding to the other algorithms.

The pair-wise comparisons and the logarithmic profile in Fig. 1 suggest that all three algorithms are robust, though PIPAL-a and especially IPOPT have an edge in terms of efficiency. PIPAL-c and PIPAL-a perform nearly identically on a large portion of the problems in the set (note the numerous zero values in the plot in the top-left), which suggests that for many of the problems a decrease in the penalty parameter was not necessary and was not required by the conditions in our dynamic updating strategy. Indeed, in Table 3, we show the numbers of problems for which the final penalty parameter (rounded to the nearest power of 10) was found to be each of the twelve possible values. The table shows that for many problems, PIPAL-a leaves ρ at its initial value, and for a large majority of problems it remains in the range 1e-01 to 1e-06, which is expected for most well-scaled problems. (We remark that a small final penalty parameter value in PIPAL-c or PIPAL-a may indicate that the algorithm has terminated at a sufficiently infeasible, yet somewhat suboptimal solution. However, a similar result is yielded by IPOPT if the Lagrange multipliers become large in norm as the termination conditions in that algorithm are scaled by the value of the average multiplier [37, pg. 28].)

A clearer comparison between the algorithms can be made if we restrict our attention to problems for which PIPAL-c required a decrease in the penalty parameter in order to find a sufficiently accurate optimal solution. In other words, if we restrict our attention to the problems in our set for which the initial penalty parameter was too large, we can pinpoint situations in which the conservative updating strategy in PIPAL-c might have been slow to decrease this parameter.

In this more focused setting, the pair-wise comparisons and the logarithmic profile in Fig. 2 (involving the 132 problems for which a decrease in the penalty parameter was required) now all suggest that PIPAL-c is the weakest algorithm. It is as robust as the other methods, but is clearly less efficient. This provides support for our development of a more adaptive updating strategy for the penalty and interior-point parameters. Indeed, PIPAL-a is a much more efficient algorithm, and is more competitive with the interior-point approach of IPOPT.

These results suggest that with a more sophisticated implementation of PIPAL-a, it has the potential to be an effective general-purpose solver. One of the main challenges in this development is attaining the type of efficiency and robustness results found in these experiments—which in large part are due to adaptive updates of ρ promoting fast convergence to the feasible region—while at the same time ensuring that ρ is not reduced much lower than required to solve problem (1.1).

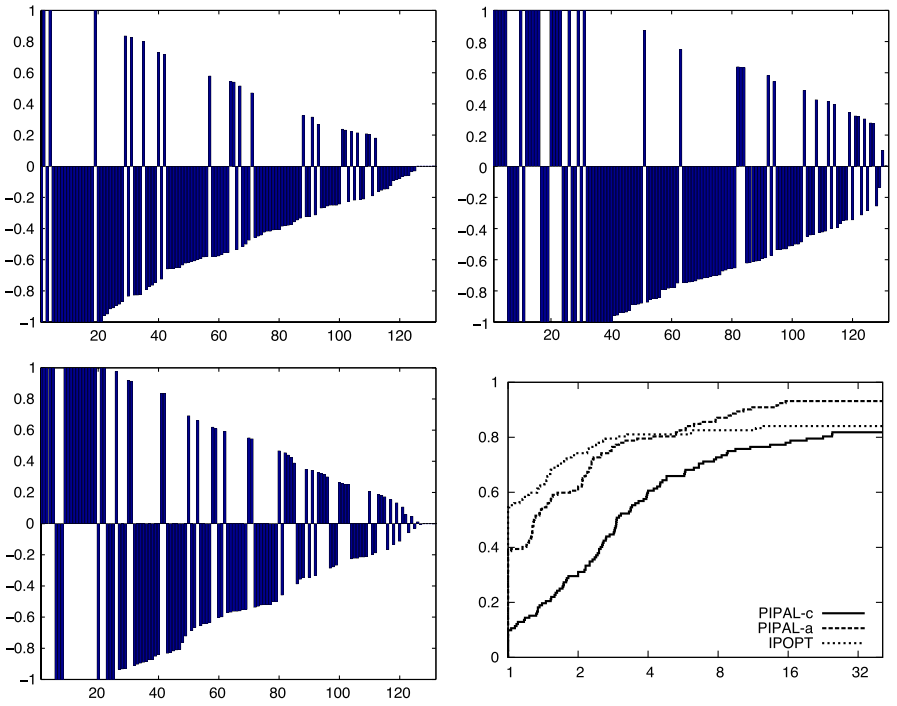

Fig. 2 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on problems from CUTEr for which `PIPAL-c` required a decrease in the penalty parameter

