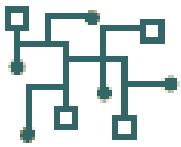

ISE

Industrial and Systems Engineering

Adaptive Augmented Lagrangian Methods for Large-Scale Equality Constrained Optimization

Frank E. Curtis

Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

Hao Jiang and Daniel P. Robinson

Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA

COR@L Technical Report 12T-016

LEHIGH UNIVERSITY

COR@L

COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Noname manuscript No.
(will be inserted by the editor)

Adaptive Augmented Lagrangian Methods for Large-Scale Equality Constrained Optimization

Frank E. Curtis · Hao Jiang · Daniel P. Robinson

December 10, 2012

Abstract We propose an augmented Lagrangian algorithm for solving large-scale equality constrained optimization problems. The novel feature of the algorithm is an adaptive update for the penalty parameter motivated by recently proposed techniques for exact penalty methods. This adaptive updating scheme greatly improves the overall performance of the algorithm without sacrificing the strengths of the core augmented Lagrangian framework, such as its attractive local convergence behavior and ability to be implemented matrix-free. This latter strength is particularly important due to interests in employing augmented Lagrangian algorithms for solving large-scale optimization problems. We focus on a trust region algorithm, but also propose a line search algorithm that employs the same adaptive penalty parameter updating scheme. We provide theoretical results related to the global convergence behavior of our algorithms and illustrate by a set of numerical experiments that they outperform traditional augmented Lagrangian methods in terms of critical performance measures.

Keywords nonlinear optimization · nonconvex optimization · large-scale optimization · augmented Lagrangians · matrix-free methods · steering methods

Mathematics Subject Classification (2010) 49M05 · 49M15 · 49M29 · 49M37 · 65K05 · 65K10 · 65K15 · 90C06 · 90C30 · 93B40

Frank E. Curtis
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA
E-mail: frank.e.curtis@gmail.com

Hao Jiang
Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD
E-mail: hjiang13@jhu.edu

Daniel P. Robinson
Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD
E-mail: daniel.p.robinson@jhu.edu

1 Introduction

Augmented Lagrangian (AL) methods, also known as methods of multipliers, have been instrumental in the development of algorithms for solving constrained optimization problems since the pioneering works by Hestenes [23] and Powell [30] in the late 1960s. Although overshadowed by sequential quadratic optimization and interior-point methods in recent decades, AL methods are experiencing a resurgence as interests grow in solving extremely large-scale problems. The attractive features of AL methods in this regard are that they can be implemented matrix-free [1,4,9,26] and possess fast local convergence guarantees under relatively weak assumptions [15,24]. Moreover, certain AL methods—e.g., the alternating direction method of multipliers (ADMM) [18,21]—have proved to be extremely efficient when solving structured large-scale problems [6,31,33].

Despite these positive features, AL methods have critical disadvantages when they are applied to solve generic problems. In particular, they suffer greatly from poor choices of the initial penalty parameter and Lagrange multipliers. If the penalty parameter is too large and/or the Lagrange multipliers are poor estimates of the optimal multipliers, then one typically finds iterations during which no progress is made in the primal space due to points veering too far away from the feasible region. This leads to wasted computational effort, especially in early iterations. The purpose of this paper is to propose, analyze, and present numerical results for AL methods specifically designed to overcome this disadvantage. We enhance traditional AL approaches with an adaptive penalty parameter update inspired by recently proposed techniques for exact penalty methods [7,8]. The adaptive procedure requires that each trial step yields a sufficiently large reduction in linearized constraint violation, thus promoting consistent progress towards constraint satisfaction. We focus on employing our adaptive updating scheme within a trust region method, but also present a line search algorithm with similar features.

The paper is organized as follows. In §2, we outline a traditional AL algorithm to discuss the inefficiencies that may arise in such an approach. Then, in §3, we present and analyze our adaptive AL trust region method. We show that the algorithm is well-posed and provide results related to its global convergence properties. In §4, we briefly describe an adaptive AL line search method, focusing on the minor modifications that are needed to prove similar convergence results as for our trust region method. In §5 we provide numerical results that illustrate the effectiveness of our adaptive penalty parameter updating strategy within both of our algorithms. Finally, in §6, we summarize and reflect on our proposed techniques.

Additional background on AL methods can be found in [2,3,5,13,17,19]. We also refer the reader to recent work on stabilized sequential quadratic optimization (SQO) methods [16,25,27,28,32], which share similarly attractive local convergence properties with AL methods. In particular, see [20] for a globally convergent AL method that behaves like a stabilized SQO method near primal solutions.

Notation We often drop function dependencies once they are defined and use subscripts to denote the iteration number of an algorithm during which a quantity is computed; e.g., we use x_k to denote the kth algorithm iterate and use $f_k := f(x_k)$ to denote the objective value computed at x_k. We also often use subscripts for constants to indicate the algorithmic quantity to which they correspond; e.g., γ_μ denotes the reduction factor for the penalty parameter μ.

2 A Basic Augmented Lagrangian Algorithm

The algorithms we consider are described in the context of solving the equality constrained optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x)\ \text{ subject to }\ c(x)=0, \tag{2.1}$$

where the objective function $f:\mathbb{R}^n\to\mathbb{R}$ and constraint function $c:\mathbb{R}^n\to\mathbb{R}^m$ are twice continuously differentiable. Defining the Lagrangian for (2.1) as

$$\ell(x,y) := f(x) - c(x)^T y,$$

our algorithms are designed to locate a first-order optimal primal-dual solution of (2.1), namely an ordered pair (x,y) satisfying

$$0 = F_{\text{OPT}}(x,y) := \begin{pmatrix}\nabla_x \ell(x,y)\\ \nabla_y \ell(x,y)\end{pmatrix} = \begin{pmatrix} g(x) - J(x)^T y\\ -c(x)\end{pmatrix}, \tag{2.2}$$

where $g(x) := \nabla f(x)$ and $J(x) := \nabla c(x)^T$. However, if (2.1) is infeasible or constraint qualifications fail to hold at solutions of (2.1), then at least the algorithm is designed to locate a stationary point for

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ v(x),\ \text{ where }\ v(x) := \tfrac{1}{2}\|c(x)\|_2^2, \tag{2.3}$$

namely a point x satisfying

$$0 = F_{\text{FEAS}}(x) := \nabla_x v(x) = J(x)^T c(x). \tag{2.4}$$

If (2.4) holds and $v(x)>0$, then x is as an infeasible stationary point for (2.1).

AL methods aim to solve (2.1), or at least (2.3), by employing unconstrained optimization techniques to minimize a weighted sum of the Lagrangian ℓ and the constraint violation measure v. In particular, scaling ℓ by a penalty parameter $\mu\geq 0$ and adding v, we obtain the augmented Lagrangian

$$\mathcal{L}(x,y,\mu) := \mu\ell(x,y) + v(x) = \mu(f(x) - c(x)^T y) + \tfrac{1}{2}\|c(x)\|_2^2.$$

For future reference, the gradient of the augmented Lagrangian with respect to x evaluated at the point (x,y,μ) is given by

$$\nabla_x \mathcal{L}(x,y,\mu) = \mu(g(x) - J(x)^T\pi(x,y,\mu)),\ \text{ where }\ \pi(x,y,\mu) := y - \tfrac{1}{\mu}c(x). \tag{2.5}$$

A basic AL algorithm proceeds as follows. Given fixed values for the Lagrange multiplier y and penalty parameter μ, the algorithm computes

$$x(y,\mu) := \underset{x\in\mathbb{R}^n}{\text{argmin}}\ \mathcal{L}(x,y,\mu). \tag{2.6}$$

(There may be multiple solutions to the optimization problem in (2.6), or the problem may be unbounded below. However, for simplicity in this discussion, we assume that $x(y,\mu)$ can be computed as a stationary point for $\mathcal{L}(\cdot,y,\mu)$.) The first-order optimality condition for the problem in (2.6) is that $x(y,\mu)$ satisfies

$$\nabla_x \mathcal{L}(x(y,\mu),y,\mu) = 0. \tag{2.7}$$

Inspection of the quantities in (2.2) and (2.7) reveals an important role that may be played by the function π in (2.5). In particular, if $c(x(y,\mu)) = 0$ for $\mu > 0$, then $\pi(x(y,\mu), y, \mu) = y$ and $F_{\text{OPT}}(x(y,\mu), y) = 0$, i.e., $(x(y,\mu), y)$ is a first-order optimal solution of (2.1). For this reason, in a basic AL algorithm, if the constraint violation at $x(y,\mu)$ is sufficiently small, then y is set to $\pi(x,y,\mu)$. Otherwise, if the constraint violation is not sufficiently small, then the penalty parameter is reduced to place a higher priority on minimizing v in subsequent iterations.

Algorithm 1 outlines a complete AL algorithm. The statement of this algorithm may differ in various ways from previously proposed AL methods, but we claim that the algorithmic structure is a good representation of a generic AL method.

Algorithm 1 Basic Augmented Lagrangian Algorithm

1: Choose constants $\{\gamma_\mu, \gamma_t\} \subset (0,1)$.
2: Choose an initial primal-dual pair (x_0, y_0) and initialize $\{\mu_0, t_0\} \subset (0,\infty)$.
3: Set $K \leftarrow 0$ and $j \leftarrow 0$.
4: **loop**
5: **if** $F_{\text{OPT}}(x_K, y_K) = 0$, **then**
6: **return** the first-order solution (x_K, y_K).
7: **if** $\|c_K\|_2 > 0$ and $F_{\text{FEAS}}(x_K) = 0$, **then**
8: **return** the infeasible stationary point x_K.
9: Compute $x(y_K, \mu_K) \leftarrow \text{argmin}_{x\in\mathbb{R}^n}\ \mathcal{L}(x, y_K, \mu_K)$.
10: **if** $\|c(x(y_K,\mu_K))\|_2 \leq t_j$, **then**
11: Set $x_{K+1} \leftarrow x(y_K, \mu_K)$.
12: Set $y_{K+1} \leftarrow \pi(x_{K+1}, y_K, \mu_K)$.
13: Set $\mu_{K+1} \leftarrow \mu_K$.
14: Set $t_{j+1} \leftarrow \gamma_t t_j$.
15: Set $j \leftarrow j + 1$.
16: **else**
17: Set $x_{K+1} \leftarrow x_K$.
18: Set $y_{K+1} \leftarrow y_K$.
19: Set $\mu_{K+1} \leftarrow \gamma_\mu \mu_K$.
20: Set $K \leftarrow K + 1$.

The techniques that we propose in the following sections can be motivated by observing a particular drawback of Algorithm 1, namely the manner in which the penalty parameter μ is updated. In Algorithm 1, μ is updated if and only if the **else** clause in line 16 is reached. This is deemed appropriate since after the augmented Lagrangian was minimized in line 9, the constraint violation was larger than the target value t_j; thus, the algorithm decreases μ to place a higher emphasis on minimizing v in subsequent iterations. Unfortunately, a side effect of this process is that no progress is made in the primal space. Indeed, in such cases, the algorithm sets $x_{K+1} \leftarrow x_K$ and the only result of the iteration—involving the minimization of the (nonlinear) augmented Lagrangian—is that μ is decreased.

The scenario described in the previous paragraph illustrates that a basic AL algorithm may be very inefficient, especially during early iterations when the penalty parameter μ may be too large or the multiplier y is a poor estimate of the optimal multiplier vector. The methods that we propose in the following sections are designed to overcome this potential inefficiency by adaptively updating the penalty parameter *during* the minimization process for the augmented Lagrangian.

We close this section by noting that the minimization in line 9 of Algorithm 1 is itself an iterative process, the iterations of which we refer to as "minor" iterations. This is our motivation for using K as the "major" iteration counter, so as to distinguish it from the iteration counter k used in our methods, which are similar—e.g., in terms of computational cost—to the "minor" iterations in Algorithm 1.

3 An Adaptive Augmented Lagrangian Trust Region Algorithm

In this section, we propose and analyze an AL trust region method with an adaptive updating scheme for the penalty parameter. The new key idea is to measure the improvement towards (linearized) constraint satisfaction obtained by a trial step and compare it to that obtained by a step computed toward minimizing the constraint violation measure v exclusively. If the former improvement is not sufficiently large compared to the latter and the current (nonlinear) constraint violation is not sufficiently small, then the penalty parameter is decreased to place a higher emphasis on minimizing v in the current iteration.

Our strategy involves a set of easily implementable conditions designed around the following models of the constraint violation measure v, Lagrangian ℓ, and augmented Lagrangian $\mathcal{L}$ at x, respectively:

$$q_v(s;x) := \tfrac{1}{2}\|c(x) + J(x)s\|_2^2 \qquad \approx v(x+s); \tag{3.1}$$

$$q_\ell(s;x,y) := \ell(x,y) + \nabla_x \ell(x,y)^T s + \tfrac{1}{2} s^T \nabla_{xx}^2 \ell(x,y) s \qquad \approx \ell(x+s,y); \tag{3.2}$$

$$q(s;x,y,\mu) := \mu q_\ell(s;x,y) + q_v(s;x) \qquad \approx \mathcal{L}(x+s,y,\mu). \tag{3.3}$$

We remark that this approximation to the augmented Lagrangian is not standard. Instead of a second-order Taylor approximation involving second derivatives of the constraint violation measure v, we use the Gauss-Newton model q_v. The motivation for this choice is that our step computation techniques require that the model of the augmented Lagrangian is convex in the limit as $\mu \to 0$; see Lemma 3.2.

Each iteration of our algorithm requires the computation of a trial step s_k toward minimizing the augmented Lagrangian. This step is defined as an approximate solution of the following quadratic subproblem:

$$\underset{s\in\mathbb{R}^n}{\text{minimize}}\ q(s;x_k,y_k,\mu_k) \ \text{ subject to } \ \|s\|_2 \le \Theta_k. \tag{3.4}$$

Here, $\Theta_k > 0$ is a trust-region radius that is set dynamically within the algorithm. Efficient methods for computing the global minimizer of (3.4) are well-known [29, §4.2]. Nonetheless, as we desire matrix-free implementations of our methods, our algorithm only requires an inexact solution of (3.4) as long as it satisfies a condition that promotes convergence; see (3.9a). To this end, we define the Cauchy step

$$\overline{s}_k := \overline{s}(x_k,y_k,\mu_k,\Theta_k) := -\overline{\alpha}(x_k,y_k,\mu_k,\Theta_k)\nabla_x\mathcal{L}(x_k,y_k,\mu_k), \tag{3.5}$$

where

$$\begin{aligned}\overline{\alpha}(x_k,y_k,\mu_k,\Theta_k) := \quad & \underset{\alpha\ge 0}{\text{argmin}} \ \ q(-\alpha\nabla_x\mathcal{L}(x_k,y_k,\mu_k);x_k,y_k,\mu_k) \\ & \text{subject to} \ \ \|\alpha\nabla_x\mathcal{L}(x_k,y_k,\mu_k)\|_2 \le \Theta_k.\end{aligned}$$

Using standard trust-region theory [10], the predicted reduction in $\mathcal{L}(\cdot, y_k, \mu_k)$ from x_k yielded by the step $\overline{s}_k$, i.e.,

$$\Delta q(\overline{s}_k; x_k, y_k, \mu_k) := q(0; x_k, y_k, \mu_k) - q(\overline{s}_k; x_k, y_k, \mu_k),$$

is guaranteed to be positive at any x_k that is not stationary for $\mathcal{L}(\cdot, y_k, \mu_k)$; see Lemma 3.2 for a more precise lower bound for this predicted change. The reduction $\Delta q(s_k; x_k, y_k, \mu_k)$ is defined similarly for the trial step s_k.

