Sequential equality-constrained optimization for nonlinear programming*

E. G. Birgin† L. F. Bueno‡ J. M. Martínez§

September 9, 2015

Abstract

A new method is proposed for solving optimization problems with equality constraints and bounds on the variables. In the spirit of Sequential Quadratic Programming and Sequential Linearly-Constrained Programming, the new method approximately solves, at each iteration, an equality-constrained optimization problem. The bound constraints are handled in outer iterations by means of an Augmented Lagrangian scheme. Global convergence of the method follows from well-established non-linear programming theories. Numerical experiments are presented.

Key words: Nonlinear programming, Sequential Equality-Constrained Optimization, Augmented Lagrangian, numerical experiments.

1 Introduction

Although nonlinearly constrained optimization is a well-established area of numerical mathematics, many challenges remain that stimulate the development of new methods. Frequently, novel algorithms are firstly developed for equality-constrained optimization

*This work was supported by PRONEX-CNPq/FAPERJ E-26/111.449/2010-APQ1, FAPESP (grants 2010/10133-0, 2013/03447-6, 2013/05475-7, 2013/07375-0, and 2015/02528-8, and CNPq (grants 309517/2014-1 and 303750/2014-6).

†Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, São Paulo, SP, Brazil. e-mail: egbirgin@ime.usp.br

‡Institute of Science and Technology, Federal University of São Paulo, São José dos Campos, SP, Brazil. e-mail: lfelipebueno@gmail.com

§Department of Applied Mathematics, Institute of Mathematics, Statistics, and Scientific Computing, State University of Campinas, Campinas, SP, Brazil. e-mail: martinez@ime.unicamp.br

and, later, their extension to equality and inequality constraints is introduced. Handling inequality constraints may lead to cumbersome combinatorial problems, if one decides to rely on active-set strategies. In Sequential Quadratic Programming (SQP) algorithms [13, 24], a quadratic function is minimized onto a polytope at each iteration. As a consequence, the behavior of SQP is strongly affected by the performance of the Quadratic Programming solver. In Sequential Linearly-Constrained Programming [23], one minimizes a (non-necessarily quadratic) Lagrangian (or augmented Lagrangian) on the polytope that represents the linearization of the constraints. Again, this is a considerably complicated subproblem in which all the combinatorial issues are incorporated to handle inequalities. In the most popular Augmented Lagrangian (AL) methods [1, 9, 11] equality constraints (and sometimes also inequality constraints [1]) are incorporated to the objective function of the subproblems, so that the only inequality constraints of subproblems are represented by the bounds on the variables, thus simplifying the combinatorial difficulties. Finally, in some Interior-Point methods [27], subproblems involve the minimization of a barrier function that tends to infinity on the boundaries, subject to the equality constraints. In this case, the behavior near the boundary is problematic. In Linear Programming, long experience with Interior-Point methods taught software developers how to deal with this inconvenience switching to the "central trajectory", but this is not so simple when we deal with nonlinear constraints.

On the other hand, minimization with only equality constraints is a very attractive subproblem. The main reason is that the standard optimality condition for this problem is a nonlinear system of equations, instead of the system with equalities and inequalities that appear in the KKT conditions of general optimization. In many cases, solving equality constrained subproblems is less difficult than solving quadratic programming subproblems with inequalities of similar dimensions.

This state of facts led us to define a method for general nonlinear programming with equality constrained subproblems, in which the bound constraints on the variables are incorporated to the objective function under the augmented Lagrangian interaction. Using mild criteria for stopping each subproblem, the method proceeds, at the inner iterations, by solving linearizations of Lagrange-like systems and by checking penalty parameters and Lagrange multipliers of the bound constraints at each outer iteration. These parameters are updated as in the AL scheme described in [9]. This is the first implementation of an Augmented Lagrangian method in which the subproblem constraints (or lower-level constraints in [1]) adopt a nontrivial definition, which, by the way, reinforce the necessity of the general theory of [1, 9].

We describe an implementation of these ideas in an algorithm called SECO (Sequential Equality-Constrained Optimization). Section 2 describes the main algorithm. The method for equality-constrained nonlinear minimization that is used for solving the subproblems is described in Section 3. Numerical experiments are described and analyzed in Section 4. Conclusions are given in Section 5.

Notation. If $x \in \mathbb{R}^n$, $x_+ = \max\{x, 0\}$ where the maximum is taken componentwise.

2 SECO algorithm

In this work we address the problem

$$\text{Minimize } f(x) \text{ subject to } h(x) = 0 \text{ and } \ell \leq x \leq u,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are sufficiently smooth. (Note that, paying the price of adding slack variables, inequality constraints of the form $g(x) \leq 0$ can be incorporated as $g(x) + s = 0$ plus $s \geq 0$.) However, to simplify the presentation and since the extension is straightforward, in the rest of this work we will deal with the problem

$$\text{Minimize } f(x) \text{ subject to } h(x) = 0 \text{ and } x \geq 0. \tag{1}$$

For all $x \in \mathbb{R}^n$, $\rho > 0$, and $v \in \mathbb{R}^n_+$, we define the Augmented Lagrangian function $L_\rho(x, v)$ (that only penalizes the non-negativity constraints) by

$$L_\rho(x, v) = f(x) + \frac{\rho}{2} \sum_{v_i/\rho \geq x_i} (v_i/\rho - x_i)^2.$$

Therefore, we have that

$$\nabla L_\rho(x, v) = \nabla f(x) - \sum_{v_i/\rho \geq x_i} (v_i - \rho x_i) e^i$$

and

$$\nabla^2 L_\rho(x, v) = \nabla^2 f(x) + \rho \sum_{v_i/\rho \geq x_i} e^i (e^i)^T,$$

where $e^i \in \mathbb{R}^n$ stands for the i-th canonical vector in $\mathbb{R}^n$. The operators ∇ and ∇^2 always indicate derivatives with respect to x. The Hessian $\nabla^2 L_\rho(x, v)$ is discontinuous at the points in which $x_i = v_i/\rho$ but we can adopt the definition above without leading to contradictions.