4.2.2 Experiments with degenerate versions of a set of Hock–Schittkowski problems

We ran our second set of numerical experiments with degenerate variants of a subset of problems from Sect. 4.2, namely those in the Hock–Schittkowski set. In particular, we created degenerate instances by manipulating the AMPL models so that for each constraint, $c^i(x) = 0$ or $c^i(x) \leq 0$, we added

$$-c^i(x)^2 \leq 0.$$

These changes have no effect on the feasible region, but clearly make each problem degenerate as constraint gradients for the added active constraints all converge to zero at any solution. Thus, the MFCQ fails at all solution points. We admit that creating instances in this manner only produces a certain type of degeneracy, but these models are sufficient for illustrating the robustness of our software on certain rank-deficient problems.

After solving these models and noting those that were solved by at least one of the algorithms, we obtained a set of 120 problems. The results for these problems are summarized in Fig. 3. (The plots all have similar meanings as in the previous subsection; refer to Table 2.) Clearly, the picture is much different than in Sect. 4.2. `IPOPT` is not only less efficient than both `PIPAL-c` and `PIPAL-a`, but it also lags slightly in terms of robustness. `PIPAL-a` and `PIPAL-c` are competitive, but still we

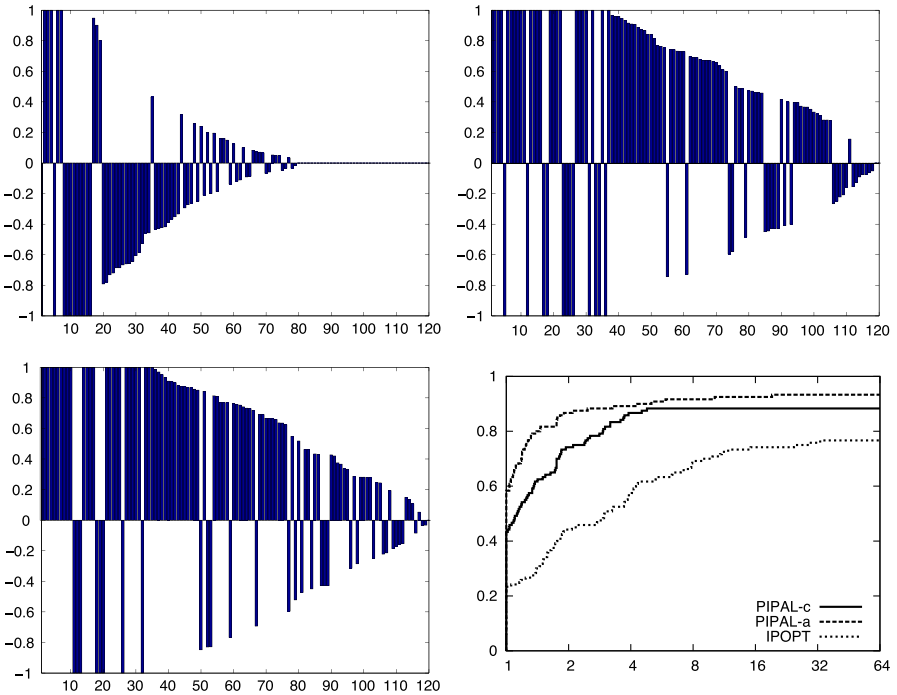

Fig. 3 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on degenerate variants of the Hock–Schittkowski problems from the CUTEr collection

see an advantage in terms of both efficiency and robustness when using the adaptive parameter updates implemented in `PIPAL-a`. It is also worthwhile to note that the robustness of `PIPAL-a` was seemingly unaffected with the addition of redundant (and degeneracy-causing) constraints.