Critical to determining whether a computed step s_k yields sufficient progress toward linearized constraint satisfaction from the current iterate, our algorithm also computes a steering step r_k as an approximate solution of the subproblem

$$\underset{r \in \mathbb{R}^n}{\text{minimize}} \; q_v(r; x_k) \;\; \text{subject to} \;\; \|r\|_2 \leq \theta_k, \tag{3.6}$$

where $\theta_k > 0$ is a trust-region radius that is set dynamically within the algorithm. Similar to that for subproblem (3.4), the Cauchy step for subproblem (3.6) is

$$\overline{r}_k := \overline{r}(x_k, \theta_k) := -\overline{\beta}(x_k, \theta_k) J_k^T c_k \tag{3.7}$$

where

$$\overline{\beta}(x_k, \theta_k) := \underset{\beta \geq 0}{\text{argmin}} \;\; q_v(-\beta J_k^T c_k; x_k) \;\; \text{subject to} \;\; \|\beta J_k^T c_k\|_2 \leq \theta_k. \tag{3.8}$$

The predicted reduction in the constraint violation from x_k yielded by $\overline{r}_k$, i.e.,

$$\Delta q_v(\overline{r}_k; x_k) := q_v(0; x_k) - q_v(\overline{r}_k; x_k),$$

is guaranteed to be positive at any x_k that is not stationary for v; see Lemma 3.2. The reduction $\Delta q_v(r_k; x_k)$ is defined similarly for the steering step r_k. We remark upfront that the computation of r_k requires extra effort beyond that for computing s_k, but the expense is minor as r_k can be computed in parallel with s_k and needs only to satisfy a Cauchy decrease condition for (3.6); see (3.9b).

We now describe the kth iteration of our algorithm, specified as Algorithm 2 on page 9. Let (x_k, y_k) be the current primal-dual iterate. We begin by checking whether (x_k, y_k) is a first-order optimal point for (2.1) or if x_k is an infeasible stationary point, and terminate in either case. Otherwise, we enter the **while** loop in line 9 to obtain a value for the penalty parameter for which the gradient of $\mathcal{L}(\cdot, y_k, \mu_k)$ is nonzero. This is appropriate as the purpose of each iteration is to compute a step towards a minimizer of the augmented Lagrangian for $y = y_k$ and $\mu = \mu_k$; if the gradient is zero, then no directions of strict descent for $\mathcal{L}(\cdot, y_k, \mu_k)$ from x_k exist. (Lemma 3.1 shows that this **while** loop terminates finitely.) Next, we enter another loop in line 15 to obtain an approximate solution s_k to problem (3.4) and an approximate solution r_k to subproblem (3.6) that satisfy

$$\Delta q(s_k; x_k, y_k, \mu_k) \geq \kappa_1 \Delta q(\overline{s}_k; x_k, y_k, \mu_k) > 0, \tag{3.9a}$$

$$\Delta q_v(r_k; x_k) \geq \kappa_2 \Delta q_v(\overline{r}_k; x_k), \tag{3.9b}$$

$$\text{and} \;\; \Delta q_v(s_k; x_k) \geq \min\{\kappa_3 \Delta q_v(r_k; x_k), v_k - \tfrac{1}{2}(\kappa_t t_j)^2\}, \tag{3.9c}$$

where $\{\kappa_1, \kappa_2, \kappa_3, \kappa_t\} \subset (0, 1)$ and $t_j > 0$ is the jth target for the constraint violation as in Algorithm 1. Here, the strict inequality in (3.9a) follows from (3.16) since $\nabla \mathcal{L}(x_k, y_k, \mu_k) \neq 0$ by the design of the **while** loop in line 15. In general,

there are many trial and steering steps that satisfy (3.9), but for the purposes of our theoretical analysis it suffices to prove that (3.9) is satisfiable with $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$, as we do in Theorem 3.4.

The conditions in (3.9) can be motivated as follows. Conditions (3.9a) and (3.9b) ensure that the trial step s_k and steering step r_k yield nontrivial decreases in the models of the augmented Lagrangian and the constraint violation, respectively, compared to their Cauchy points. The former requirement is needed to ensure that the augmented Lagrangian will be sufficiently reduced if the trust-region radius is sufficiently small. The motivation for condition (3.9c) is more complex as it involves a minimum of two values on the right-hand side, but this condition is critical as it ensures that the reduction in the constraint violation model is sufficiently large for the trial step. The first quantity on the right-hand side, if it were the minimum of the two, would require the decrease in the model q_v yielded by s_k to be a fraction of that obtained by the steering step r_k; see [7,8] for similar conditions enforced in exact penalty methods. The second quantity is the difference between the current constraint violation and a measure involving a fraction of the target value t_j. Note that this second term allows the minimum to be negative. Therefore, this condition allows for the trial step s_k to predict an increase in the constraint violation, but only if the current constraint violation is sufficiently within the target value t_j. It is worthwhile to note that in general one may consider allowing the penalty parameter to increase as long as the resulting trial step satisfies conditions (3.9a)–(3.9c) and the parameter eventually remains fixed at a small enough value to ensure that constraint violation is minimized. However, as this is only a heuristic and not interesting from a theoretical point of view, we ignore this possibility and simply have the parameter decrease monotonically.

With the trial step s_k in hand, we proceed to compute the ratio

$$\rho_k \leftarrow \frac{\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_k + s_k, y_k, \mu_k)}{\Delta q(s_k; x_k, y_k, \mu_k)} \tag{3.10}$$

of actual-to-predicted decrease in $\mathcal{L}(\cdot, y_k, \mu_k)$. Since $\Delta q(s_k; x_k, y_k, \mu_k)$ is positive by (3.9a), it follows that if $\rho_k \geq \eta_s$, then the augmented Lagrangian has been sufficiently reduced. In such cases, we accept $x_k + s_k$ as the next iterate. Moreover, if we find that $\rho_k \geq \eta_{vs}$ for $\eta_{vs} \geq \eta_s$, then our choice of trust region radius may have been overly cautious so we multiply the upper bound for the trust region radius (δ_k) by $\Gamma_\delta > 1$. If $\rho_k < \eta_s$, then the trust region radius may have been too large, so we counter this by multiplying the upper bound by $\gamma_\delta \in (0, 1)$.

With the primal step updated, next we determine whether to update the multiplier vector. A necessary condition for such an update is that the constraint violation at x_{k+1} must be sufficiently small compared to t_j. If this requirement is met, then we compute any multiplier estimate $\widehat{y}_{k+1}$ that satisfies

$$\|g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}\|_2 \leq \|g_{k+1} - J_{k+1}^T y_k\|_2. \tag{3.11}$$

Of course, computing $\widehat{y}_{k+1}$ as an approximate least-length least-squares multiplier estimate—e.g., by an iterative method—is an attractive option, but for flexibility in the statement of our algorithm we simply enforce (3.11). The second condition that we enforce before an update to the multipliers is considered is that the gradient with respect to x of the Lagrangian (with the multiplier estimate $\widehat{y}_{k+1}$) or of the augmented Lagrangian must be sufficiently small with respect to a target

value $T_j > 0$. If this condition is satisfied, then we proceed to choose new target values $t_{j+1} < t_j$ and $T_{j+1} < T_j$, then set $Y_{j+1} \geq Y_j$ and

$$y_{k+1} \leftarrow (1 - \alpha_y)y_k + \alpha_y \widehat{y}_{k+1}, \tag{3.12}$$

where α_y as the largest value in $[0, 1]$ such that

$$\|(1 - \alpha_y)y_k + \alpha_y \widehat{y}_{k+1}\|_2 \leq Y_{j+1}. \tag{3.13}$$

Note that this updating procedure is well-defined since the choice $\alpha_y \leftarrow 0$ results in $y_{k+1} \leftarrow y_k$, which at least means that (3.13) is satisfiable by this α_y.

For future reference, we define the subset of iterations where line 29 is reached:

$$\mathcal{Y} := \left\{k_j : \|c_{k_j}\|_2 \leq t_j, \min\{\|g_{k_j} - J_{k_j}^T \widehat{y}_{k_j}\|_2, \|\nabla_x \mathcal{L}(x_{k_j}, y_k, \mu_k)\|_2\} \leq T_j\right\}. \tag{3.14}$$

3.1 Well-posedness

We prove that iteration k of Algorithm 2 is well-posed—i.e., either the algorithm will terminate finitely or produce an infinite sequence $\{(x_k, y_k, \mu_k)\}_{k \geq 0}$ of iterates—under the following assumption.

Assumption 3.1 *At x_k, the objective function f and constraint function c are both twice-continuously differentiable.*

Well-posedness in our context requires that the **while** loops in lines 9 and 15 of Algorithm 2 terminate finitely. Our first lemma addresses the first of these loops.

Lemma 3.1 *Suppose Assumption 3.1 holds and that line 9 is reached in Algorithm 2. Then, $\nabla \mathcal{L}(x_k, y_k, \mu) \neq 0$ for all sufficiently small $\mu > 0$.*

Proof Suppose that line 9 is reached and, to reach a contradiction, suppose also that there exists an infinite positive sequence $\{\xi_l\}_{l \geq 0}$ such that $\xi_l \to 0$ and

$$\nabla_x \mathcal{L}(x_k, y_k, \xi_l) = \xi_l(g_k - J_k^T y_k) + J_k^T c_k = 0 \text{ for all } l \geq 0. \tag{3.15}$$

It follows from (3.15) and the fact that $\xi_l \to 0$ that

$$0 = \lim_{l \to \infty} \xi_l(g_k - J_k^T y_k) + J_k^T c_k = J_k^T c_k.$$

If $c_k \neq 0$, then Algorithm 2 would have terminated in line 8; hence, since line 9 is reached, we must have $c_k = 0$. We then may conclude from (3.15) and the fact that $\{\xi_l\}$ is a positive sequence that $g_k - J_k^T y_k = 0$. Combining this with the fact that $c_k = 0$, it follows that (x_k, y_k) is a first-order optimal point for (2.1). However, under these conditions, Algorithm 2 would have terminated in line 6. Overall, we have a contradiction to the existence of the sequence $\{\xi_l\}$, proving the lemma. □

Our goal now is to prove that the **while** loop in line 15 terminates finitely. To prove this, we require the following well-known result; e.g., see [10, Theorem 6.3.1].

Algorithm 2 Adaptive Augmented Lagrangian Trust Region Algorithm

Choose constants $\{\gamma_\mu, \gamma_t, \gamma_T, \gamma_\delta, \kappa_F, \kappa_1, \kappa_2, \kappa_3, \kappa_t, \eta_s, \eta_{vs}\} \subset (0,1)$, $\{\delta, \delta_R, \epsilon, Y\} \subset (0,\infty)$, and $\Gamma_\delta > 1$ such that $\eta_{vs} \geq \eta_s$.
Choose an initial primal-dual pair (x_0, y_0) and initialize $\{\mu_0, t_0, T_0, \delta_0, Y_1\} \subset (0,\infty)$ such that $Y_1 \geq Y$ and $\|y_0\|_2 \leq Y_1$.
Set $k \leftarrow 0$, $k_0 \leftarrow 0$, and $j \leftarrow 1$.
loop
  if $F_{\text{OPT}}(x_k, y_k) = 0$, **then**
    return the first-order solution (x_k, y_k).
  if $\|c_k\|_2 > 0$ and $F_{\text{FEAS}}(x_k) = 0$, **then**
    return the infeasible stationary point x_k.
  while $\nabla_x \mathcal{L}(x_k, y_k, \mu_k) = 0$, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$.
  Set $\theta_k \leftarrow \min\{\delta_k, \delta\|J_k^T c_k\|_2\}$ and $\Theta_k \leftarrow \min\{\delta_k, \delta\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2\}$.
  Compute the Cauchy points $\overline{s}_k$ and $\overline{r}_k$ for (3.4) and (3.6), respectively.
  Compute an approximate solution s_k to (3.4) satisfying (3.9a).
  Compute an approximate solution r_k to (3.6) satisfying (3.9b).
  while $\nabla_x \mathcal{L}(x_k, y_k, \mu_k) = 0$ or (3.9c) is not satisfied, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$ and $\Theta_k \leftarrow \min\{\delta_k, \delta\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2\}$.
    Compute the Cauchy point $\overline{s}_k$ for (3.4).
    Compute an approximate solution s_k to (3.4) satisfying (3.9a).
  Compute ρ_k in (3.10).
  if $\rho_k \geq \eta_{vs}$, **then**
    Set $x_{k+1} \leftarrow x_k + s_k$ and $\delta_{k+1} \leftarrow \max\{\delta_R, \Gamma_\delta \delta_k\}$.
  else if $\rho_k \geq \eta_s$, **then**
    Set $x_{k+1} \leftarrow x_k + s_k$ and $\delta_{k+1} \leftarrow \max\{\delta_R, \delta_k\}$.
  else
    Set $x_{k+1} \leftarrow x_k$ and $\delta_{k+1} \leftarrow \gamma_\delta \delta_k$.
  if $\|c_{k+1}\|_2 \leq t_j$, **then**
    Compute any $\widehat{y}_{k+1}$ satisfying (3.11).
    if $\min\{\|g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}\|_2, \|\nabla_x \mathcal{L}(x_{k+1}, y_k, \mu_k)\|_2\} \leq T_j$, **then**
      Set $k_j \leftarrow k+1$ and $Y_{j+1} \leftarrow \max\{Y, t_{j-1}^{-\epsilon}\}$.
      Set $t_{j+1} \leftarrow \min\{\gamma_t t_j, t_j^{1+\epsilon}\}$ and $T_{j+1} \leftarrow \gamma_T T_j$.
      Set y_{k+1} from (3.12) where α_y yields (3.13).
      Set $j \leftarrow j+1$.
    else
      Set $y_{k+1} \leftarrow y_k$.
  else
    Set $y_{k+1} \leftarrow y_k$.
  Set $\mu_{k+1} \leftarrow \mu_k$.
  Set $k \leftarrow k+1$.

Lemma 3.2 *Suppose Assumption 3.1 holds and that Ω is any value such that*

$$\Omega \geq \max\{\|\mu_k \nabla_{xx}^2 \ell(x_k, y_k) + J_k^T J_k\|_2, \|J_k^T J_k\|_2\}.$$

Then, the Cauchy step for subproblem (3.4) *yields*

$$\Delta q(\overline{s}_k; x_k, y_k, \mu_k) \geq \tfrac{1}{2}\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2 \min\left\{\frac{\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2}{1+\Omega}, \Theta_k\right\} \quad (3.16)$$

and the Cauchy step for subproblem (3.6) *yields*

$$\Delta q_v(\overline{r}_k; x_k) \geq \tfrac{1}{2}\|J_k^T c_k\|_2 \min\left\{\frac{\|J_k^T c_k\|_2}{1+\Omega}, \theta_k\right\}. \quad (3.17)$$

We also require the following result illustrating critical relationships between the quadratic models q_v and q as $\mu \to 0$. The proof of this result reveals our motivation for using the Gauss-Newton model q_v for the constraint violation measure.