The main model algorithm that we now describe is a particular case of Algorithm 4.1 of [9, p.33].

Algorithm 2.1: SECO

Let $v_{\max} > 0$, $\gamma > 1$, $0 < \tau < 1$, $\bar{v}^1 \in [0, v_{\max}]^n$, and $\rho_1 > 0$ be given. Initialize $k \leftarrow 1$.

Step 1. Find $x^k \in \mathbb{R}^n$ as an approximate solution of

$$\text{Minimize } L_{\rho_k}(x, \bar{v}^k) \text{ subject to } h(x) = 0. \tag{2}$$

Step 2. Compute a new approximation of the Lagrange multipliers

$$v^{k+1} = (\bar{v}^k - \rho_k x^k)_+. \tag{3}$$

Step 3. Compute $V^k = \min\{x^k, \bar{v}^k/\rho_k\}$. If $k = 1$ or

$$\|V^k\|_\infty \leq \tau \|V^{k-1}\|_\infty, \tag{4}$$

choose $\rho_{k+1} \geq \rho_k$. Otherwise, choose $\rho_{k+1} \geq \gamma \rho_k$.

Step 4. Choose $\bar{v}^{k+1} \in [0, v_{\max}]^n$.

Step 5. Set $k \leftarrow k + 1$ and go to Step 1.

At Step 1, one tries to find an approximate solution of (2). In fact, at iteration k, we require a point $x^k \in I\!R^n$ such that there exists $\lambda^k \in I\!R^m$ satisfying

$$\begin{aligned} \|\nabla L_{\rho_k}(x^k, \bar{v}^k) + \nabla h(x^k)\lambda^k\|_\infty &\leq \varepsilon_k^{\text{opt}} \\ \|h(x^k)\|_\infty &\leq \varepsilon_k^{\text{feas}}, \end{aligned} \tag{5}$$

where $\{\varepsilon_k^{\text{feas}}\}$ and $\{\varepsilon_k^{\text{opt}}\}$ are sequences that tend to zero in a way that will be specified later.

Theorem 2.1. *Assume that $\{x^k\}$ is a sequence generated by SECO. Then,*

1. *Every limit point of $\{x^k\}$ satisfies the AKKT (Approximate Karush-Kuhn-Tucker) conditions of the problem*

$$\textit{Minimize } \sum_{i=1}^{n} \max\{0, -x_i\}^2 \textit{ subject to } h(x) = 0. \tag{6}$$

2. *If a limit point of $\{x^k\}$ satisfies the Cone-Continuity property [5] with respect to $h(x) = 0$, then this limit point satisfies the KKT conditions of the problem (6).*

3. *If a limit point of $\{x^k\}$ is feasible for problem (1), then it satisfies the AKKT conditions of (1).*

4. *If a limit point of $\{x^k\}$ is feasible and satisfies the Cone-Continuity property with respect to the constraints $h(x) = 0$, $x \geq 0$, then this point satisfies the KKT conditions of (1).*

Proof. The first part follows from Theorem 6.3 of [9]. The second part follows from Theorem 3.3 of [9] or Theorem 2.2 of [5], where the Cone-Continuity property has been defined. The third part of the thesis follows from Theorem 6.2 of [9] and, as the second

part, the last part of the thesis follows from Theorem 3.3 of [9] or Theorem 2.2 of [5]. □

The Cone-Continuity property is the weakest constraint qualification that guarantees that AKKT implies KKT [5]. Therefore, it is weaker than constraint qualifications as LICQ (Linear Independence of the gradients of active constraints), Mangasarian-Fromovitz [20], CPLD (Constant Positive Linear Dependence) [26, 6], RCPLD (Relaxed Positive Linear Dependence) [3], and CPG (Constant Positive Generators) [4].

By the AKKT property [2] and Theorem 6.1 of [9], given $\varepsilon^{\text{feas}} > 0$ and $\varepsilon^{\text{opt}} > 0$, there exists k such that, on success, $(x^k, \lambda^k, v^{k+1})$ satisfies

$$\begin{aligned} \|\nabla f(x^k) - \textstyle\sum_{i=1}^{n} v^{k+1} e^i + \nabla h(x^k)\lambda^k\|_\infty &\leq \varepsilon^{\text{opt}}, \\ \|h(x^k)\|_\infty &\leq \varepsilon^{\text{feas}}, \\ -x^k &\leq \varepsilon^{\text{feas}}, \\ \min\{x^k, v^{k+1}\} &\leq \varepsilon^{\text{feas}}. \end{aligned} \tag{7}$$

3 Algorithm for equality-constrained minimization

In this section, we describe the method being proposed to approximately solve the subproblems (2) of the SECO algorithm.