4.2.3 Experiments with infeasible versions of a set of Hock–Schittkowski problems

We ran our third set of numerical experiments with infeasible variants of the Hock–Schittkowski problems. In particular, we created infeasible instances by manipulating the AMPL models so that, for each constraint $c^i(x) = 0$ or $c^i(x) \leq 0$, we added

$$c^i(x)^2 \leq -1.$$

Clearly, this made each problem infeasible with a minimum constraint violation measure equal to 1. Moreover, note that if an algorithm converges to an infeasible stationary point at which an added constraint is active, then the corresponding constraint gradient converges to zero.

After solving these models and only considering those that were solved by at least one of the algorithms, we obtained a set of 105 problems. The results for these problems are summarized in Fig. 4. (The plots all have similar meanings as in the previous

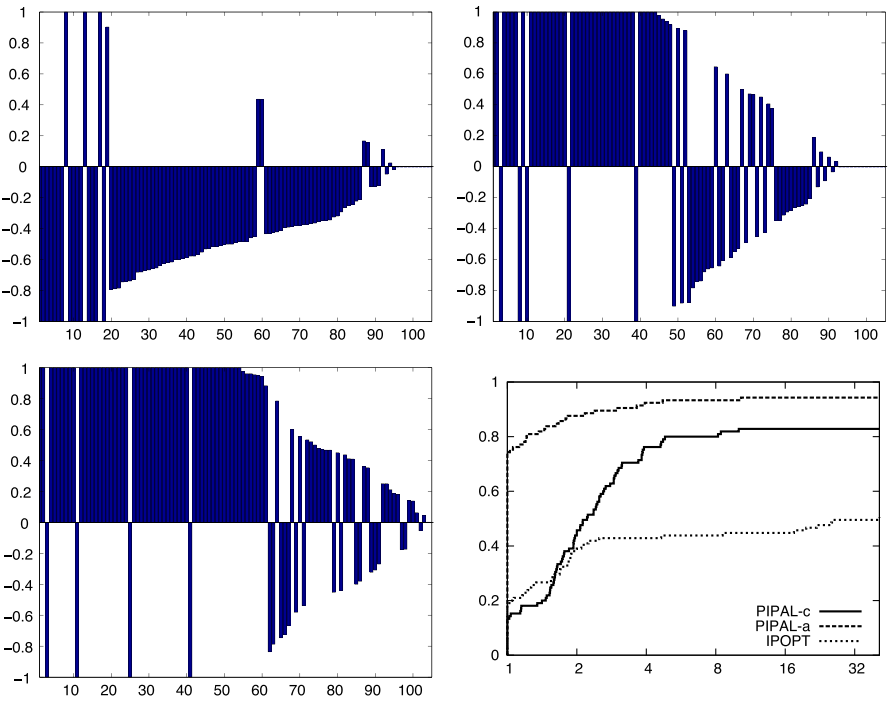

Fig. 4 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on infeasible variants of the Hock–Schittkowski problems from the CUTEr collection

subsections; refer to Table 2.) Again, the picture is much different than in Sect. 4.2. `IPOPT` is as efficient as `PIPAL-c` on a small number of problems, but fails for many of the remaining problems. `PIPAL-a` and `PIPAL-c` are much more robust, providing evidence of the benefits of regularization through penalization. However, again, `PIPAL-a` represents a vast improvement in efficiency over `PIPAL-c`, suggesting that adaptive updates for ρ and μ are beneficial. Finally, we remark that if condition (3.19) was not used in the definition of an admissible pair at infeasible points, then the results were worse in terms of efficiency, providing evidence for our claim that infeasibility detection is enhanced by rapid reduction of the penalty parameter in a neighborhood of an infeasible stationary point.