Lemma 3.3 *Suppose Assumption 3.1 holds and let*

$$\Theta_k(\mu) := \min\{\delta_k, \delta\|\nabla_x \mathcal{L}(x_k, y_k, \mu)\|_2\}.$$

Then, the following hold true:

$$\lim_{\mu\to 0}\left(\max_{\|s\|_2\le\delta_k} |q(s; x_k, y_k, \mu) - q_v(s; x_k)|\right) = 0, \tag{3.18a}$$

$$\lim_{\mu\to 0} \nabla_x \mathcal{L}(x_k, y_k, \mu) = J_k^T c_k, \tag{3.18b}$$

$$\lim_{\mu\to 0} \overline{s}(x_k, y_k, \mu, \Theta_k(\mu)) = \overline{r}_k, \tag{3.18c}$$

$$\text{and} \quad \lim_{\mu\to 0} \Delta q_v(\overline{s}(x_k, y_k, \mu, \Theta_k(\mu)); x_k) = \Delta q_v(\overline{r}_k; x_k). \tag{3.18d}$$

Proof Since x_k and y_k are fixed, for the purposes of this proof we drop them from all function dependencies. From the definitions of q and q_v, it follows that for some $M > 0$ independent of μ we have

$$\max_{\|s\|_2\le\delta_k} |q(s;\mu) - q_v(s)| = \mu \max_{\|s\|_2\le\delta_k} |q_\ell(s)| \le \mu M.$$

Hence, (3.18a) follows. Similarly, we have

$$\nabla_x \mathcal{L}(\mu) - J_k^T c_k = \mu(g_k - J_k^T y_k),$$

from which it is clear that (3.18b) holds.

We now prove that (3.18c) and (3.18d) hold by considering two cases.

Case 1: Suppose $J_k^T c_k = 0$. This implies $\theta_k = \min\{\delta_k, \delta\|J_k^T c_k\|_2\} = 0$, so $\overline{r}_k = 0$ and $\Delta q_v(\overline{r}_k) = 0$. Moreover, from (3.18b) we have $\Theta_k(\mu) \to 0$ as $\mu \to 0$, which means that $\overline{s}(\mu, \Theta_k(\mu)) \to 0 = \overline{r}_k$ and $\Delta q_v(\overline{s}(\mu, \Theta_k(\mu))) \to 0 = \Delta q_v(\overline{r}_k)$ as $\mu \to 0$. Thus, (3.18c) and (3.18d) both hold.

Case 2: Suppose $J_k^T c_k \neq 0$. We prove (3.18c), after which (3.18d) follows immediately. For a proof by contradiction, suppose (3.18c) does not hold. By (3.18b), we have $\Theta_k(\mu) \to \theta_k$ as $\mu \to 0$, meaning that there exists an infinite positive sequence $\{\xi_l\}_{l\ge 0}$ with $\xi_l \to 0$ and a vector s_* such that

$$\lim_{l\to\infty} \overline{s}(\xi_l, \Theta_k(\xi_l)) = s_* \neq \overline{r}_k \ \text{ and } \ \|s_*\|_2 \le \theta_k. \tag{3.19}$$

Combining this with (3.18b) yields

$$s_* = \lim_{l\to\infty} \overline{s}(\xi_l, \Theta_k(\xi_l)) = \lim_{l\to\infty} -\overline{\alpha}(\xi_l, \Theta_k(\xi_l))\nabla_x \mathcal{L}(\xi_l) = -\alpha_* J_k^T c_k, \tag{3.20}$$

where $\alpha_* := \|s_*\|_2/\|J_k^T c_k\|_2 \ge 0$. The optimality of $\overline{r}_k$ for the convex model q_v within $\{r : \|r\|_2 \le \theta_k\}$ and along the non-zero strict descent direction $-J_k^T c_k$ may be combined with (3.19) and (3.20) to conclude that

$$q_v(s_*) > q_v(\overline{r}_k). \tag{3.21}$$

On the other hand, it follows from the definition of the Cauchy point $\bar{s}(\xi_l, \Theta_k(\xi_l))$ and (3.18b) that there exists a positive function $\alpha : \mathbb{R} \to \mathbb{R}$ such that

$$\lim_{l\to\infty} \alpha(\xi_l) = \|\bar{r}_k\|_2 / \|J_k^T c_k\|_2, \tag{3.22a}$$

$$\|\alpha(\xi_l)\nabla_x \mathcal{L}(\xi_l)\|_2 \le \Theta(\xi_l) \qquad \text{for all } l \ge 0, \tag{3.22b}$$

$$\text{and } q(\bar{s}(\xi_l, \Theta_k(\xi_l)); \xi_l) \le q(-\alpha(\xi_l)\nabla_x \mathcal{L}(\xi_l); \xi_l) \qquad \text{for all } l \ge 0. \tag{3.22c}$$

Taking the limit of (3.22c) as $l \to \infty$ and using (3.18a), (3.20), (3.18b), and (3.22a), we have

$$\begin{aligned} q_v(s_*) &= \lim_{l\to\infty} q(\bar{s}(\xi_l, \Theta_k(\xi_l)); \xi_l) \le \lim_{l\to\infty} q(-\alpha(\xi_l)\nabla_x \mathcal{L}(\xi_l); \xi_l) \\ &= q_v\left(-\frac{\|\bar{r}_k\|_2}{\|J_k^T c_k\|_2} J_k^T c_k\right) = q_v(\bar{r}_k), \end{aligned}$$

which contradicts (3.21). Thus, (3.18c) and (3.18d) follow.

We have proved that (3.18) holds in all cases. □

In the following theorem, we combine the lemmas above to prove that Algorithm 2 is well-posed.

Theorem 3.4 *Suppose Assumption 3.1 holds. Then, the kth iteration of Algorithm 2 is well-posed. That is, either the algorithm will terminate in line 6 or 8, or it will compute* $\mu_k > 0$ *such that* $\nabla_x \mathcal{L}(x_k, y_k, \mu_k) \neq 0$ *and for the steps* $s_k = \bar{s}_k$ *and* $r_k = \bar{r}_k$ *the conditions in* (3.9) *will be satisfied, in which case* $(x_{k+1}, y_{k+1}, \mu_{k+1})$ *will be computed.*

Proof If in the kth iteration Algorithm 2 terminates in line 6 or 8, then there is nothing to prove. Therefore, for the remainder of the proof, we assume that line 9 is reached. Lemma 3.1 then ensures that

$$\nabla \mathcal{L}(x_k, y_k, \mu) \neq 0 \text{ for all } \mu > 0 \text{ sufficiently small.} \tag{3.23}$$

Consequently, the **while** loop in line 9 will terminate for a sufficiently small $\mu_k > 0$.

By construction, conditions (3.9a) and (3.9b) are satisfied for any $\mu_k > 0$ by $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$. Thus, all that remains is to show that for a sufficiently small $\mu_k > 0$, (3.9c) is also satisfied by $s_k = \bar{s}_k$ and $r_k = \bar{r}_k$. From (3.18d), we have that

$$\lim_{\mu\to 0} \Delta q_v(s_k; x_k) = \lim_{\mu\to 0} \Delta q_v(\bar{s}_k; x_k) = \Delta q_v(\bar{r}_k; x_k) = \Delta q_v(r_k; x_k). \tag{3.24}$$

If $\Delta q_v(r_k; x_k) > 0$, then (3.24) implies that (3.9c) will be satisfied for sufficiently small $\mu_k > 0$. On the other hand, suppose

$$\Delta q_v(r_k; x_k) = \Delta q_v(\bar{r}_k; x_k) = 0, \tag{3.25}$$

which along with (3.17) means that $J_k^T c_k = 0$. If $c_k \neq 0$, then Algorithm 2 would have terminated in line 8 and, therefore, we must have $c_k = 0$. This and (3.25) imply that

$$\min\{\kappa_3 \Delta q_v(r_k; x_k), v_k - \tfrac{1}{2}(\kappa_t t_j)^2\} = -\tfrac{1}{2}(\kappa_t t_j)^2 < 0 \tag{3.26}$$

since $t_j > 0$ by construction and $\kappa_t \in (0, 1)$ by choice. Therefore, we can deduce that (3.9c) will be satisfied for sufficiently small $\mu_k > 0$ by observing (3.24), (3.25) and (3.26). Combining this with (3.23) and the fact that the **while** loop on line 15 ensures that μ_k will eventually be as small as required, guarantees that the **while** loop will terminate finitely. This completes the proof as all remaining steps in the kth iteration involve explicit computations. □

3.2 Global Convergence

We analyze global convergence properties of Algorithm 2 under the assumption that the algorithm does not terminate finitely. That is, in this section we assume that neither a first-order optimal solution nor an infeasible stationary point is found so that the sequence $\{(x_k, y_k, \mu_k)\}_{k\geq 0}$ is infinite.

We provide global convergence guarantees under the following assumption.

Assumption 3.2 *The iterates $\{x_k\}_{k\geq 0}$ are contained in a convex compact set over which the objective function f and constraint function c are both twice-continuously differentiable.*

This assumption and the bound on the multipliers enforced in Algorithm 2 imply that there exists a positive monotonically increasing sequence $\{\Omega_j\}_{j\geq 0}$ such that for all $k_j \leq k < k_{j+1}$ we have

$$\|\nabla^2_{xx}\mathcal{L}(\omega, y_k, \mu_k)\|_2 \leq \Omega_j \text{ for all } \omega \text{ on the segment } [x_k, x_k + s_k], \tag{3.27a}$$

$$\|\mu_k \nabla^2_{xx}\ell(x_k, y_k) + J_k^T J_k\|_2 \leq \Omega_j, \tag{3.27b}$$

$$\text{and } \|J_k^T J_k\|_2 \leq \Omega_j. \tag{3.27c}$$

We begin our analysis in this section by proving the following lemma, which provides critical bounds on differences in (components of) the augmented Lagrangian summed over sequences of iterations.

Lemma 3.5 *Suppose Assumption 3.2 holds. Then, the following hold true.*

(i) If $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, then there exist positive constants M_f, M_c, and $M_{\mathcal{L}}$ such that for all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1} \mu_k(f_k - f_{k+1}) < M_f, \tag{3.28}$$

$$\sum_{k=0}^{p-1} \mu_k y_k^T(c_{k+1} - c_k) < M_c, \tag{3.29}$$

$$\text{and } \sum_{k=0}^{p-1} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) < M_{\mathcal{L}}. \tag{3.30}$$

(ii) If $\mu_k \to 0$, then the sums

$$\sum_{k=0}^{\infty} \mu_k(f_k - f_{k+1}), \tag{3.31}$$

$$\sum_{k=0}^{\infty} \mu_k y_k^T(c_{k+1} - c_k), \tag{3.32}$$

$$\text{and } \sum_{k=0}^{\infty} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) \tag{3.33}$$

converge and are finite, and

$$\lim_{k\to\infty} \|c_k\|_2 = c_L \text{ for some } c_L \geq 0. \tag{3.34}$$

Proof Under Assumption 3.2 we may conclude that for some constant $\overline{M}_f > 0$ and all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1}(f_k - f_{k+1}) = f_0 - f_p < \overline{M}_f.$$

If $\mu_k = \mu$ for all sufficiently large k, then this implies that (3.28) clearly holds for some sufficiently large M_f. Otherwise, if $\mu_k \to 0$, then it follows from Dirichlet's Test [11, §3.4.10] and the fact that $\{\mu_k\}_{k\geq 0}$ is a monotonically decreasing sequence that converges to zero that (3.31) converges and is finite.

Next, we show that for some constant $\overline{M}_c > 0$ and all integers $p \geq 1$ we have

$$\sum_{k=0}^{p-1} y_k^T(c_{k+1} - c_k) < \overline{M}_c. \tag{3.35}$$

First, suppose that $\mathcal{Y}$ defined in (3.14) is finite. It follows that there exists $k' \geq 0$ and y such that $y_k = y$ for all $k \geq k'$. Moreover, under Assumption 3.2 there exists a constant $\widehat{M}_c > 0$ such that for all $p \geq k' + 1$ we have

$$\sum_{k=k'}^{p-1} y_k^T(c_{k+1} - c_k) = y^T \sum_{k=k'}^{p-1}(c_{k+1} - c_k) = y^T(c_p - c_{k'}) \leq \|y\|_2\|c_p - c_{k'}\|_2 < \widehat{M}_c.$$

It is now clear that (3.35) holds in this case. Second, suppose that $|\mathcal{Y}| = \infty$ so that the sequence $\{k_j\}_{j\geq 0}$ in Algorithm 2 is infinite. By construction we have $t_j \to 0$, so for some $j' \geq 1$ we have

$$t_j = t_{j-1}^{1+\epsilon} \text{ and } Y_{j+1} = t_{j-1}^{-\epsilon} \text{ for all } j \geq j'. \tag{3.36}$$

From the definition of the sequence $\{k_j\}_{j\geq 0}$, we know that

$$\begin{aligned}\sum_{k=k_j}^{k_{j+1}-1} y_k^T(c_{k+1} - c_k) = y_{k_j}^T \sum_{k=k_j}^{k_{j+1}-1}(c_{k+1} - c_k) &= y_{k_j}^T(c_{k_{j+1}} - c_{k_j}) \\ &\leq \|y_{k_j}\|_2\|c_{k_{j+1}} - c_{k_j}\|_2 \leq 2Y_{j+1}t_j = 2t_{j-1} \text{ for all } j \geq j'\end{aligned}$$

where the last inequality follows from (3.36). Using these relationships, summing over all $j' \leq j \leq j' + q$ for an arbitrary integer $q \geq 1$, and using the fact that $t_{j+1} \leq \gamma_t t_j$ by construction, leads to

$$\begin{aligned}\sum_{j=j'}^{j'+q}\left(\sum_{k=k_j}^{k_{j+1}-1} y_k^T(c_{k+1} - c_k)\right) &\leq 2\sum_{j=j'}^{j'+q} t_{j-1} \\ &\leq 2t_{j'-1}\sum_{l=0}^{q}\gamma_t^l = 2t_{j'-1}\frac{1-\gamma_t^{q+1}}{1-\gamma_t} \leq \frac{2t_{j'-1}}{1-\gamma_t}.\end{aligned}$$

It is now clear that (3.35) holds in this case as well.

We have shown that (3.35) always holds. Thus, if $\mu_k = \mu$ for all sufficiently large k, then (3.29) holds for some sufficiently large M_c. Otherwise, if $\mu_k \to 0$, then it follows from Dirichlet's Test [11, §3.4.10], (3.35) and the fact that $\{\mu_k\}_{k\geq 0}$ is

a monotonically decreasing sequence that converges to zero that (3.32) converges and is finite.