In order to avoid cumbersome notation, we denote $F(x) = L_{\rho_k}(x, \bar{v}^k)$. Then, F has Lipschitz-continuous first derivatives and $\nabla F(x)$ is semismooth [25]. Problem (2) becomes

$$\text{Minimize } F(x) \text{ subject to } h(x) = 0. \tag{8}$$

Moreover, for the algorithm described in this section we will also use the index k to identify iterations and iterates, that should not be confused with the ones of the main algorithm described in the previous section. We now describe a method that, basically, consists of solving problem (8) by applying Newton's method to its KKT conditions.

The KKT conditions of problem (8) are given by

$$\begin{aligned} \nabla \mathcal{L}(x, \lambda) &= 0 \\ h(x) &= 0, \end{aligned} \tag{9}$$

where $\mathcal{L}(x, \lambda) = F(x) + h(x)^T\lambda$, and, thus, $\nabla \mathcal{L}(x, \lambda) = \nabla F(x) + \nabla h(x)\lambda$.

The Newtonian linear system associated with the nonlinear system of equations (9) is given by

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x, \lambda) & \nabla h(x) \\ \nabla h(x)^T & 0 \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla \mathcal{L}(x, \lambda) \\ h(x) \end{pmatrix}.$$

However, the linear system that we solve at each iteration is of the form

$$\begin{pmatrix} \nabla^2\mathcal{L}(x,\lambda)+\epsilon^{\rm nw}I & \nabla h(x) \\ \nabla h(x)^T & -\epsilon^{\rm se}I \end{pmatrix}\begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = -\begin{pmatrix} \nabla\mathcal{L}(x,\lambda) \\ h(x) \end{pmatrix}, \tag{10}$$

where $\epsilon^{\rm nw}, \epsilon^{\rm se} \geq 0$ are real numbers such that the coefficients' matrix in (10) has n positive eigenvalues and m negative eigenvalues (and no null eigenvalues). The way in which $\epsilon^{\rm nw}$ and $\epsilon^{\rm se}$ are determined will be given below.

At iteration k, given the iterate (x^k, λ^k), a linear system like (10) is solved to find (d_x^k, d_λ^k). Ideally, we would like to compute the new iterate (x^{k+1}, λ^{k+1}) as $(x^k, \lambda^k) + (d_x^k, d_\lambda^k)$. However, in practice, the next iterate will be given by

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ \lambda^k \end{pmatrix} + \alpha_k \begin{pmatrix} \gamma_x^k d_x^k \\ \gamma_\lambda^k d_\lambda^k \end{pmatrix},$$

where $\gamma_x^k, \gamma_\lambda^k \in (0,1]$ have the role of limiting the size of the search directions $\tilde{d}_x^k = \gamma_x^k d_x^k$ and $\tilde{d}_\lambda^k = \gamma_\lambda^k d_\lambda^k$, respectively, and $\alpha_k \in (0,1]$ is computed in order to obtain simple decrease of a merit function not yet specified (for which the search direction d_x^k is a descent direction).

Therefore, the three main ingredients of the developed Newton's method applied to the KKT system (9) of problem (8) are: (a) the correction of the inertia of the coefficients' matrix in (10); (b) the control step (computation of γ_x^k and γ_λ^k); and (c) the backtracking procedure to compute α_k and the decision of the merit function considered in the backtracking.

The description of the algorithm requires the definition of the "squared infeasibility measurement function" given by

$$\Phi(x) = \frac{1}{2}\|h(x)\|_2^2. \tag{11}$$

We also define, in a rather unconventional way, the Augmented Lagrangian function associated with problem (8) as

$$L_\epsilon(x,\lambda) = \epsilon\mathcal{L}(x,\lambda) + \Phi(x),$$

were $\epsilon \geq 0$ is given. The whole method is described in the algorithm below.

Algorithm 3.1.

Let $\varepsilon^{\rm feas}, \varepsilon^{\rm opt} > 0$ be given tolerances related to the stopping criteria. Let $\hat{x}^0$ and λ^0 be initial estimations of the primal and dual variables, respectively. Let PERTURBX0 $\in \{$TRUE, FALSE$\}$ and NOBCKTRCKATALL $\in \{$TRUE, FALSE$\}$ be given parameters. Set $k \leftarrow 0$, NOMORESTPCNTRL $\leftarrow$ FALSE, and NOMOREBCKTRCK $\leftarrow$ FALSE.

Step 0. *Perturbation (or not) of initial guess*

If PERTURBX0 then set

$$x_i^0 = \hat{x}_i^0 + 0.01\xi_i|\hat{x}_i^0|, \text{ for } i = 1, \ldots, n,$$

where ξ_i is a random variable with uniform distribution within the interval $[-1, 1]$ for $i = 1, \ldots, n$. Otherwise, set $x^0 = \hat{x}^0$.

Step 1. *Stopping criteria*

Step 1.1 If

$$\|h(x^k)\|_\infty \leq \varepsilon^{\text{feas}} \text{ and } \|\nabla \mathcal{L}(x^k, \lambda^k)\|_\infty \leq \varepsilon^{\text{opt}}$$

then stop declaring having found an approximate KKT point with the required feasibility and optimality tolerances.

Step 1.2 If

$$\|h(x^k)\|_\infty \leq \varepsilon^{\text{feas}} \text{ and } f(x^k) \leq -10^{10}$$

then stop declaring that the objective function appears to be unbounded from below.