5 Conclusion

We have developed and tested a penalty-interior-point algorithm for large-scale constrained optimization. As a penalty method, the algorithm is designed to handle problems with difficult constraint sets and infeasible problem instances as comfortably as it handles feasible problems with easier constraints. As an interior-point method, it is designed to have linear system solves, rather than expensive linear or quadratic optimization subproblem solves, as the main computational component per iteration. It has been stressed throughout this work that in order for a method of this type to

be practical as a general-purpose solver, the updates for the penalty and interior-point parameters have to be considered with great care. We believe that the strategy we have developed is novel and sound, and is the main contributor toward the encouraging numerical results that we have obtained, especially for degenerate and infeasible problem instances.

Our experiments support the claim that when an interior-point method is performing at its best, it is difficult to outperform such an approach in terms of efficiency. For one thing, our penalty-interior-point subproblem (1.5) has many more degrees of freedom than (1.4) (i.e., the penalty parameter and the additional slack variables), the presence of which may result in slower convergence. However, the results also highlight two key strengths of our approach: (1) Our algorithm is practically as efficient on most problems, and (2) our algorithm exhibits superior behavior on degenerate and infeasible problems. This suggests that, with a more sophisticated implementation, our algorithm has the potential to be a successful general-purpose solver for large-scale problems. It also suggests that our approach represents an advancement in the development of nonlinear optimization methods that are equally robust and efficient on problems with difficult constraint sets. This latter distinction is especially important in, for two examples, the context of mathematical programs with complementarity constraints [34], a class of problems for which great advancements have been made possible by penalty techniques (e.g., see [23,25]), and the context of mixed-integer nonlinear optimization [21], where a potentially large number of infeasible subproblems can arise in branch-and-bound frameworks (e.g., see [2,33]).

Acknowledgments The author would like to thank Richard A. Waltz for numerous productive conversations about the presented methods and for aiding in the software development. He would also like to thank Jorge Nocedal, whose comments greatly improved the presentation. Finally, the author is indebted to three anonymous referees whose comments and suggestions greatly improved the paper and the implementation.

References

1. Benson, H.Y., Sen, A., Shanno, D.F.: Convergence analysis of an interior-point method for nonconvex nonlinear programming. Optimiz. Methods Softw. submitted (2010)
2. Borchers, B., Mitchell, J.E.: An improved branch and bound algorithm for mixed integer nonlinear programming. Comput. Operat. Res. **21**(4), 359–367 (1994)
3. Breitfeld, M.G., Shanno, D.F.: A globally convergent penalty-barrier algorithm for nonlinear programming and its computational performance. RUTCOR Research Report, RRR 12-94, Rutgers University, New Brunswick, NJ, USA. Technical report (1994)
4. Breitfeld, M.G., Shanno, D.F.: Computational experience with penalty-barrier methods for nonlinear programming. Ann. Operat. Res. **62**, 439–463 (1996)
5. Byrd, R.H., Curtis, F.E., Nocedal, J.: Infeasibility detection and SQP methods for nonlinear optimization. SIAM J. Optimiz. **20**(5), 2281–2299 (2008)
6. Byrd, R.H., Gilbert, J.-Ch., Nocedal, J.: Trust region method based on interior point techniques for nonlinear programming. Math. Programm. **89**(1), 149–185 (2000)
7. Byrd, R.H., Gould, N.I.M., Nocedal, J., Waltz, R.A.: An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. Math. Program. **100**(1), 27–48 (2004)
8. Byrd, R.H., Gould, N.I.M., Nocedal, J., Waltz, R.A.: On the convergence of successive linear-quadratic programming algorithms. SIAM J. Optimiz. **16**(2), 471 (2005)
9. Byrd, R.H., Nocedal, J., Waltz, R.A.: Steering exact penalty methods for nonlinear programming. Optimiz. Methods Softw. **23**(2), 197–213 (2008)
10. Chen, L., Goldfarb, D.: Interior-point L2 penalty methods for nonlinear programming with strong global convergence properties. Math. Program. **108**(1), 1–36 (2006)