Finally, observe that

$$\begin{aligned}
&\sum_{k=0}^{p-1} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \\
&= \sum_{k=0}^{p-1} \mu_k (f_k - f_{k+1}) + \sum_{k=0}^{p-1} \mu_k y_k^T (c_{k+1} - c_k) + \tfrac{1}{2} \sum_{k=0}^{p-1} (\|c_k\|_2^2 - \|c_{k+1}\|_2^2) \\
&= \sum_{k=0}^{p-1} \mu_k (f_k - f_{k+1}) + \sum_{k=0}^{p-1} \mu_k y_k^T (c_{k+1} - c_k) + \tfrac{1}{2} (\|c_0\|_2^2 - \|c_p\|_2^2). \qquad (3.37)
\end{aligned}$$

If $\mu_k = \mu$ for all sufficiently large k, then it follows from Assumption 3.2, (3.28), (3.29), and (3.37) that (3.30) will hold for some sufficiently large $M_{\mathcal{L}}$. Otherwise, consider when $\mu_k \to 0$. Taking the limit of (3.37) as $p \to \infty$, we have from Assumption 3.2 and conditions (3.31) and (3.32) that

$$\sum_{k=0}^{\infty} (\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k)) < \infty.$$

Since the terms in this sum are all nonnegative, it follows from the Monotone Convergence Theorem that (3.33) converges and is finite. Moreover, we may again take the limit of (3.37) as $p \to \infty$ and use (3.31), (3.32), and (3.33) to conclude that (3.34) holds. □

In the following subsections, we consider different situations depending on the number of times that the Lagrange multiplier vector is updated.

3.2.1 Finite number of multiplier updates

In this section, we consider cases when $\mathcal{Y}$ in (3.14) is finite. In such cases, the counter j in Algorithm 2, which tracks the number of times that the dual vector is updated, satisfies

$$j \in \{1, 2, \ldots \bar{j}\} \text{ for some finite } \bar{j}. \qquad (3.38)$$

For the purposes of our analysis in this section, we define

$$t := t_{\bar{j}} > 0 \text{ and } T := T_{\bar{j}} > 0. \qquad (3.39)$$

We consider two subcases depending on whether the penalty parameter stays bounded away from zero, or if it converges to zero. The following lemma concerns cases when the penalty parameter stays bounded away from zero. This is possible, for example, if the algorithm converges to an infeasible stationary point that is also stationary for the augmented Lagrangian.

Lemma 3.6 *Suppose Assumption 3.2 holds. Then, if $|\mathcal{Y}| < \infty$ and $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, then with t defined in* (3.39) *there exist a vector y and a scalar $\overline{k} \geq 0$ such that*

$$y_k = y \text{ and } \|c_k\|_2 \geq t \text{ for all } k \geq \overline{k}. \qquad (3.40)$$

Moreover, we have the limits

$$\lim_{k\to\infty} \|J_k^T c_k\|_2 = 0 \ \text{ and } \ \lim_{k\to\infty} (g_k - J_k^T y) = 0. \tag{3.41}$$

Therefore, every limit point of $\{x_k\}_{k\geq 0}$ *is an infeasible stationary point.*

Proof Since $|\mathcal{Y}| < \infty$, we know that (3.38) and (3.39) both hold for some $\bar{j} \geq 0$. Since we also suppose that $\mu_k = \mu > 0$ for all sufficiently large k, it follows by construction in Algorithm 2 that there exists y and a scalar $k' \geq k_{\bar{j}}$ such that

$$\mu_k = \mu \ \text{ and } \ y_k = y \ \text{ for all } k \geq k'. \tag{3.42}$$

Consequently, it follows with very minor modifications to the statements and proofs of [10, Theorems 6.4.3, 6.4.5, 6.4.6] that

$$\lim_{k\to\infty} \nabla_x \mathcal{L}(x_k, y_k, \mu_k) = \lim_{k\to\infty} \nabla_x \mathcal{L}(x_k, y, \mu) = 0, \tag{3.43}$$

which implies that

$$\lim_{k\to\infty} \|s_k\|_2 \leq \lim_{k\to\infty} \Theta_k = \lim_{k\to\infty} \min\{\delta_k, \delta \|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2\} = 0. \tag{3.44}$$

From (3.43), it follows that there exists $\overline{k} \geq k'$ such that $\|c_k\|_2 \geq t$ for all $k \geq \overline{k}$, or else for some $k \geq \overline{k}$ the algorithm would set $j \leftarrow \bar{j} + 1$, violating (3.38). Thus, we have shown that (3.40) holds.

We now turn to the limits in (3.41). From (3.44) and Assumption 3.2, we have

$$\lim_{k\to\infty} \Delta q_v(s_k; x_k) = 0. \tag{3.45}$$

We then have from the definition of v and (3.40) that

$$v_k - \tfrac{1}{2}(\kappa_t t_k)^2 \geq \tfrac{1}{2} t^2 - \tfrac{1}{2}(\kappa_t t)^2 = \tfrac{1}{2}(1 - \kappa_t^2) t^2 > 0 \ \text{ for all } \ k \geq \overline{k}. \tag{3.46}$$

We now prove that $J_k^T c_k \to 0$. To see this, first note that

$$\nabla_x \mathcal{L}(x_k, y, \mu) \neq 0 \ \text{ for all } k \geq \overline{k},$$

or else the algorithm would set $\mu_{k+1} < \mu$ in line 10, violating (3.42). We may then use [10, Theorem 6.4.2] to deduce that there must be infinitely many successful iterations, which we denote by $\mathcal{S}$, for $k \geq \overline{k}$. If $J_k^T c_k \nrightarrow 0$, then for some $\zeta > 0$ there exists an infinite subsequence

$$\mathcal{S}_\zeta = \{k \in \mathcal{S} : k \geq \overline{k} \ \text{ and } \ \|J_{k+1}^T c_{k+1}\|_2 \geq \zeta\}.$$

We may then observe the updating strategies for θ_k and δ_k to conclude that

$$\begin{aligned} \theta_{k+1} &= \min\{\delta_{k+1}, \delta \|J_{k+1}^T c_{k+1}\|_2\} \\ &\geq \min\{\max\{\delta_R, \delta_k\}, \delta\zeta\} \geq \min\{\delta_R, \delta\zeta\} > 0 \ \text{ for all } \ k \in \mathcal{S}_\zeta. \end{aligned} \tag{3.47}$$

Using (3.9b), (3.17), (3.27c), (3.38), and (3.47), we then find for $k \in \mathcal{S}_\zeta$ that

$$\Delta q_v(r_{k+1}; x_{k+1}) \geq \kappa_2 \Delta q_v(\overline{r}_{k+1}; x_{k+1}) \geq \tfrac{1}{2}\kappa_2 \zeta \min\left\{ \frac{\zeta}{1 + \Omega_{\bar{j}}}, \delta_R, \delta\zeta \right\} =: \zeta' > 0. \tag{3.48}$$

We may now combine (3.48), (3.46), and (3.45) to state that (3.9c) must be violated for sufficiently large $(k-1) \in \mathcal{S}_\zeta$ and, consequently, the penalty parameter will be decreased. However, this is a contradiction to (3.42), so we conclude that $J_k^T c_k \to 0$. The fact that every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point follows since $\|c_k\|_2 \geq t$ for all $k \geq \overline{k}$ in (3.40) and $J_k^T c_k \to 0$ in (3.41).

The latter limit in (3.41) follows from the former limit in (3.41) and (3.42). □

Our second lemma considers cases when μ converges to zero.

Lemma 3.7 *Suppose Assumption 3.2 holds. Then, if $|\mathcal{Y}| < \infty$ and $\mu_k \to 0$, then there exist a vector y and scalar $\overline{k} \geq 0$ such that*

$$y_k = y \;\; \text{for all} \;\; k \geq \overline{k}. \tag{3.49}$$

Moreover, for some constant $c_L > 0$, we have the limits

$$\lim_{k\to\infty} \|c_k\|_2 = c_L > 0 \;\; \text{and} \;\; \lim_{k\to\infty} \|J_k^T c_k\|_2 = 0. \tag{3.50}$$

Therefore, every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point.

Proof Since $|\mathcal{Y}| < \infty$, we know that (3.38) and (3.39) both hold for some $\bar{j} \geq 0$ and, as in the proof of Lemma 3.6, it follows that (3.49) holds for some $\overline{k} \geq k_{\bar{j}}$.

From (3.34), it follows that $\|c_k\|_2 \to c_{\text{L}}$ for some $c_{\text{L}} \geq 0$. If $c_{\text{L}} = 0$, then by Assumption 3.2, the definition of $\nabla_x \mathcal{L}$, (3.49), and the fact that $\mu_k \to 0$ it follows that $J_k^T c_k \to \nabla_x \mathcal{L}(x_k, y, \mu_k) \to 0$. As in the proof of Lemma 3.6, this would imply that for some $k \geq \overline{k}$ the algorithm would set $j \leftarrow \bar{j} + 1$, violating (3.38). Thus, we conclude that $c_{\text{L}} > 0$, which proves the first part of (3.50).

Next, we prove that

$$\liminf_{k\geq 0} \|J_k^T c_k\|_2 = 0. \tag{3.51}$$

If (3.51) does not hold, then there exists $\zeta > 0$ and $k' \geq \overline{k}$ such that

$$\|J_k^T c_k\|_2 \geq 2\zeta \;\; \text{for all} \;\; k \geq k'. \tag{3.52}$$

Hence, by (3.52) and the fact that $\mu_k \to 0$, there exists $k'' \geq k'$ such that

$$\|\nabla_x \mathcal{L}(x_k, y, \mu_k)\|_2 \geq \zeta \;\; \text{for all} \;\; k \geq k''. \tag{3.53}$$

We now show that the trust region radius Θ_k is bounded away from zero for $k \geq k''$. In order to see this, suppose that for some $k \geq k''$ we have

$$0 < \Theta_k \leq \min\left\{ \frac{\zeta}{1+\Omega_{\bar{j}}}, \frac{(1-\eta_{vs})\kappa_1 \zeta}{1+2\Omega_{\bar{j}}} \right\} =: \delta_{\text{thresh}} \tag{3.54}$$

where Ω_j appears in (3.27). It then follows from (3.9a), (3.16), (3.27b), (3.53), and (3.54) that

$$\Delta q(s_k; x_k, y, \mu_k) \geq \kappa_1 \Delta q(\overline{s}_k; x_k, y, \mu_k) \geq \tfrac{1}{2}\kappa_1 \zeta \min\left\{ \frac{\zeta}{1+\Omega_{\bar{j}}}, \Theta_k \right\} \geq \tfrac{1}{2}\kappa_1 \zeta \Theta_k. \tag{3.55}$$

Using the definition of q, Taylor's Theorem, (3.27a), (3.27b), and the trust-region constraint, we may conclude that for some ω_k on the segment $[x_k, x_k + s_k]$ we have

$$\begin{aligned} & |q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)| \\ = \; & \tfrac{1}{2}|s_k^T(\mu_k \nabla^2_{xx}\ell(x_k, y_k) + J_k^T J_k)s_k - s_k^T \nabla^2_{xx}\mathcal{L}(\omega_k, y, \mu_k)s_k| \\ \leq \; & \Omega_{\bar{j}}\|s_k\|_2^2 \leq \Omega_{\bar{j}}\Theta_k^2. \end{aligned} \tag{3.56}$$

The definition of ρ_k, (3.56), (3.55), and (3.54) then yield

$$\begin{aligned} |\rho_k - 1| &= \left| \frac{q(s_k; x_k, y, \mu_k) - \mathcal{L}(x_k + s_k, y, \mu_k)}{\Delta q(s_k; x_k, y, \mu_k)} \right| \\ &\leq \frac{2\Omega_{\bar{j}}\Theta_k^2}{\kappa_1 \zeta \Theta_k} = \frac{2\Omega_{\bar{j}}\Theta_k}{\kappa_1 \zeta} \leq 1 - \eta_{vs}. \end{aligned}$$

This implies that a very successful iteration will occur and along with (3.53) we may conclude that the trust region radius will not be decreased any further. Consequently, we have shown that the trust region radius updating strategy in Algorithm 2 guarantees that for some $\delta_{\min} \in (0, \delta_{\text{thresh}})$ we have

$$\Theta_k \geq \delta_{\min} \;\text{ for all }\; k \geq k''. \tag{3.57}$$

Now, since Θ_k is bounded below, there must exist an infinite subsequence, call it $\mathcal{S}$, of successful iterates. If we define $\mathcal{S}'' := \{k \in \mathcal{S} : k \geq k''\}$, then we may conclude from the fact that $x_{k+1} = x_k$ when $k \notin \mathcal{S}$, (3.49), (3.10), (3.55), and (3.57) that

$$\begin{aligned} & \sum_{k=k''}^{\infty} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \\ = \; & \sum_{k \in \mathcal{S}''} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \\ \geq \; & \sum_{k \in \mathcal{S}''} \eta_s \Delta q(s_k; x_k, y, \mu_k) \geq \sum_{k \in \mathcal{S}''} \tfrac{1}{2}\eta_s \kappa_1 \zeta \delta_{\min} = \infty, \end{aligned} \tag{3.58}$$

contradicting (3.33). Therefore, we conclude that (3.51) holds.

Now we prove the second part of (3.50). By contradiction, suppose $J_k^T c_k \nrightarrow 0$. This supposition and (3.51) imply that the subsequence of successful iterates $\mathcal{S}$ is infinite and there exist a constant $\varepsilon > 0$ and sequences $\{\underline{k}_i\}_{i \geq 0}$ and $\{\overline{k}_i\}_{i \geq 0}$ defined in the following manner: $\underline{k}_0$ is the first iterate in $\mathcal{S}$ such that $\|J_{\underline{k}_0}^T c_{\underline{k}_0}\|_2 \geq 4\varepsilon > 0$; $\overline{k}_i$ for $i \geq 0$ is the first iterate strictly greater than $\underline{k}_i$ such that

$$\|J_{\overline{k}_i}^T c_{\overline{k}_i}\|_2 < 2\varepsilon; \tag{3.59}$$

$\underline{k}_i$ for $i \geq 1$ is the first iterate in $\mathcal{S}$ strictly greater than $\overline{k}_{i-1}$ such that

$$\|J_{\underline{k}_i}^T c_{\underline{k}_i}\|_2 \geq 4\varepsilon > 0. \tag{3.60}$$

We may now define

$$\mathcal{K} := \{k \in \mathcal{S} : \underline{k}_i \leq k < \overline{k}_i \;\text{ for some }\; i \geq 0\}.$$

Since $\mu_k \to 0$, we may use (3.49) to conclude that there exists k''' such that

$$\|\nabla_x \mathcal{L}(x_k, y, \mu_k)\|_2 \geq \varepsilon \text{ for all } k \in \mathcal{K} \text{ such that } k \geq k'''. \tag{3.61}$$

It follows from the definition of $\mathcal{K}$, (3.9a), (3.16), (3.49), (3.27b), and (3.61) that

$$\begin{aligned}
&\mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \\
&\geq \eta_s \Delta q(s_k; x_k, y_k, \mu_k) \\
&\geq \tfrac{1}{2}\eta_s \kappa_1 \|\nabla_x \mathcal{L}(x_k, y, \mu_k)\|_2 \min\left\{\frac{\|\nabla_x \mathcal{L}(x_k, y, \mu_k)\|_2}{1 + \Omega_{\bar{j}}}, \Theta_k\right\} \\
&\geq \tfrac{1}{2}\eta_s \kappa_1 \varepsilon \min\left\{\frac{\varepsilon}{1 + \Omega_{\bar{j}}}, \Theta_k\right\} \text{ for all } k \in \mathcal{K} \text{ such that } k \geq k'''.
\end{aligned} \tag{3.62}$$