Step 2. *Inertia correction*

Step 2.1.

Case $k = 0$: Set $\epsilon^{\text{nw}} \leftarrow 0$. If $m \leq n$, set $\epsilon^{\text{se}} \leftarrow 0$, otherwise set $\epsilon^{\text{se}} \leftarrow 10^{-8}$.

Case $k \geq 1$: If $\gamma_x^{k-1} = 1$, set $\epsilon^{\text{nw}} \leftarrow 0.1\epsilon_{k-1}^{\text{nw}}$, otherwise set $\epsilon^{\text{nw}} \leftarrow 3\epsilon_{k-1}^{\text{nw}}$. Set $\epsilon^{\text{se}} \leftarrow 0.1\epsilon_{k-1}^{\text{se}}$.

Step 2.2. Compute the inertia (i_+, i_-, i_0), where i_+, i_-, and i_0 represent the number of positive, negative, and null eigenvalues, respectively, of matrix

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x^k, \lambda^k) + \epsilon^{\text{nw}} I & \nabla h(x^k) \\ \nabla h(x^k)^T & -\epsilon^{\text{se}} I \end{pmatrix}. \tag{12}$$

Step 2.3. If $i_+ < n$ then increase ϵ^{nw} by setting $\epsilon^{\text{nw}} \leftarrow \max\{10^{-8}, 3\epsilon^{\text{nw}}\}$.

Step 2.4. If $i_- < m$ then increase ϵ^{se} by setting $\epsilon^{\text{se}} \leftarrow \max\{\ 10^{-8}, 3\epsilon^{\text{se}}\}$.

Step 2.5. If ϵ^{nw} or ϵ^{se} were increased at Steps 2.3 or 2.4, respectively, go to Step 2.2.

Step 2.6. Define $\epsilon_k^{\text{nw}} = \epsilon^{\text{nw}}$ and $\epsilon_k^{\text{se}} = \epsilon^{\text{se}}$.

Step 3. *Search direction and step control*

Step 3.1. Compute d_x^k and d_λ^k as the (unique) solution of the linear system

$$\begin{pmatrix} \nabla^2 \mathcal{L}(x^k, \lambda^k) + \epsilon_k^{\text{nw}} I & \nabla h(x^k) \\ \nabla h(x^k)^T & -\epsilon_k^{\text{se}} I \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla \mathcal{L}(x^k, \lambda^k) \\ h(x^k) \end{pmatrix}.$$

Step 3.2. If NoMoreStpCntrl, define $\gamma_x^k = 1$ and $\gamma_\lambda^k = 1$. Otherwise, define

$$\gamma_x^k = \min\left\{1, \frac{100 \max\{1, \|x^k\|_\infty\}}{\|d_x^k\|_\infty}\right\} \text{ and } \gamma_\lambda^k = \min\left\{1, \frac{100 \max\{1, \|\lambda^k\|_\infty\}}{\|d_\lambda^k\|_\infty}\right\}.$$

Step 3.3. Define $\tilde{d}_x^k = \gamma_x^k d_x^k$ and $\tilde{d}_\lambda^k = \gamma_\lambda^k d_x^k$.

Step 4. *Backtracking (on primal variables only)*

Step 4.1. If NoBcktrckAtAll or NoMoreBcktrck, define $\alpha_k = 1$ and go to Step 5.

Step 4.2. Set $\alpha \leftarrow 1$.

Step 4.3. Set $x_{\text{trial}} \leftarrow x^k + \alpha \tilde{d}_x^k$.

Step 4.4. Consider conditions

$$\|h(x^k)\|_\infty > \varepsilon^{\text{feas}} \;\text{ and }\; L_{\epsilon_k^{\text{se}}}(x_{\text{trial}}, \lambda^k) \le L_{\epsilon_k^{\text{se}}}(x^k, \lambda^k), \tag{13}$$

$$\|h(x^k)\|_\infty \le \varepsilon^{\text{feas}},\ \epsilon_k^{\text{se}} = 0, \;\text{ and }\; \mathcal{L}(x_{\text{trial}}, \lambda^k) \le \mathcal{L}(x^k, \lambda^k), \tag{14}$$

and

$$\|h(x^k)\|_\infty \le \varepsilon^{\text{feas}},\ \epsilon_k^{\text{se}} \neq 0, \;\text{ and }\; L_{\epsilon_k^{\text{se}}}(x_{\text{trial}}, \lambda^k) \le L_{\epsilon_k^{\text{se}}}(x^k, \lambda^k). \tag{15}$$

If (13), (14), or (15) hold, define $\alpha_k = \alpha$ and go to Step 5.

Step 4.5. Set $\alpha \leftarrow \alpha/2$ and go to Step 4.3.

Step 5. *Update the iterate and iterate*

Step 5.1. Define $x^{k+1} = x^k + \alpha_k \tilde{d}_x^k$ and $\lambda^{k+1} = \lambda^k + \alpha_k \tilde{d}_\lambda^k$.

Step 5.2. Consider conditions

$$\text{fl}(L_{\epsilon_k^{\text{se}}}(x^{k+1}, \lambda^k)) = \text{fl}(L_{\epsilon_k^{\text{se}}}(x^k, \lambda^k)) \tag{16}$$

and

$$\text{fl}(\mathcal{L}(x^{k+1}, \lambda^k)) = \text{fl}(\mathcal{L}(x^k, \lambda^k)), \tag{17}$$

where $\text{fl}(\cdot)$ represents the result of performing an operation in floating point arithmetic. If (13,16), (14,17), or (15,16) hold, set NoMoreStpCntrl $\leftarrow$ true and NoMoreBcktrck $\leftarrow$ true.