11. Chen, L., Goldfarb, D.: On the fast local convergence of interior-point l2 penalty methods for nonlinear programming, technical report. Department of Industrial Engineering and Operations Research, Columbia University, New York, NY, USA. Technical Report x (2006)
12. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. Ser A **91**, 201–213 (2002)
13. Fiacco, A.V., McCormick, G.P.: Nonlinear programming: sequential unconstrained minimization techniques. Classics in Applied Mathematics. SIAM, Philadelphia, PA, USA (1990)
14. Forsgren, A., Gill, P.E., Wright, M.H.: Interior methods for nonlinear optimization. SIAM Rev. **44**(4), 525–597 (2002)
15. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: a modeling language for mathematical programming. Brooks/Cole, Pacific Grove (2002)
16. Goldfarb, D., Polyak, R.A., Scheinberg, K., Yuzefovich, I.: A modified barrier-augmented lagrangian method for constrained minimization. Comput. Optimiz. Appl. **14**, 55–74 (1999)
17. Gould, N.I.M., Bongartz, I., Conn, A.R., Toint, Ph.L.: CUTE: constrained and unconstrained testing environment. ACM Transact. Math. Softw. **21**(1), 123–160 (1995)
18. Gould, N.I.M., Orban, D., Toint, Ph.L.: An interior-point l1-penalty method for nonlinear optimization, technical report. Rutherford Appleton Laboratory, Chilton, Oxfordshire, England. Technical report (2003)
19. Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEr and SifDec: a constrained and unconstrained testing environment, revisited. ACM Transact. Math. Softw. **29**(4), 373–394 (2003)
20. Gould, N.I.M., Robinson, D.P.: A second derivative SQP method: global convergence. SIAM J. Optimiz. submitted (2010)
21. Grossmann, I.E.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optimiz. Eng. **3**(3), 227–252 (2002)
22. Hock, W., Schittkowski, K.: Test examples for nonlinear programming codes. J. Optimiz. Theory Appl. **30**(1), 127–129 (1980)
23. Hu, X., Ralph, D.: Convergence of a penalty method for mathematical programming with complementarity constraints. J. Optimiz. Theory Appl. **123**(2), 365–390 (2004)
24. Jittorntrum, K., Osborne, M.: A modified barrier function method with improved rate of convergence for degenerate problems. J. Aus. Math. Soc. Ser. B **21**, 305–329 (1980)
25. Leyffer, S., Lopez-Calva, G., Nocedal, J.: Interior methods for mathematical programs with complementarity constraints. SIAM J. Optimiz. **17**(1), 52 (2006)
26. Maratos, N.: Exact Penalty Function Algorithms for Finite-Dimensional and Control Optimization Problems. PhD thesis, Department of Computing and Control, University of London (1978)
27. Morales, J.L.: A numerical study of limited memory BFGS methods. Appl. Math. Lett. **15**(4), 481–488 (2002)
28. Nocedal, J., Wächter, A., Waltz, R.A.: Adaptive barrier update strategies for nonlinear interior methods. SIAM J. Optimiz. **19**(4), 1674–1693 (2009)
29. Nocedal, J., Wright, S.J.: Numerical Optimization, Springer Series in Operations Research, 2nd edn. Springer, New York (2006)
30. Polyak, R.A.: Smooth optimization methods for solving nonlinear extremal and equilibrium problems with constraints. In: Eleventh international symposium on mathematical programming, Bonn, Germany (1982)
31. Polyak, R.A.: Modified barrier functions (theory and methods). Math. Program. **54**(1–3), 177–222 (1992)
32. Polyak, R.A.: Primal-dual exterior point method for convex optimization. Optimiz. Methods Softw. **23**(1), 141–160 (2008)
33. Quesada, I., Grossmann, I.E.: An LP/NLP based branch and bound algorithm for convex minlp optimization problems. Comput. Chem. Eng. **16**(10–11), 937–947 (1992)
34. Scheel, H., Scholtes, S.: Mathematical programs with complementarity constraints: stationarity, optimality, and sensitivity. Math. Operat. Res. **25**(1), 1–22 (2000)
35. Vanderbei, R.J., Shanno, D.F.: An interior-point algorithm for nonconvex nonlinear programming. Comput. Optimiz. Appl. **13**, 231–252 (1999)
36. Wächter, A., Biegler, L.T.: Line search filter methods for nonlinear programming: motivation and global convergence. SIAM J. Optimiz. **16**, 1–31 (2005)
37. 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)