It also follows from (3.33) and since $\mathcal{L}(x_{k+1}, y_k, \mu_k) \leq \mathcal{L}(x_k, y_k, \mu_k)$ for all $k \geq 0$ that

$$\begin{aligned}
\infty &> \sum_{k=0}^{\infty} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \\
&= \sum_{k \in \mathcal{S}} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k) \\
&\geq \sum_{k \in \mathcal{K}} \mathcal{L}(x_k, y_k, \mu_k) - \mathcal{L}(x_{k+1}, y_k, \mu_k).
\end{aligned} \tag{3.63}$$

Summing (3.62) for $k \in \mathcal{K}$ and using (3.63) yields

$$\lim_{k \in \mathcal{K}} \Theta_k = 0,$$

so we may use (3.62) and (3.49) to obtain

$$\mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \geq \tfrac{1}{2}\eta_s \kappa_1 \varepsilon \Theta_k > 0 \text{ for all sufficiently large } k \in \mathcal{K}. \tag{3.64}$$

By the triangle-inequality and (3.64), there exists some $\bar{i} \geq 1$ such that

$$\begin{aligned}
\|x_{\underline{k}_i} - x_{\overline{k}_i}\|_2 &\leq \sum_{j=\underline{k}_i}^{\overline{k}_i - 1} \|x_j - x_{j+1}\|_2 = \sum_{j=\underline{k}_i}^{\overline{k}_i - 1} \|s_j\|_2 \leq \sum_{j=\underline{k}_i}^{\overline{k}_i - 1} \Theta_j \\
&\leq \frac{2}{\eta_s \kappa_1 \varepsilon} \sum_{j=\underline{k}_i}^{\overline{k}_i - 1} \mathcal{L}(x_j, y, \mu_j) - \mathcal{L}(x_{j+1}, y, \mu_j) \text{ for } i \geq \bar{i}.
\end{aligned}$$

Summing over all $i \geq \bar{i}$ and using (3.33), we find

$$\begin{aligned}
\sum_{i=\bar{i}}^{\infty} \|x_{\underline{k}_i} - x_{\overline{k}_i}\|_2 &\leq \frac{2}{\eta_s \kappa_1 \varepsilon} \sum_{i=\bar{i}}^{\infty} \left(\sum_{j=\underline{k}_i}^{\overline{k}_i - 1} \mathcal{L}(x_j, y, \mu_j) - \mathcal{L}(x_{j+1}, y, \mu_j) \right) \\
&\leq \frac{2}{\eta_s \kappa_1 \varepsilon} \sum_{k \in \mathcal{K}} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) \\
&\leq \frac{2}{\eta_s \kappa_1 \varepsilon} \sum_{k=0}^{\infty} \mathcal{L}(x_k, y, \mu_k) - \mathcal{L}(x_{k+1}, y, \mu_k) < \infty,
\end{aligned}$$

which implies that

$$\lim_{i\to\infty} \|x_{\underline{k}_i} - x_{\overline{k}_i}\|_2 = 0.$$

It follows from (3.60) and Assumption 3.2 that for i sufficiently large $\|J_{\overline{k}_i}^T c_{\overline{k}_i}\|_2 > 2\varepsilon$, contradicting (3.59). We may conclude that the second part of (3.50) holds.

The fact that every limit point of $\{x_k\}_{k\geq 0}$ is an infeasible stationary point follows from (3.50). □

This completes the analysis for the case that the set $\mathcal{Y}$ is finite. The next section considers the complementary situation when the Lagrange multiplier vector is updated an infinite number of times.

3.2.2 Infinite number of multiplier updates

We now consider cases when $|\mathcal{Y}| = \infty$. In such cases, it follows by construction in Algorithm 2 that

$$\lim_{j\to\infty} t_j = \lim_{j\to\infty} T_j = 0. \tag{3.65}$$

As in the previous subsection, we consider two subcases depending on whether the penalty parameter remains bounded away from zero, or if it converges to zero. Our next lemma shows that when the penalty parameter does remain bounded away, then a subsequence of iterates corresponding to $\mathcal{Y}$ in (3.14) converges to a first-order optimal point. In general, this is the ideal case for a feasible problem.

Lemma 3.8 *Suppose Assumption 3.2 holds. Then, if $|\mathcal{Y}| = \infty$ and $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, then*

$$\lim_{k\in\mathcal{Y}} c_{k+1} = 0 \tag{3.66}$$

$$\text{and} \quad \lim_{k\in\mathcal{Y}} \left(g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}\right) = 0. \tag{3.67}$$

Thus, any limit point (x_*, y_*) of $\{(x_{k+1}, \widehat{y}_{k+1})\}_{k\in\mathcal{Y}}$ is first-order optimal for (2.1).

Proof The limit (3.66) follows from the definition of $\mathcal{Y}$, (3.65), and lines 26 and 30 of Algorithm 2. To prove (3.67), we first define

$$\mathcal{Y}' = \{k \in \mathcal{Y} : \|g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}\|_2 \leq \|\nabla_x \mathcal{L}(x_{k+1}, y_k, \mu_k)\|_2\}.$$

It follows from (3.65) and line 28 of Algorithm 2 that

$$\lim_{k\in\mathcal{Y}'} (g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}) = 0 \ \text{ and } \ \lim_{k\in\mathcal{Y}\setminus\mathcal{Y}'} \nabla_x \mathcal{L}(x_{k+1}, y_k, \mu_k) = 0. \tag{3.68}$$

Under Assumption 3.2, this latter equation may be combined with (3.66) and the fact that $\mu_k = \mu$ for some $\mu > 0$ to deduce that

$$\lim_{k\in\mathcal{Y}\setminus\mathcal{Y}'} (g_{k+1} - J_{k+1}^T y_k) = 0. \tag{3.69}$$

We may now combine (3.69) with (3.11) to state that

$$\lim_{k\in\mathcal{Y}\setminus\mathcal{Y}'} (g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}) = 0. \tag{3.70}$$

The desired result (3.67) now follows from (3.68) and (3.70). □

The following corollary considers the effect of using asymptotically exact least-length least-squares Lagrange multiplier estimates for $\widehat{y}_{k+1}$ in line 27 of Algorithm 2. This is an important special case as such multipliers can be computed using matrix-free iterative methods.

Corollary 3.9 *Suppose Assumption 3.2 holds, $|\mathcal{Y}| = \infty$, $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k, and*

$$\lim_{k \in \mathcal{Y}} \|\widehat{y}_{k+1} - y_{k+1}^{LS}\|_2 = 0, \tag{3.71}$$

where

$$y_{k+1}^{LS} := \operatorname*{argmin}_{y \in \mathbb{R}^m} \|y\|_2 \;\text{ subject to }\; y \in \operatorname*{argmin}_{y \in \mathbb{R}^m} \; \|g_{k+1} - J_{k+1}^T y\|_2^2 \tag{3.72}$$

denotes the least-length least-squares multiplier estimates at x_{k+1}. Then, for any limit point x_* of $\{x_{k+1}\}_{k \in \mathcal{Y}}$, there exists a subsequence $\mathcal{Y}' \subseteq \mathcal{Y}$ such that

$$\lim_{k \in \mathcal{Y}'} (x_{k+1}, y_{k+1}) = (x_*, y_*)$$

where (x_*, y_*) is a first-order optimal point for problem (2.1).

Proof Let x_* be any limit point of $\{x_{k+1}\}_{k \in \mathcal{Y}}$. Then, there exists an infinite subsequence $\mathcal{Y}' \subseteq \mathcal{Y}$ such that

$$\lim_{k \in \mathcal{Y}'} x_{k+1} = x_*. \tag{3.73}$$

We know from (3.65) that $t_j \to 0$, which implies from line 29 of Algorithm 2 that

$$\lim_{j \to \infty} Y_{j+1} = \lim_{j \to \infty} t_{j-1}^{-\varepsilon} = \infty. \tag{3.74}$$

It now follows from (3.71), (3.72), (3.73), and (3.74) that

$$\|\widehat{y}_{k+1}\|_2 \leq Y_{j+1} \;\text{ for sufficiently large }\; k \in \mathcal{Y}'.$$

Moreover, the update for y_{k+1} given by (3.12) and (3.13) uses $\alpha_y = 1$ and yields

$$y_{k+1} = \widehat{y}_{k+1} \;\text{ for sufficiently large }\; k \in \mathcal{Y}'. \tag{3.75}$$

We now may use (3.75), (3.73), and (3.71) to conclude that

$$\lim_{k \in \mathcal{Y}'} y_{k+1} = \lim_{k \in \mathcal{Y}'} \widehat{y}_{k+1} = y_*,$$

where y_* is the least-length least-squares multiplier at x_*. Finally, we may combine the previous equation with (3.73) and Lemma 3.8 to conclude that (x_*, y_*) is a first-order optimal point for problem (2.1). □

Finally, we consider the case when the penalty parameter converges to zero.

Lemma 3.10 *Suppose Assumption 3.2 holds, $|\mathcal{Y}| = \infty$, and $\mu_k \to 0$. Then,*

$$\lim_{k \to \infty} c_k = 0. \tag{3.76}$$

Proof It follows from (3.34) that

$$\lim_{k \to \infty} \|c_k\|_2 = c_{\text{L}} \geq 0. \tag{3.77}$$

However, it also follows from (3.65) and line 26 of Algorithm 2 that

$$\lim_{j \to \infty} \|c_{k_j}\|_2 \leq \lim_{j \to \infty} t_j = 0. \tag{3.78}$$

The limit (3.76) now follows from (3.77) and (3.78). □

3.2.3 A global convergence theorem

We combine the lemmas in the previous sections to obtain the following result.

Theorem 3.11 *Suppose Assumption 3.2 holds. Then, one of the following must hold:*

(i) each limit point x_* of $\{x_k\}$ is an infeasible stationary point;
(ii) $\mu_k = \mu$ for some $\mu > 0$ for all sufficiently large k and there exists a subsequence $\mathcal{K}$ such that each limit point (x_*, y_*) of $\{(x_k, \widehat{y}_k)\}_{k\in\mathcal{K}}$ is first-order optimal for (2.1);
(iii) $\mu_k \to 0$ and each limit point x_* of $\{x_k\}$ is feasible.

Proof Lemmas 3.6, 3.7, 3.8, and 3.10 involve the four possible outcomes of Algorithm 2. Hence, under Assumption 3.2, the result follows from these lemmas.

We end this section with a discussion of Theorem 3.11. As for all penalty methods for solving nonconvex optimization problems, Algorithm 2 may converge to an infeasible stationary point. This potential outcome is unavoidable as we do not assume, explicitly or implicitly, that problem (2.1) is feasible. By far, the most common outcome of our numerical results in Section 5 is case (ii) of the theorem, in which the penalty parameter ultimately remains fixed and convergence to a first-order primal-dual solution of (2.1) is observed. In fact, these numerical tests show that our adaptive algorithm is far more efficient and at least as robust as methods related to Algorithm 1. The outcome in case (iii) of the theorem is that the penalty parameter and the constraint violation both converge to zero. In fact, this possible outcome should not be a surprise. Global convergence guarantees for other algorithms avoid this possibility by assuming some sort of constraint qualification, such as LICQ and MCFQ, but we make no such assumption here and so obtain a weaker result. Nonetheless, we remain content with our theoretical guarantees since the numerical tests in Section 5 show that Algorithm 2 is far superior to a more basic strategy, and over a large set of feasible test problems the penalty parameter consistently remains bounded away from zero.

In the end, any shortcomings of Theorem 3.11 can be ignored if one simply employs our adaptive strategy as a mechanism for obtaining an improved initial penalty parameter and Lagrange multiplier estimate. That is, given arbitrary initial values for these quantities, Algorithm 2 can be employed until sufficient progress in minimizing constraint violation has been observed, at which point the algorithm can transition to a traditional augmented Lagrangian method. This easily implementable approach both benefits from our adaptive penalty parameter updating strategy and inherits the well-documented global and local convergence guarantees of traditional augmented Lagrangian methods.

4 An Adaptive Augmented Lagrangian Line Search Algorithm

In this section, we present an AL line search method with an adaptive penalty parameter update. This algorithm is very similar to the trust region method developed in §3, so for the sake of brevity we simply highlight the differences between the two approaches. Overall, we argue that our line search method has similar global convergence guarantees as our trust region method.

Our line search algorithm is stated as Algorithm 3 on page 23. The first difference between Algorithms 2 and 3 is the quadratic model employed for the augmented Lagrangian. In our line search method, we replace (3.4) with

$$\underset{s\in\mathbb{R}^n}{\text{minimize}}\ \widetilde{q}(s;x_k,y_k,\mu_k)\ \text{ subject to }\ \|s\|_2\leq\Theta_k, \tag{4.1}$$

where $\widetilde{q}$ is the "convexified" model

$$\begin{aligned}\widetilde{q}(s;x,y,\mu) &= \mathcal{L}(x,y,\mu)+\nabla_x\mathcal{L}(x,y,\mu)^Ts+\max\{\tfrac{1}{2}s^T(\mu\nabla^2_{xx}\ell(x,y)+J(x)^TJ(x))s,0\}\\ &\approx \mathcal{L}(x+s,y,\mu).\end{aligned}$$

The Cauchy point (see (3.5)) corresponding to this convexified model is

$$\widetilde{s}_k := \widetilde{s}(x_k,y_k,\mu_k,\Theta_k) := -\widetilde{\alpha}(x_k,y_k,\mu_k,\Theta_k)\nabla_x\mathcal{L}(x_k,y_k,\mu_k), \tag{4.2}$$

where

$$\begin{aligned}\widetilde{\alpha}(x_k,y_k,\mu_k,\Theta_k) := \quad & \underset{\alpha\geq0}{\text{argmin}}\ \ \widetilde{q}(-\alpha\nabla_x\mathcal{L}(x_k,y_k,\mu_k))\\ & \text{subject to}\ \ \|\alpha\nabla_x\mathcal{L}(x_k,y_k,\mu_k)\|_2\leq\Theta_k.\end{aligned}$$

We then replace condition (3.9a) with

$$\Delta\widetilde{q}(s_k;x_k,y_k,\mu_k)\geq\kappa_1\Delta\widetilde{q}(\widetilde{s}_k;x_k,y_k,\mu_k)>0, \tag{4.3}$$

where the predicted reduction in $\mathcal{L}(\cdot,y,\mu)$ from x yielded by s is defined as

$$\Delta\widetilde{q}(s;x,y,\mu) := \widetilde{q}(0;x,y,\mu)-\widetilde{q}(s;x,y,\mu).$$

(In fact, in our implementation described in §5, the search direction s_k in Algorithm 3 is computed from subproblem (3.4), not from (4.1). This is allowed as long as one ensures that s_k satisfies (4.3), which is easily done; see §5.)

The purpose of using the convexified model in condition (4.3) is that it ensures that s_k is a descent direction for $\mathcal{L}(\cdot,y_k,\mu_k)$ at x_k; see Lemma 4.1. Note that in a sufficiently small neighborhood of a strict local minimizer of the augmented Lagrangian at which the second-order sufficient conditions are satisfied, the Hessian of q is positive definite, in which case conditions (3.9a) and (4.3) are equivalent.