Step 5.3. Set $k \leftarrow k + 1$ and go to Step 1.

Remarks. It is well-known that backtracking may be harmful in some well characterized situations. This is why the algorithm includes the possibility of avoiding backtracking by setting the logical parameter NoBcktrckAtAll. Perturbation of the initial guess may be adequate to avoid some undesirable situations in which symmetry prevents convergence. This is the reason why the method includes the possibility of perturbing the initial guess by setting the logical parameter Perturbx0.

Algorithm 3.1 may be considered as an heuristic Newton-based procedure for solving the equality-constrained minimization problem (8). For obtaining global convergence,

this algorithm may be naturally coupled with the globally convergent Flexible Inexact-Restoration (FIR) procedure for solving (8) introduced in [10]. Each iteration of the FIR algorithm, as other Inexact-Restoration methods, has two phases: Feasibility and Optimality. In the Feasibility Phase the algorithm aims to improve feasibility and in the Optimality Phase the algorithm minimizes approximately a Lagrangian approximation subject to the linearization of the constraints [7, 12, 14, 15, 16, 17, 18, 19, 21, 22]. As a consequence, a trial point is obtained which is accepted as new iterate or not according to the value of a Sharp Lagrangian merit function [12, 21]. When the trial point is not accepted, a new trial point is obtained that satisfies the linearized constraints. If the feasible phase is well-defined, it can be proved that the algorithm generates AKKT sequences and, so, that limit points are KKT when the Cone-Continuity property holds. The conditions required in the Feasibility Phase are mild and, thus, many different (perhaps heuristic) methods may be employed for this phase. Our proposal here essentially consists of employing Algorithm 3.1 as the feasibilization procedure of the FIR method. More precisely, the globalization procedure can be sketched in the following Algorithm 3.2.

Algorithm 3.2

Initialize the standard algorithmic parameters of FIR and the parameters of Algorithm 3.1. Let $N > 0$, $\zeta \in (0, 1)$, and $\upsilon > 0$. Initialize $k \leftarrow 0$.

Step 1. Run Algorithm 3.1 employing a maximum of N iterations and obtaining the point y^{k+1}. If the standard stopping criterion of Algorithm 3.1 is satisfied, stop. Otherwise, if $\|h(y^{k+1})\| \leq \zeta \|h(x^k)\|$ and $\|y^{k+1} - x^k\| \leq \upsilon \|h(x^k)\|$ go to Step 2, else stop declaring Failure in the Feasibility Phase.

Step 2. Proceed as in FIR updating the penalty parameter, minimizing the Lagrangian in the tangent set and making the necessary comparisons by means of which, eventually, x^{k+1} is obtained. Update $k \leftarrow k + 1$ and go to Step 1.

4 Numerical experiments

We implemented Algorithms 2.1 and 3.1 in Fortran 90. This means that the SECO subproblems will be solved by an heuristic method and that the globalization scheme suggested in the previous section will not be considered; the main reason for this choice has been keeping the implemented method simple. All tests were conducted on a computer with 3.5 GHz Intel Core i7 processor and 16GB 1600 MHz DDR3 RAM memory, running OS X Yosemite (version 10.10.4). Codes were compiled by the GFortran Fortran compiler of GCC (version 4.9.2) with the -O3 optimization directive enabled.

Regarding the parameters of Algorithm 2.1, arbitrarily but based on previous experimentation with the Augmented Lagrangian solver Algencan [1, 9], we set $v_{\max} = 10^{20}$, $\gamma = 10$, and $\tau = 0.5$. We also set $\varepsilon^{\text{feas}} = \varepsilon^{\text{opt}} = 10^{-8}$. The choice of the sequences $\{\varepsilon_k^{\text{feas}}\}$ and

$\{\varepsilon_k^{\rm opt}\}$ (see (5)) is given by

$$\varepsilon_1^{\rm feas} = \sqrt{\varepsilon^{\rm feas}} \quad \text{and} \quad \varepsilon_1^{\rm opt} = \sqrt{\varepsilon^{\rm opt}}$$

and, for $k > 1$, if

$$\|h(x^{k-1})\|_\infty \leq \sqrt{\varepsilon^{\rm feas}} \text{ and } \|\nabla \mathcal{L}(x^{k-1}, \lambda^{k-1})\|_\infty \leq \sqrt{\varepsilon^{\rm opt}}$$

then

$$\begin{aligned} \varepsilon_k^{\rm feas} &= \max\{\varepsilon^{\rm feas}, \min\{\tfrac{1}{10}\varepsilon_{k-1}^{\rm feas}, \tfrac{1}{2}\|h(x^{k-1})\|_\infty\}\} \\ \varepsilon_k^{\rm opt} &= \max\{\varepsilon^{\rm opt}, \min\{\tfrac{1}{10}\varepsilon_{k-1}^{\rm opt}, \tfrac{1}{2}\|\nabla \mathcal{L}(x^{k-1}, \lambda^{k-1})\|_\infty\}\}. \end{aligned}$$

Otherwise, $\varepsilon_k^{\rm feas} = \varepsilon_{k-1}^{\rm feas}$ and $\varepsilon_k^{\rm opt} = \varepsilon_{k-1}^{\rm opt}$. The value of the initial penalty parameter ρ_1, follows exactly the settings considered in Algencan (see [9, p.153]). In this way, differences between SECO and Algencan are concentrated in the choice of lower- and upper-level constraints, making their comparison useful for the purposes of the present work. The stopping criterion of Algorithm 2.1 is given by (7).

Subroutine MA57 from HSL [28] was used to compute the inertia-revealing factorizations at Step 2.2 of Algorithm 3.1 and to solve the linear systems at Step 3.1. The values of parameters PERTURBX0 $\in \{$TRUE, FALSE$\}$ and NOBCKTRCKATALL $\in \{$TRUE, FALSE$\}$ will be the subject of numerical experimentation.

4.1 When the initial point should be perturbed

Consider the problem

$$\text{Minimize } (x + y - 10)^2 \text{ subject to } xy = 1. \tag{18}$$

Let $\delta = \sqrt{10^2 - 4} \approx 9.8$, $r_1 = (10 + \delta)/2 \approx 9.9$, and $r_2 = (10 - \delta)/2 \approx 0.1$. It is easy to see that $(r_1, 1/r_1)^T$ and $(r_2, 1/r_2)^T$ are global minimizers of problem (18). They both annihilate the objective function, satisfy the LICQ constraint qualification (since any feasible point satisfies it), and, hence, satisfy the KKT condition (both with null multiplier). Should not any optimization method have them as a target?

Since the point $(5, 5)^T$ is an unconstrained global minimizer (annihilates the objective function but does not satisfy the constraint), it might be a natural initial guess for any iterative solver trying to solve problem (18). However, starting from $\hat{x}^0 = (5, 5)^T$ (with $\lambda^0 = 0$, PERTURBX0 = NOBCKTRCKATALL = FALSE), in 9 iterations and never abandoning the line $x = y$, Algorithm 3.1 converges to the KKT point $(1, 1)^T$, that is local *maximizer* (there is another local maximizer at $(-1, -1)^T$ and the objective function goes to infinity within the feasible set when, for example, x goes to infinity and $y = 1/x$).

Some readers may think that having found a local maximizer is not an issue at all since a KKT point was found. However, from the authors' point of view, this is, at least, an

undesired situation, since the Optimizers' main goal is, in most cases, to obtain the lowest possible functional values within the feasible region.

In this subsection we claim that a very simple and affordable way of avoiding the behaviour described above is to slightly perturb the initial guess. When the initial point $\hat{x}^0 = (5, 5)^T$ is perturbed, as described at Step 0 of Algorithm 3.1 (and the actual initial point given by $x^0 \approx (4.9534, 5.0371)^T$), Algorithm 3.1 (with $\lambda^0 = 0$, PERTURBX0 = TRUE, and NOBCKTRCKATALL = FALSE) converges to the approximate global minimizer $(0.10102, 9.98990)^T \approx (r_2, 1/r_2)^T$ in 4 iterations.

4.2 When backtracking should be avoided

Algorithm 3.1 presented in Section 3 aims to solve problems of the form (8) with

$$F(x) = L_{\rho_k}(x, \bar{v}^k) = f(x) + \frac{\rho_k}{2} \sum_{v_i/\rho_k \geq x_i} (\bar{v}_i^k/\rho_k - x_i)^2.$$

As a consequence, it is natural to evaluate Algorithm 3.1 considering objective functions of this form in which ρ_k is possibly large. Thus, in the present section, we consider problems of the form

$$\begin{array}{ll} \text{Min} & \frac{1}{2}x^T Ax + b^T x + r \sum_{i=1}^n \max\{0, -x_i\}^2 \\ \text{subject to} & \frac{1}{2}x^T A_j x + b_j^T x + c_j = 0,\ j = 1, \ldots, m, \end{array} \tag{19}$$

where

$$A = \left[\frac{1}{\max\{1, \max_{i,j}\{\bar{a}_{ij}\}\}}\right] (\bar{A}^T \bar{A}) \text{ and } A_j = \frac{1}{2}(\bar{A}_j^T + \bar{A}_j), \tag{20}$$

and $\bar{A} \in I\!R^{n\times n}$, $b \in I\!R^n$, $\bar{A}_j \in I\!R^{n\times n}$, and $b_j \in I\!R^n$ for $j = 1, \ldots, m$ have random elements with uniform distribution within the interval $[-1, 1]$. In order to generate a feasible problem, a point $\bar{x}$ with random non-negative elements $\bar{x}_i$ with uniform distribution in $[0, 1]$ is generated and $c_j \in I\!R$ is defined as

$$c_j = -[\tfrac{1}{2}\bar{x}^T A_j \bar{x} + b_j^T \bar{x}] \text{ for } j = 1, \ldots, m. \tag{21}$$