The second difference between Algorithms 2 and 3 is the manner in which the primal iterate is updated. In line 19 of Algorithm 3 we perform a backtracking line search along the nonzero descent direction s_k. To be precise, for a given $\gamma_\alpha\in(0,1)$, we compute the smallest integer $l\geq0$ such that

$$\mathcal{L}(x_k+\gamma_\alpha^ls_k,y_k,\mu_k)\leq\mathcal{L}(x_k,y_k,\mu_k)-\eta_s\gamma_\alpha^l\Delta\widetilde{q}(s_k;x_k,y_k,\mu_k), \tag{4.4}$$

then set $\alpha_k\leftarrow\gamma_\alpha^l$ and update $x_{k+1}\leftarrow x_k+\alpha_ks_k$.

The final difference between our trust region and line search algorithms is in the manner in which the trust region radii are set; compare line 11 of Algorithm 2 and line 11 of Algorithm 3. With these trust region radii in the line search algorithm, the Cauchy decreases in Lemma 3.2 are now given—under Assumption 3.2—by

$$\Delta\widetilde{q}(\widetilde{s}_k;x_k,y_k,\mu_k)\geq\tfrac{1}{2}\|\nabla_x\mathcal{L}(x_k,y_k,\mu_k)\|_2^2\min\left\{\frac{1}{1+\Omega_j},\delta\right\} \tag{4.5}$$

Algorithm 3 Adaptive Augmented Lagrangian Line Search Algorithm

Choose constants $\{\gamma_\mu, \gamma_t, \gamma_T, \gamma_\alpha, \kappa_F, \kappa_1, \kappa_2, \kappa_3, \kappa_t, \eta_s\} \subset (0,1)$ and $\{\delta, \epsilon, Y\} \subset (0,\infty)$.
Choose an initial primal-dual pair (x_0, y_0) and initialize $\{\mu_0, t_0, T_0, Y_1\} \subset (0,\infty)$ such that $Y_1 \geq Y$ and $\|y_0\|_2 \leq Y_1$.
Set $k \leftarrow 0$ and $j \leftarrow 1$.
loop
  if $F_{\text{OPT}}(x_k, y_k) = 0$, **then**
    return the first-order solution (x_k, y_k).
  if $\|c_k\|_2 > 0$ and $J_k^T c_k = 0$, **then**
    return the infeasible stationary point x_k.
  while $\nabla_x \mathcal{L}(x_k, y_k, \mu_k) = 0$, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$.
  Set $\theta_k \leftarrow \delta \|J_k^T c_k\|_2$ and $\Theta_k \leftarrow \delta \|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2$.
  Compute the Cauchy points $\widetilde{s}_k$ and $\overline{r}_k$ for (4.1) and (3.6), respectively.
  Compute an approximate solution s_k to (4.1) satisfying (4.3).
  Compute an approximate solution r_k to (3.6) satisfying (3.9b).
  while $\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2 = 0$ or (3.9c) is not satisfied, **do**
    Set $\mu_k \leftarrow \gamma_\mu \mu_k$ and $\Theta_k \leftarrow \delta \|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2$.
    Compute the Cauchy point $\widetilde{s}_k$ for (4.1).
    Compute an approximate solution s_k to (4.1) satisfying (4.3).
  Set $\alpha_k \leftarrow \gamma_\alpha^l$ where $l \geq 0$ is the smallest integer satisfying (4.4).
  Set $x_{k+1} \leftarrow x_k + \alpha_k s_k$.
  if $\|c_{k+1}\|_2 \leq t_j$, **then**
    Compute any $\widehat{y}_{k+1}$ that satisfies (3.11).
    if $\min\left\{\|g_{k+1} - J_{k+1}^T \widehat{y}_{k+1}\|_2,\ \|\nabla_x \mathcal{L}(x_{k+1}, y_k, \mu_k)\|_2\right\} \leq T_j$, **then**
      Set $k_j \leftarrow k+1$ and $Y_{j+1} \leftarrow \max\{Y, t_{j-1}^{-\epsilon}\}$.
      Set $t_{j+1} \leftarrow \min\{\gamma_t t_j, t_j^{1+\epsilon}\}$ and $T_{j+1} \leftarrow \gamma_T T_j$.
      Set y_{k+1} from (3.12) where α_y yields (3.13).
      Set $j \leftarrow j+1$.
    else
      Set $y_{k+1} \leftarrow y_k$.
  else
    Set $y_{k+1} \leftarrow y_k$.
  Set $\mu_{k+1} \leftarrow \mu_k$.
  Set $k \leftarrow k+1$.

and

$$\Delta q(\overline{r}_k; x_k) \geq \tfrac{1}{2} \|J_k^T c_k\|_2^2 \min\left\{\frac{1}{1+\Omega_j}, \delta\right\}. \tag{4.6}$$

Importantly, this implies that the Cauchy decreases for both models converge to zero if and only if the gradients of their respective models converge to zero.

Since a complete global convergence analysis of Algorithm 3 would be very similar to the analysis of Algorithm 2 provided in §3, we do not provide a detailed analysis of Algorithm 3. Rather, we prove two critical lemmas and claim that they can be used in the context of Algorithm 3 to provide the same global convergence guarantees as we have provided for Algorithm 2.

We begin by proving our earlier claim that s_k is a descent direction for the augmented Lagrangian.

Lemma 4.1 *Suppose Assumption 3.2 holds. Then, s_k satisfying* (4.3) *is a descent direction for $\mathcal{L}(\cdot, y_k, \mu_k)$ at x_k. In particular,*

$$\nabla_x \mathcal{L}(x_k, y_k, \mu_k)^T s_k \leq -\Delta \widetilde{q}(s_k; x_k, y_k, \mu_k) \leq -\kappa_1 \Delta \widetilde{q}(\widetilde{s}_k; x_k, y_k, \mu_k) < 0. \tag{4.7}$$

Proof From the definition of $\widetilde{q}$, we find

$$\begin{aligned}
&\Delta\widetilde{q}(s_k; x_k, y_k, \mu_k) \\
&= \widetilde{q}(0; x_k, y_k, \mu_k) - \widetilde{q}(s_k; x_k, y_k, \mu_k) \\
&= -\nabla_x \mathcal{L}(x_k, y_k, \mu_k)^T s_k - \max\{\tfrac{1}{2} s_k^T (\mu_k \nabla^2_{xx} \ell(x_k, y_k) + J_k^T J_k) s_k, 0\} \\
&\leq -\nabla_x \mathcal{L}(x_k, y_k, \mu_k)^T s_k.
\end{aligned}$$

It follows that s_k satisfying (4.3) yields

$$\nabla_x \mathcal{L}(x_k, y_k, \mu_k)^T s_k \leq -\Delta\widetilde{q}(s_k; x_k, y_k, \mu_k) \leq -\kappa_1 \Delta\widetilde{q}(\widetilde{s}_k; x_k, y_k, \mu_k) < 0,$$

as desired. □

Our second lemma states that the step-size α_k remains bounded away from zero. Results of this kind are critical in proving global convergence guarantees for line search methods.

Lemma 4.2 *Suppose that Assumption 3.2 holds. Then, the step-size α_k computed in line 19 of Algorithm 3 satisfies*

$$\alpha_k \geq C$$

for some constant $C > 0$ independent of k.

Proof By Taylor's Theorem and Lemma 4.1, it follows under Assumption 3.2 that there exists $\tau > 0$ such that for all sufficiently small $\alpha > 0$ we have

$$\mathcal{L}(x_k + \alpha s_k, y_k, \mu_k) - \mathcal{L}(x_k, y_k, \mu_k) \leq -\alpha \Delta\widetilde{q}(s_k; x_k, y_k, \mu_k) + \tau \alpha^2 \|s_k\|^2. \tag{4.8}$$

On the other hand, during the line search a step-size α is rejected if

$$\mathcal{L}(x_k + \alpha s_k, y_k, \mu_k) - \mathcal{L}(x_k, y_k, \mu_k) > -\eta_s \alpha \Delta\widetilde{q}(s_k; x_k, y_k, \mu_k). \tag{4.9}$$

Combining (4.8) and (4.9), we have that a rejected step-size α satisfies

$$\alpha > \frac{(1-\eta_s)\Delta\widetilde{q}(s_k; x_k, y_k, \mu_k)}{\tau \|s_k\|_2^2} \geq \frac{(1-\eta_s)\Delta\widetilde{q}(s_k; x_k, y_k, \mu_k)}{\tau \Theta_k^2}.$$

From this bound, the fact that if the line search rejects a step-size α it multiplies it by $\gamma_\alpha \in (0, 1)$, (4.3), (4.5), and (3.27b), it follows that

$$\begin{aligned}
\alpha_k &\geq \frac{\gamma_\alpha (1-\eta_s)\Delta\widetilde{q}(s_k; x_k, y_k, \mu_k)}{\tau \Theta_k^2} \\
&\geq \frac{\kappa_1 \gamma_\alpha (1-\eta_s) \|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2^2 \min\left\{\frac{1}{1+\Omega_j}, \delta\right\}}{2\tau\delta^2 \|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_2^2} \\
&= \frac{\kappa_1 \gamma_\alpha (1-\eta_s)}{2\tau\delta^2} \min\left\{\frac{1}{1+\Omega_j}, \delta\right\} =: C > 0,
\end{aligned}$$

as desired. □

We now briefly summarize the analyses in §3.1 and §3.2 and mention the few minor differences required to prove similar results for Algorithm 3. First, recall that we have already discussed the necessary changes to Lemma 3.2; see (4.5) and (4.6). As for Lemma 3.1, it remains true without any changes, but Lemma 3.3 remains true only after minor alterations to the trust region radii. These alternations have no significant effect since the proof of Lemma 3.3 only uses the fact that $\Theta(\mu) \to \theta$ as $\mu \to 0$, and this remains true. Theorem 3.4 then follows in an identical fashion, so we have argued that Algorithm 3 is well-posed.

Now consider the global convergence analysis in §3.2. Lemma 3.5 remains true since the updating strategy for the Lagrange multiplier vector has not changed. Similarly, Lemma 3.6 remains true, but only after modifications to the proof in two places. First, condition (3.43) is still true, but it now follows from standard analysis of line search methods as a result of Lemmas 4.1 and 4.2, the Cauchy decrease (4.5), and the fact that $\mathcal{L}$ is bounded below under Assumption 3.2. Second, the argument that lead to (3.48) is now trivial using (4.6).

Next, consider Lemma 3.7. The proof of (3.49) and the first part of (3.50) is the same, and the proof of the second part of (3.50) can be simplified. In particular, if $\|J_k^T c_k\|_2 \nrightarrow 0$, then there exists a subsequence along which $\{\|J_k^T c_k\|_2\}$ and $\{\|\nabla \mathcal{L}(x_k, y_k, \mu_k)\|_2\}$ are uniformly bounded away from zero. This follows since $\mu_k \to 0$ by assumption. Combining this observation with (4.4), the manner in which α_k is computed, Lemmas 4.1 and 4.2, and (4.5), leads to a contradiction with (3.33). Thus, the second part of (3.50) remains true.

Finally, Lemma 3.8, Corollary 3.9, Lemma 3.10, and Theorem 3.11 remain true without any significant changes required in the proofs of these results.

5 Numerical Experiments

In this section we describe the details of implementations of our trust region and line search methods. We also discuss the results of numerical experiments.

5.1 Implementation details

Algorithms 2 and 3 were implemented in Matlab. Hereinafter, we refer to the former as `AAL-TR` and the latter as `AAL-LS`. There were many similar components of the two implementations, so for the most part our discussion of details applies to both algorithms. For comparison purposes, we also implemented two variants of Algorithm 1: one with a trust region method employed in line 9 and one with a line search method employed. We refer to these methods as `BAL-TR` and `BAL-LS`, respectively. The components of these implementations were also very similar to those for `AAL-TR` and `AAL-LS`. For example, the implemented update for the trust region radii in `BAL-TR` and the line search in `BAL-LS` were exactly the same as those implemented for `AAL-TR` and `AAL-LS`, respectively.

A critical component of all the implementations was the approximate solution of the trust region subproblem (3.4) and, in the cases of `AAL-TR` and `AAL-LS`, subproblem (3.6). For these subproblems we implemented a conjugate gradient (CG) algorithm with Steihaug-Toint stopping criteria [10,12]. (In this manner, the Cauchy points for each subproblem were obtained in the first iteration of CG.)

For each subproblem for which it was employed, the CG iteration was run until either the trust region boundary was reached (perhaps due to a negative curvature direction being obtained) or the gradient of the subproblem objective was below a predefined constant $\kappa_{\text{cg}} > 0$. Note that truncating CG in this manner did not automatically guarantee that condition (4.3) held. Therefore, in `BAL-LS` and `AAL-LS`, we stored and ultimately returned the last CG iterate that satisfied (4.3), knowing at least that this condition was satisfied by the Cauchy point. In the cases of `BAL-TR` and `BAL-LS`, the solution obtained from CG was used in a minor iteration—see step 9 of Algorithm 1—whereas in the cases of `AAL-TR` and `AAL-LS`, the solutions obtained from CG were used as explicitly stated in Algorithms 2 and 3.

A second component of `AAL-TR` and `AAL-LS` that requires specification was the strategy employed for computing the trial multipliers $\{\widehat{y}_{k+1}\}$. For this computation, the first-order multipliers defined by π in (2.5) were used as long as they satisfied (3.11); otherwise, $\widehat{y}_{k+1}$ was set to y_k so that (3.11) held true.

Each algorithm terminated with a declaration of optimality if

$$\|\nabla_x \mathcal{L}(x_k, y_k, \mu_k)\|_\infty \leq \kappa_{\text{opt}} \quad \text{and} \quad \|c_k\|_\infty \leq \kappa_{\text{feas}}, \tag{5.1}$$

and terminated with a declaration that an infeasible stationary point was found if

$$\|J_k^T c_k\|_\infty \leq \kappa_{\text{opt}}, \quad \|c_k\|_\infty > \kappa_{\text{feas}}, \quad \text{and} \quad \mu_k \leq \mu_{\min}. \tag{5.2}$$

Note that in the latter case our implementations differ slightly from Algorithms 1, 2, and 3 as we did not declare that an infeasible stationary point was found until the penalty parameter was below a prescribed tolerance. The motivation for this was to avoid premature termination at (perhaps only slightly) infeasible points at which the gradient of the infeasibility measure $\|J_k^T c_k\|_\infty$ was relatively small compared to $\|c_k\|_\infty$. Also, each algorithm terminated with a declaration of failure if neither (5.1) nor (5.2) was satisfied within an iteration limit $k_{\max}$. The problem functions were pre-scaled so that the ℓ_∞-norms of the gradients of each at the initial point would be less than or equal to a prescribed constant $G > 0$. This helped to improve performance when solving poorly-scaled problems.

Table 5.1 below summarizes the input parameter values that were chosen. Note that our choice for Y means that we did not impose explicit bounds on the norms of the multipliers, meaning that we effectively always chose $\alpha_y \leftarrow 1$ in (3.12).

Table 5.1 Input parameter values used in `AAL-TR`, `AAL-LS`, `BAL-TR`, and `BAL-LS`.