We consider 11 instances of problem (19) with $n = 1\,000$, $m = 500$, and $r \in \{0, 1, 10, 100, \ldots, 10^{10}\}$. The initial point x^0 is always given by

$$x_i^0 = \bar{x}_i + 10^{-8}\xi_i|\bar{x}_i|, \text{ for } i = 1, \ldots, n,$$

where ξ_i is a random variable with uniform distribution in $[-1, 1]$, i.e. x^0 is a very slight perturbation of the known feasible point used to generate the problem. At x^0 we have that $\max_{1\leq j\leq m}\{|x^T A_j x + b_j^T x + c_j|\} \approx 10^{-6} \not\leq \varepsilon^{\text{feas}} = 10^{-8}$. With these settings, the situation we are trying to mimic is the following: (a) Since the Jacobian of the constraints $h'(x^k)$ has

full row rank with high probability, we will have ϵ_k^{se} (the perturbation applied to the south-east block of the Jacobian matrix (12) of the KKT system to correct its inertia) equal to 0 for all k; (b) As a consequence, the primal direction d_x^k will satisfy $h'(x^k)d_x^k + h(x^k) = 0$ for all k and, since $\|h(x^k)\|$ is relatively "small", we will have $h'(x^k)d_x^k \approx 0$; and (c) With high probability, the merit function (that coincides with the feasibility measure $\frac{1}{2}\|h(x)\|_2^2$ if x^k is infeasible (i.e. $\|h(x^k)\|_\infty \not\leq \varepsilon^{\text{feas}}$) and $\epsilon_k^{\text{se}} = 0$) will decrease only for very small steps along d_x^k. This *gedanken* experiment illustrates a well-known inconvenience of the usage of merit functions that gave rise to the development of alternative merit functions, nonmonotone strategies and filter methods, among others. Our claim at the present moment is: try Newton as pure as possible avoiding other alternatives.

Table 1 shows the results of applying Algorithm 3.1 with PERTURBX0 = FALSE and NOBCKTRCKATALL = TRUE to the eleven instances described in the paragraph above. The results for the case PERTURBX0 = TRUE and NOBCKTRCKATALL = TRUE are very similar to those displayed in Table 1, i.e. as expected, a small perturbation in the initial guess has no meaningful effect in the behavior of the method, since symmetry is not an issue in this problem.

r	# it	# fcnt	CPU time (s)	$f(x^*)$	$\min_{\{1\leq i\leq n\}}\{x_i^*\}$
0	45	46	142.92	5.5997D+03	-2.0293D+00
1	74	75	233.22	5.5683D+03	-2.2747D+00
10	57	58	179.92	7.2960D+03	-1.6530D+00
10^2	57	58	174.40	1.3483D+04	-8.7815D−01
10^3	85	86	268.49	1.9558D+04	-2.0632D−01
10^4	90	91	273.76	2.1172D+04	-2.9567D−02
10^5	121	122	377.82	2.1376D+04	-3.1172D−03
10^6	76	77	239.00	2.1418D+04	-3.2517D−04
10^7	127	128	397.06	2.1656D+04	-2.8549D−05
10^8	130	131	412.17	2.1953D+04	-2.5745D−06
10^9	332	333	1057.63	2.1486D+04	-2.8450D−07
10^{10}	169	170	534.56	2.1922D+04	-2.9008D−08

Table 1: Performance of Algorithm 3.1 with PERTURBX0 = FALSE and NOBCKTRCK-ATALL = TRUE applied to eleven instances of problem (19) with $r \in \{0, 1, 10, \ldots, 10^{10}\}$.

On the other hand, the behavior of the method with NOBCKTRCKATALL = FALSE and PERTURBX0 = FALSE (in order to preserve the properties of the picked initial guess) is completely different. For the case $r = 0$ (the only one we evaluate), the method get virtually stuck at the initial guess. In its first 30 iterations, it performs 24 functional evaluations per iteration and takes a step of size $2^{-23} \approx 10^{-7}$. The squared Euclidian norm of the constraints is approximately 1.5275×10^{-10} (the sup-norm is approx. $2.2899 \times$

10^{-6}) at the initial guess and, after 30 iterations, those values are exactly the same, showing that the method made no progress at all. This situation represents an extreme case. However, similar results can be observed when the initial guess x^0 has random elements with uniform distribution within the interval $[0, 1]$. With no backtracking at all, Algorithm 3.1, when applied to problem (19) with $r = 0$, satisfies the stopping criterion using 83 iterations (and 84 functional evaluations) using 255.00 seconds of CPU time. Similarly to the values reported in Table 1 for a different initial guess, the objective functional value at the final point is approximately 5.2558×10^3 and the value of the most negative entrance of the final iterate is approximately -2.0929. On the other hand, with backtracking, the algorithm makes an excruciatingly slow progress satisfying the stopping criterion after 6 792 iterations that require 106 795 functional evaluations (in average, approximately 16 per iterations) and consume 49 977.22 seconds of CPU time (approx. 14 hours). The achieved objective functional value is approximately 5.7927×10^3 and the value of the most negative entrance of the final iterate is approximately -2.2919.

4.3 Problems with a variable number of bound constraints

In this section, we consider the problem

$$\begin{array}{ll} \text{Min} & \frac{1}{2}x^T Ax + b^T x \\ \text{subject to} & \frac{1}{2}x^T A_j x + b_j^T x + c_j = 0,\ j = 1, \ldots, m, \\ & x_i \geq 0,\ i \in I, \end{array} \tag{22}$$

where $I \subseteq \{1, \ldots, n\}$. As in the previous section, A and A_j ($j = 1, \ldots, m$) are given by (20) and $\bar{A} \in I\!R^{n\times n}$, $b \in I\!R^n$, $\bar{A}_j \in I\!R^{n\times n}$, and $b_j \in I\!R^n$ for $j = 1, \ldots, m$ have random elements with uniform distribution within the interval $[-1, 1]$. Moreover, in order to obtain a feasible problem, $c_j \in I\!R$ is given by (21), where $\bar{x}$ has non-negative elements $\bar{x}_i$ with uniform distribution in $[0, 1]$. The considered initial guess x^0 has random elements with uniform distribution within the interval $[0, 1]$.