Par.	Val.	Par.	Val.	Par.	Val.	Par.	Val.
γ_μ	5e-01	κ_3	1e-04	Y	∞	κ_{cg}	1e-10
γ_t	5e-01	κ_t	9e-01	Γ_δ	6e-01	κ_{opt}	1e-06
γ_T	5e-01	η_s	1e-02	μ_0	1e+00	κ_{feas}	1e-06
γ_δ	5e-01	η_{vs}	9e-01	t_0	1e+00	$\mu_{\min}$	1e-10
κ_F	9e-01	δ	1e+04	T_0	1e+00	$k_{\max}$	1e+03
κ_1	1e+00	δ_R	1e-04	δ_0	1e+00	G	1e+02
κ_2	1e+00	ϵ	5e-01	Y_1	∞		

In the next two sections we discuss the results of some numerical experiments. In §5.2 we discuss in detail a single test problem from the CUTEr [22] test set. This problem was chosen to illustrate the computational benefits of our adaptive updating scheme for the penalty parameter μ. In §5.3 we provide numerical results for a larger subset of the CUTEr collection.

5.2 An illustrative example

To illustrate the benefits of our adaptive updating strategy for the penalty parameter, we study the CUTEr problem `CATENA`. We have chosen a relatively small instance of the problem that consists of 15 variables and 4 equality constraints. We compare our results with that obtained by `Lancelot`, a very mature Fortran-90 implementation whose overall approach is well-represented by Algorithm 1. In fact, the algorithm in `Lancelot` benefits from a variety of advanced features from which the implementation of our methods could also benefit. However, since ours are only preliminary implementations of our methods, we set input parameters as described in Table 5.2 to have as fair a comparison as possible. The first two parameters disallowed a nonmonotone strategy and the possibility of using so-called "magical steps". The third and fourth parameters ensured that a ℓ_2-norm trust region constraint was used and allowed the possibility of inexact subproblem solves, roughly along the lines as we allowed and have described in §5.1. The last input parameter ensured that the initial value of the penalty parameter was the same as for our implementation (see Table 5.1).

Table 5.2 Input parameter values used in `Lancelot`.

Parameter	Value
HISTORY-LENGTH-FOR-NON-MONOTONE-DESCENT	0.0
MAGICAL-STEPS-ALLOWED	NO
USE-TWO-NORM-TRUST-REGION	YES
SUBPROBLEM-SOLVED-ACCURATELY	NO
INITIAL-PENALTY-PARAMETER	1.0

With the above alterations to its default parameters, we solved `CATENA` using `Lancelot`. A stripped version of the output is given in Figure 5.1. The columns represent the iteration number (`Iter`), cumulative number of gradient evaluations (`#g.ev`), objective value (`f`), projected gradient norm (`proj.g`), penalty parameter value (`penalty`), constraint violation target (`target`), constraint violation value (`infeas`), and a flag for which a value of 1 indicates that a multiplier update occurred. (An empty entry indicates that a value did not change since the previous iteration.) We make a couple observations. First, `Lancelot` only attempted to update the penalty parameter and multiplier vector on iterations 11, 16, 21, 26, 31, 35, 38, 41, and 43 when an approximate minimizer of the augmented Lagrangian was identified. Second, the constraint violation target was only satisfied in iterations 35, 41, and 43. Thus, it is reasonable to view the iterations that only lead to a decrease in the penalty parameter as wasted iterations. For this example, this includes iterations 1–31 and 36–38, which accounted for ≈79% of the iterations.

Next we solved `CATENA` using `AAL-TR` and `AAL-LS`. The results are provided in Figures 5.2 and 5.3. The columns represent the iteration number (`Iter.`), objective value (`Objective`), a measure of infeasibility (`Infeas.`), $\|g_k - J_k^T y_k\|_\infty$ (`Lag.Err.`), the target constraint violation for the current value of the penalty parameter and Lagrange mutiplier vector (`Fea.Tar.`), the value of the penalty parameter (`Pen.Par.`), $\Delta q_v(r_k; x_k)$ (`Dqv(r)`), $\Delta q_v(s_k; x_k)$ (`Dqv(s)`), and a flag for which a value of 1 indicates that a multiplier update occurred (`y`).

Iter	#g.ev	f	proj.g	penalty	target	infeas	y
0	1	-5.89E+03	1.2E+03	1.0E+00			
1	1	-5.89E+03	1.2E+03				
2	2	-1.38E+04	1.2E+03				
3	3	-2.88E+04	1.2E+03				
4	4	-5.07E+04	1.6E+03				
5	5	-6.79E+04	1.6E+03				
6	6	-7.92E+04	1.8E+03				
7	7	-7.94E+04	2.8E+03				
8	8	-8.43E+04	5.8E+02				
9	9	-8.48E+04	9.1E+01				
10	10	-8.49E+04	4.0E+00				
11	11	-8.49E+04	8.6E-03		1.0E-01	1.7E+02	
12	12	-1.21E+04	3.0E+03	1.0E-01			
13	13	-3.79E+04	6.5E+02				
14	14	-3.99E+04	7.1E+01				
15	15	-3.99E+04	1.3E+00				
16	16	-3.99E+04	4.5E-04		1.0E-01	3.6E+01	
17	17	-7.08E+03	3.0E+03	1.0E-02			
18	18	-1.88E+04	6.4E+02				
19	19	-1.96E+04	6.9E+01				
20	20	-1.96E+04	1.2E+00				
21	21	-1.96E+04	3.7E-04		7.9E-02	7.4E+00	
22	22	-6.19E+03	2.9E+03	1.0E-03			
23	23	-1.10E+04	6.0E+02				
24	24	-1.13E+04	5.7E+01				
25	25	-1.13E+04	7.1E-01				
26	26	-1.13E+04	1.1E-04		6.3E-02	1.4E+00	
27	27	-7.57E+03	2.6E+03	1.0E-04			
28	28	-8.79E+03	3.9E+02				
29	29	-8.82E+03	1.4E+01				
30	30	-8.82E+03	1.7E-02				
31	31	-8.82E+03	7.0E-08		5.0E-02	2.1E-01	
32	32	-8.36E+03	1.4E+03	1.0E-05			
33	33	-8.40E+03	2.8E+01				
34	34	-8.40E+03	1.2E-02				
35	35	-8.40E+03	1.0E-06		4.0E-02	2.3E-02	1
36	36	-8.30E+03	2.7E+01				
37	37	-8.30E+03	1.2E-02				
38	38	-8.30E+03	2.8E-09		1.3E-06	2.9E-04	
39	39	-8.34E+03	5.0E-02	1.0E-06			
40	40	-8.34E+03	3.7E-06				
41	41	-8.34E+03	2.1E-10		3.2E-02	3.0E-05	1
42	42	-8.34E+03	6.4E-04				
43	43	-8.34E+03	1.4E-09				

Fig. 5.1 Output from `Lancelot` for problem `CATENA`.

One can see that both of our methods solved the problem efficiently since they quickly realized that a decrease in the penalty parameter would be beneficial. Moreover, in both cases the feasibility measure `Fea.Err.` and optimality measure `Lag.Err.` converged quickly. This was due, in part, to the fact that the multiplier vector was updated during most iterations near the end of the run. In particular, `AAL-TR` updated the multiplier vector during 5 of the last 6 iterations and `AAL-LS` updated the multiplier vector during 4 of the last 5 iterations.

The most instructive column for witnessing the benefits of our penalty parameter updating strategy is `Dqv(s)` since this column shows the predicted decrease in the constraint violation yielded by the trial step s_k. When this quantity was positive the trial step predicted progress toward constraint satisfaction, and when it was negative the trial step predicted an increase in constraint violation. This quantity was compared with the quantity in column `Dqv(r)` in the steering condition (3.9c). It is clear in the output that the penalty parameter was decreased

```
=======+============================+==========+==========+======================+===
 Iter. | Objective  Infeas. Lag.Err. | Fea.Tar. | Pen.Par. |   Dqv(r)     Dqv(s)  | y
=======+============================+==========+==========+======================+===
     0 | -5.89e+03 2.80e-01 1.00e+00 | 2.52e-01 | 1.00e+00 | +1.52e-01  -1.82e-01 |
       |                             |          | 5.00e-01 |            -1.81e-01 |
       |                             |          | 2.50e-01 |            -1.77e-01 |
       |                             |          | 1.25e-01 |            -1.71e-01 |
       |                             |          | 6.25e-02 |            -1.59e-01 |
       |                             |          | 3.12e-02 |            -1.35e-01 |
       |                             |          | 1.56e-02 |            -9.35e-02 |
       |                             |          | 7.81e-03 |            -2.82e-02 |
       |                             |          | 3.91e-03 |            +3.98e-02 |
     1 | -7.94e+03 4.71e-01 3.91e-03 | 2.52e-01 | 3.91e-03 | +1.91e-01  +7.72e-02 |
     2 | -9.42e+03 6.31e-01 3.91e-03 | 2.52e-01 | 3.91e-03 | +2.44e-01  -1.11e-02 |
       |                             |          | 1.95e-03 |            +2.43e-01 |
     3 | -9.42e+03 6.31e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +2.37e-01  +2.06e-01 |
     4 | -9.53e+03 3.01e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +8.30e-02  +1.48e-02 |
     5 | -9.53e+03 3.01e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +5.09e-02  +1.15e-02 |
     6 | -9.68e+03 3.22e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +6.95e-02  +5.00e-02 |
     7 | -9.68e+03 3.22e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +4.00e-02  +3.22e-02 |
     8 | -9.62e+03 2.83e-01 1.95e-03 | 2.52e-01 | 1.95e-03 | +2.70e-02  -1.44e-02 |
       |                             |          | 9.77e-04 |            +2.98e-02 | 1
     9 | -9.40e+03 2.25e-01 9.77e-04 | 1.26e-01 | 9.77e-04 | +3.62e-02  +3.78e-02 |
    10 | -9.05e+03 1.64e-01 9.77e-04 | 1.26e-01 | 9.77e-04 | +1.80e-02  -2.77e-03 |
       |                             |          | 4.88e-04 |            +2.02e-02 | 1
    11 | -8.74e+03 9.13e-02 4.88e-04 | 4.47e-02 | 4.88e-04 | +1.06e-02  +8.95e-03 |
    12 | -8.74e+03 9.13e-02 4.88e-04 | 4.47e-02 | 4.88e-04 | +7.09e-03  +4.46e-03 |
    13 | -8.71e+03 9.13e-02 4.88e-04 | 4.47e-02 | 4.88e-04 | +6.05e-03  -1.11e-03 |
       |                             |          | 2.44e-04 |            +6.17e-03 | 1
    14 | -8.55e+03 4.42e-02 3.94e-05 | 9.46e-03 | 2.44e-04 | +2.03e-03  +2.02e-03 | 1
    15 | -8.36e+03 1.75e-03 1.09e-05 | 9.20e-04 | 2.44e-04 | +2.71e-06  +1.76e-06 |
    16 | -8.35e+03 9.90e-04 1.07e-05 | 9.20e-04 | 2.44e-04 | +9.36e-07  -9.78e-09 |
       |                             |          | 1.22e-04 |            +6.94e-07 | 1
    17 | -8.35e+03 5.01e-04 1.90e-09 | 2.79e-05 | 1.22e-04 | +2.42e-07  +2.42e-07 | 1
    18 | -8.35e+03 4.74e-06 1.01e-09 | 1.47e-07 | 1.22e-04 | +2.80e-11  +2.80e-11 | 1
    19 | -8.35e+03 1.42e-07 8.84e-14 | 1.00e-07 | 1.22e-04 | ---------  --------- | -
=======+============================+==========+==========+======================+===
```

Fig. 5.2 Output from `AAL-TR` for problem `CATENA`.

when the steering step predicted an increase in constraint violation, i.e., when `Dqv(s)` was negative, though exceptions were made when the constraint violation was well within `Fea.Tar.`, the constraint violation target.

This particular problem shows that our adaptive strategy has the potential to be very effective on problems that require the penalty parameter to be reduced from its initial value. In the next section we test the effectiveness of our new adaptive strategy on a large subset of the CUTEr test problems.

5.3 The CUTEr test problems

In this section we observe the effects of our penalty parameter updating strategy on a subset of the CUTEr [22] test problems. We obtained our subset by first delineating all constrained problems with equality constraints only. Next, we eliminated *aug2dc* and *dtoc3* because they are quadratic problems that were too large for our Matlab implementation to solve. Next, we eliminated *argtrig, artif, bdvalue, bdvalues, booth, bratu2d, bratu2dt, brownale, broydn3d, cbratu2d, cbratu3d, chandheu, chnrsbne, cluster, coolhans, cubene, drcavty1, drcavty2, drcavty3, eigenau, eigenb, eigenc, flosp2th, flosp2tl, flosp2tm, gottfr, hatfldf, hatfldg, heart8, himmelba, himmelbc, himmelbe, hypcir, integreq, msqrta, msqrtb, powellbs, powellsq, recipe, rsnbrne, sinvalne,*

Iter.	Objective	Infeas.	Lag.Err.	Fea.Tar.	Pen.Par.	Dqv(r)	Dqv(s)	y
0	-5.89e+03	2.80e-01	1.00e+00	2.52e-01	1.00e+00	+6.48e-02	-2.75e+03	
					5.00e-01		-6.82e+02	
					2.50e-01		-1.68e+02	
					1.25e-01		-4.08e+01	
					6.25e-02		-9.56e+00	
					3.12e-02		-2.06e+00	
					1.56e-02		-3.41e-01	
					7.81e-03		+9.93e-03	
1	-7.57e+03	3.55e-01	7.81e-03	2.52e-01	7.81e-03	+1.90e-01	-5.41e-02	
					3.91e-03		+7.39e-02	
2	-9.36e+03	6.71e-01	3.91e-03	2.52e-01	3.91e-03	+2.57e-01	+1.36e-01	
3	-1.00e+04	5.94e-01	3.91e-03	2.52e-01	3.91e-03	+2.72e-01	-1.07e-02	
					1.95e-03		+2.58e-01	
4	-1.00e+04	5.16e-01	1.95e-03	2.52e-01	1.95e-03	+2.34e-01	+1.99e-01	
5	-9.89e+03	3.64e-01	1.95e-03	2.52e-01	1.95e-03	+7.67e-02	+4.43e-02	
6	-9.88e+03	4.12e-01	1.95e-03	2.52e-01	1.95e-03	+7.07e-02	+5.54e-02	
7	-9.60e+03	2.93e-01	1.95e-03	2.52e-01	1.95e-03	+4.41e-02	-3.15e-03	
					9.77e-04		+2.92e-02	1
8	-9.37e+03	2.20e-01	9.77e-04	1.26e-01	9.77e-04	+4.07e-02	+3.55e-02	
9	-9.09e+03	1.76e-01	9.77e-04	1.26e-01	9.77e-04	+3.08e-02	+7.54e-03	
10	-9.03e+03	1.62e-01	9.77e-04	1.26e-01	9.77e-04	+2.58e-02	-9.48e-03	
					4.88e-04		+1.93e-02	
11	-8.98e+03	1.43e-01	4.88e-04	1.26e-01	4.88e-04	+2.03e-02	+1.43e-02	1
12	-8.78e+03	9.98e-02	4.88e-04	4.47e-02	4.88e-04	+1.01e-02	+5.37e-03	
13	-8.77e+03	9.57e-02	4.88e-04	4.47e-02	4.88e-04	+9.55e-03	+2.94e-03	
14	-8.76e+03	8.88e-02	4.88e-04	4.47e-02	4.88e-04	+5.38e-03	+2.20e-03	
15	-8.72e+03	8.31e-02	4.88e-04	4.47e-02	4.88e-04	+3.52e-03	-1.70e-04	
					2.44e-04		+2.61e-03	
16	-8.64e+03	5.20e-02	2.44e-04	4.47e-02	2.44e-04	+3.84e-03	+2.56e-03	
17	-8.55e+03	4.48e-02	2.44e-04	4.47e-02	2.44e-04	+2.03e-03	+4.64e-05	1
18	-8.55e+03	4.32e-02	3.41e-05	9.46e-03	2.44e-04	+1.99e-03	+1.99e-03	1
19	-8.36e+03	2.67e-03	6.97e-06	9.20e-04	2.44e-04	+3.89e-06	+3.31e-06	1
20	-8.35e+03	7.95e-04	6.93e-08	2.79e-05	2.44e-04	+5.70e-07	+5.69e-07	1
21	-8.35e+03	2.00e-05	1.43e-09	1.47e-07	2.44e-04	+4.11e-10	+4.10e-10	
22	-8.35e+03	5.38e-07	3.86e-09	1.47e-07	2.44e-04	---------	---------	-

Fig. 5.3 Output from `AAL-LS` for problem `CATENA`.

spmsqrt, trigger, yatp1sq, yatp2sq, yfitne, and *zangwil3* because `Lancelot` recognized them as not having an objective function, in which case a penalty parameter was not required. Next, we removed *heart6, hydcar20, hydcar6, methanb8,* and *methanl8* because `Lancelot` solved them without introducing a penalty parameter (though we were not sure how this was possible). Finally, we removed *arglcle, junkturn,* and *woodsne* because all of the solvers (including `Lancelot`) converged to a point that was recognized as an infeasible stationary point. This left us with a total of 91 problems composing our subset: *bt1, bt10, bt11, bt12, bt2, bt3, bt4, bt5, bt6, bt7, bt8, bt9, byrdsphr, catena, chain, dixchlng, dtoc1l, dtoc2, dtoc4, dtoc5, dtoc6, eigena2, eigenaco, eigenb2, eigenbco, eigenc2, eigencco, elec, genhs28, gridnetb, gridnete, gridneth, hager1, hager2, hager3, hs100lnp, hs111lnp, hs26, hs27, hs28, hs39, hs40, hs42, hs46, hs47, hs48, hs49, hs50, hs51, hs52, hs56, hs6, hs61, hs7, hs77, hs78, hs79, hs8, hs9, lch, lukvle1, lukvle10, lukvle11, lukvle12, lukvle13, lukvle14, lukvle15, lukvle16, lukvle17, lukvle18, lukvle2, lukvle3, lukvle4, lukvle6, lukvle7, lukvle8, lukvle9, maratos, mss1, mwright, optctrl3, optctrl6, orthrdm2, orthrds2, orthrega, orthregb, orthregc, orthregd, orthrgdm, orthrgds,* and *s316-322*.

Let us first compare `AAL-TR` and `BAL-TR` with `Lancelot` since all are trust region methods. As previously mentioned, both `Lancelot` and `BAL-TR` are based on Algo-

rithm 1, though the details of their implementations are quite different. Therefore, we use this first comparison to illustrate that our implementation of `BAL-TR` is roughly on par with `Lancelot` in terms of performance.

To measure performance, we have chosen to use performance profiles as introduced by Dolan and Moré [14]. They provide a concise mechanism for comparing algorithms over a collection of problems. Roughly, a performance profile chooses a relative metric, e.g., the number of iterations, and then plots the fraction of problems (y-axis) that are solved within a given factor (x-axis) of the best algorithm according to this metric. Roughly speaking, more robust algorithms are "on top" towards the right side of the plot, and more efficient algorithms are "on top" near the left side of the plot. Thus, algorithms that are "on top" throughout the plot may be considered more efficient and more robust, which is the preferred outcome.

The performance profiles in Figures 5.4 and 5.5 compare `AAL-TR`, `BAL-TR`, and `Lancelot` in terms of iterations and gradient evaluations, respectively. (Note that in this comparison, we compare the number of "minor" iterations in `Lancelot` with the number of iterations in our methods.) These figures show that `BAL-TR` performs similarly to `Lancelot` in terms of iterations (Figure 5.4) and gradient evaluations (Figure 5.5). Moreover, it is also clear that our adaptive penalty parameter updating scheme in `AAL-TR` yields better efficiency and robustness when compared to both `BAL-TR` and `Lancelot` on this collection of problems.

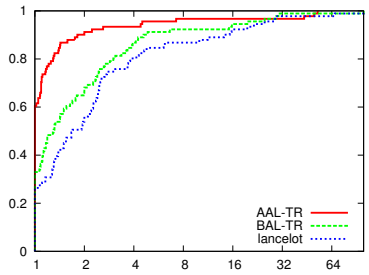

Fig. 5.4 Performance profile for iterations comparing `AAL-TR`, `BAL-TR`, and `Lancelot`.

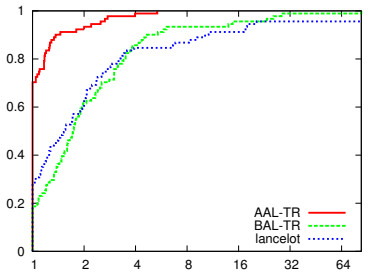

Fig. 5.5 Performance profile for gradient evaluations comparing `AAL-TR`, `BAL-TR`, and `Lancelot`.

Next we compare all four of our implemented methods in Figures 5.6 and 5.7. Figure 5.7 indicates that our adaptive trust region and line search methods are both superior to their traditional counterparts in terms of numbers of required iterations. Figure 5.6 tells a similar story in terms of gradient evaluations.

Finally, it is pertinent to compare the final value of the penalty parameter for `Lancelot` and our adaptive algorithms. We present these outcomes in Table 5.3. The column μ_{final} gives ranges for the final value of the penalty parameter, while the remaining columns give the numbers of problems out of the total 91 whose final penalty parameter fell within the specified range.

We make two observations about the results provided in Table 5.3. First, we observe that our adaptive updating strategy generally does not drive the penalty parameter smaller than does `Lancelot`. This is encouraging since the traditional updating approach used in `Lancelot` is very conservative, so it appears that our

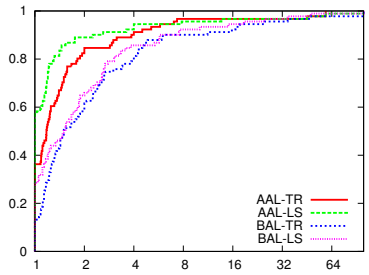

Fig. 5.6 Performance profile for iterations comparing `AAL-TR`, `AAL-LS`, `BAL-TR`, and `BAL-LS`.

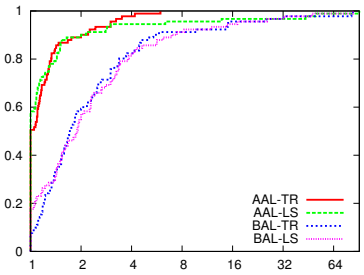

Fig. 5.7 Performance profile for gradient evaluations comparing `AAL-TR`, `AAL-LS`, `BAL-TR`, and `BAL-LS`.

Table 5.3 Numbers of CUTEr problems for which the final penalty parameter value was in the given ranges.

μ_{final}	`Lancelot`	`AAL-TR`	`AAL-LS`
1	7	15	18
$[10^{-1}, 1)$	34	7	7
$[10^{-2}, 10^{-1})$	16	18	12
$[10^{-3}, 10^{-02})$	13	20	19
$[10^{-4}, 10^{-03})$	7	10	8
$[10^{-5}, 10^{-04})$	5	6	3
$[10^{-6}, 10^{-05})$	2	4	8
$[10^{-7}, 10^{-06})$	2	7	7
$\leq 10^{-7}$	5	4	9

adaptive strategy obtains superior performance without driving the penalty parameter to unnecessarily small values. Second, we observe that our strategy maintains the penalty parameter at its initial value ($\mu_0 \leftarrow 1$) on more problems that does `Lancelot`. This phenomenon can be explained by considering the situations in which `Lancelot` and our methods update the penalty parameter. `Lancelot` considers an update once the augmented Lagrangian has been minimized for $y = y_k$ and $\mu = \mu_k$. If the constraint violation is too large and/or the Lagrange multipliers are poor estimates of the optimal multipliers—both of which are likely to be true at the initial point—then the penalty parameter is decreased if/when such a minimizer of the augmented Lagrangian is not sufficiently close to the feasible region. That is, `Lancelot` looks *globally* at the progress towards feasibility obtained from a given point to the next minimizer of the augmented Lagrangian. Our method, on the other hand, observes the *local* reduction in linearized constraint violation. As long as this local reduction is sufficiently large, then no reduction of the penalty parameter occurs, no matter the progress towards nonlinear constraint satisfaction that has occurred so far. Overall, we feel that this behavior illustrated in Table 5.3 highlights a strength of our algorithms, which is that they are less sensitive to the nonlinear constraint violation targets than is `Lancelot`.

6 Conclusion

We have proposed, analyzed, and tested two AL algorithms for large-scale equality constrained optimization. The novel feature of the algorithms is an adaptive strategy for updating the penalty parameter. We have proved that our algorithms are well-posed, possess global convergence guarantees, and outperform traditional AL methods in terms of efficiency on a wide range of test problems. In particular, we believe that the primary strength of our methods is that they are less sensitive than traditional AL methods to a poor choice of the initial penalty parameter, Lagrange multipliers, and nonlinear constraint violation target.

One potential deficiency of our proposed techniques is the lack of a guarantee that the penalty parameter will remain bounded away from zero when they are applied to solve problems where constraint qualifications (e.g., MFCQ) are satisfied at all solution points. We stress that in our numerical experiments, we did not find that our methods lead to unnecessary decreases in the penalty parameter, but this is an important matter to consider in general. Fortunately, as previously mentioned, a simple technique to avoid this issue is to transition from one of our algorithms to a traditional AL approach once consistent progress towards nonlinear constraint satisfaction is observed. In this fashion, the convergence guarantees of traditional AL methods can be relied upon in neighborhoods of solution points.

References

1. Roberto Andreani, Ernesto G. Birgin, José Mario Martínez, and Maria Laura Schuverdt, *Augmented Lagrangian methods under the constant positive linear dependence constraint qualification*, Math. Program., 111 (2008), pp. 5–32.
2. Dimitri P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Computer Science and Applied Mathematics, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982.
3. ———, *Constrained optimization and Lagrange multiplier methods*, Athena Scientific, Belmont, Massachusetts, 1996.
4. Ernesto G. Birgin and J. M. Martínez, *Augmented Lagrangian method with nonmonotone penalty parameters for constrained optimization*, Comput. Optim. Appl., 51 (2012), pp. 941–965.
5. P. T. Boggs and J. W. Tolle, *A family of descent functions for constrained optimization*, SIAM J. Numer. Anal., 21 (1984), pp. 1146–1161.
6. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 1–122.
7. R. H. Byrd, G. Lopez-Calva, and J. Nocedal, *A Line Search Exact Penalty Method Using Steering Rules*, Mathematical Programming, 133 (2012), pp. 39–73.
8. Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz, *Steering exact penalty methods for nonlinear programming*, Optim. Methods Softw., 23 (2008), pp. 197–213.
9. Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint, *A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds*, SIAM J. Numer. Anal., 28 (1991), pp. 545–572.
10. ———, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.
11. Kenneth R. Davidson and Allan P. Donsig, *Real analysis and applications*, Undergraduate Texts in Mathematics, Springer, New York, 2010. Theory in practice.
12. Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
13. G. DiPillo and L. Grippo, *A new class of augmented Lagrangians in nonlinear programming*, SIAM J. Control Optim., 17 (1979), pp. 618–628.

14. Elizabeth D. Dolan and Jorge J. Moré, *Benchmarking optimization software with COPS*, Technical Memorandum ANL/MCS-TM-246, Argonne National Laboratory, Argonne, IL, 2000.
15. Damián Fernández and Mikhail Solodov, *Local convergence of exact and inexact augmented Lagrangian methods under the second-order sufficiency condition*, 2010. IMPA preprint A677.
16. ———, *Stabilized sequential quadratic programming for optimization and a stabilized Newton-type method for variational problems*, Math. Program., 125 (2010), pp. 47–73.
17. Roger Fletcher, *A class of methods for nonlinear programming with termination and convergence properties*, in Integer and Nonlinear Programming, J. Abadie, ed., North-Holland, The Netherlands, 1970, pp. 157–175.
18. D. Gabay and B. Mercier, *A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximations*, Computers and Mathematics with Applications, 2 (1976), pp. 17–40.
19. Philip E. Gill, Walter Murray, and Michael A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Rev., 47 (2005), pp. 99–131.
20. Philip E. Gill and Daniel P. Robinson, *A globally convergent stabilized sqp method*, Numerical Analysis Report 12-02, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2012.
21. R. Glowinski and A. Marroco, *Sur l'Approximation, par Elements Finis d'Ordre Un, el la Resolution, par Penalisation-Dualité, d'une Classe de Problèmes de Dirichlet Nonlineares*, Revue Française d'Automatique, Informatique et Recherche Opérationelle, 9 (1975), pp. 41–76.
22. Nicholas I. M. Gould, D. Orban, and Philippe L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.
23. M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.
24. Alexey F. Izmailov and Mikhail V. Solodov, *On attraction of linearly constrained Lagrangian methods and of stabilized and quasi-Newton SQP methods to critical multipliers*, Math. Program., 126 (2011), pp. 231–257.
25. ———, *Stabilized SQP revisited*, Math. Program., 133 (2012), pp. 93–120.
26. Michal Kočvara and Michael Stingl, *PENNON: a generalized augmented Lagrangian method for semidefinite programming*, in High performance algorithms and software for nonlinear optimization (Erice, 2001), vol. 82 of Appl. Optim., Kluwer Acad. Publ., Norwell, MA, 2003, pp. 303–321.
27. Dong-Hui Li and Liqun Qi, *A stabilized SQP method via linear equations*, technical Report AMR00/5, School of Mathematics, University of New South Wales, Sydney, 2000.
28. El-Sayed Mostafa, Luis Vicente, and Stephen Wright, *Numerical behavior of a stabilized SQP method for degenerate NLP problems*, in Global Optimization and Constraint Satisfaction, Christian Bliek, Christophe Jermann, and Arnold Neumaier, eds., vol. 2861 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 123–141.
29. Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
30. Michael J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, Roger Fletcher, ed., London and New York, 1969, Academic Press, pp. 283–298.
31. Z. Qin, D. Goldfarb, and S. Ma, *An alternating direction method for total variation denoising*, arXiv preprint arXiv:1108.1587, (2011).
32. Stephen J. Wright, *Superlinear convergence of a stabilized SQP method to a degenerate solution*, Comput. Optim. Appl., 11 (1998), pp. 253–275.
33. J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for tvl1-l2 signal reconstruction from partial fourier data*, Selected Topics in Signal Processing, IEEE Journal of, 4 (2010), pp. 288–297.