With the aim of generating a set of instances with solutions with an increasing number of active bound constraints, we consider instances with an increasing number of bound constraints. At a given instance, the probability of $i \in I$ is given by a constant $p \in [0, 1]$ (for $i = 1, \ldots, n$) and six instances with $p \in \{0, 0.01, 0.1, 0.5, 0.75, 1\}$ are considered. In all cases we set $n = 1\,000$ and $m = 500$. All instances were solved with Algencan 3.0.0 (inhibiting its acceleration process [8]) and SECO. Both methods found an approximate stationary point in all the six instances. Table 2 shows the results. In the table, "# act" is the number of active bound constraints at the final iterate x^*. In the case of Algencan, this number is given by the cardinality of the set $\{i \mid x_i^* = 0\}$; while in SECO it is given by the cardinality of the set $\{i \mid |x_i^*| \leq \varepsilon^{\text{feas}} = 10^{-8}\}$. Column "# it" displays the number of outer iterations and the number of inner iterations separated by a slash. "fcnt",

"gcnt", and "hcnt" represent the number of functional evaluations and first (gradients) and second-order (Hessians) derivatives evaluations, respectively. "CPU time (s)" means the elapsed CPU time in seconds and $f(x^*)$ is the value of the objective function at the final iterate. The sup-norm of the equality constraints as well as the sup-norm of the gradient of the Lagrangian are always smaller that $\varepsilon^{\text{feas}}$ and ε^{opt}, respectively (recall that both tolerances are equal to 10^{-8}). Last but not least, bound constraints are satisfied with zero tolerance in Algencan (meaning that $x^* \geq 0$ holds); while SECO satisfies the bound constraints with tolerance $\varepsilon^{\text{feas}}$ (meaning that $x_i^* \geq -\varepsilon^{\text{feas}}$ for $i = 1, \ldots, n$). Analyzing the figures in the table, it can be seen that SECO found smaller functional values in the cases $p = 0$ and $p = 1$; while Algencan found smaller functional values in the remaining four cases. If, on the one hand, the smaller functional values found by SECO may be attributed to the admissible enlarged bound constraints, on the other hand Algencan found smaller functional values more times. This means that, in this experiment, Algencan appears to exhibit a larger tendency to find better stationary points than SECO. On the efficiency side, both methods appear to require similar efforts for the case $p = 1$ (instance with the largest number of active constraints at the final iterates); while SECO is considerably faster in the other instances. Thus, disregarding the functional values, if the goal were to find stationary points, using SECO would be a reasonable choice.

Algencan 3.0.0 without the acceleration process								
p	$\|I\|$	# act	# it	fcnt	gcnt	hcnt	CPU time (s)	$f(x^*)$
0	0	0	12 / 1083	4258	1141	1083	3741.28	5.3520D+03
0.01	10	3	14 / 1175	4709	1255	1172	8306.38	5.3294D+03
0.1	109	36	13 / 1185	3266	1243	1175	5930.13	6.9063D+03
0.5	492	182	5 / 822	1794	851	812	2539.07	1.1931D+04
0.75	735	253	6 / 895	1773	918	883	2249.93	1.6419D+04
1	1000	302	7 / 535	1034	560	523	1087.35	2.1492D+04

SECO (Algorithms 2.1 and 3.1) with no backtracking								
p	$\|I\|$	# act	# it	fcnt	gcnt	hcnt	CPU time (s)	$f(x^*)$
0	0	0	1 / 83	84	84	83	261.78	5.2558D+03
0.01	10	5	2 / 66	71	71	66	202.94	5.7933D+03
0.1	109	42	3 / 106	113	113	106	314.55	7.1996D+03
0.5	492	183	3 / 50	57	57	50	155.13	1.2821D+04
0.75	735	258	3 / 271	278	278	271	818.61	1.6527D+04
1	1000	299	3 / 348	355	355	348	1063.91	2.1418D+04

Table 2: Performance of Algencan 3.0.0 without the acceleration process and SECO (Algorithms 2.1 and 3.1) with no backtracking applied to six instances of problem (22) with $n = 1000$, $m = 500$, and $|I| \approx pn$ and $p \in \{0, 0.01, 0.1, 0.5, 0.75, 1\}$.

5 Conclusions

General Augmented Lagrangian methods [1, 9] are based on the partition of the constraints of the nonlinear programming problem into two sets; the first one is defined by

$$h(x) = 0 \text{ and } g(x) \leq 0 \tag{23}$$

and the second is given by

$$\underline{h}(x) = 0 \text{ and } \underline{g}(x) \leq 0, \tag{24}$$

where $h : I\!R^n \to I\!R^m$, $g : I\!R^n \to I\!R^p$, $\underline{h} : I\!R^n \to I\!R^{\underline{m}}$, and $\underline{g} : I\!R^n \to I\!R^{\underline{p}}$. The subproblems of the AL algorithm consist on the minimization of the Augmented Lagrangian defined by f, h, and g subject to the constraints (24).

The constraints (24) have been called "lower-level constraints", "simple constraints", or "subproblem constraints" in different papers. In most practical algorithms the constraints (24) are really simple, for example, when they define a box. However, in the present paper we show that the denomination "simple" for the constraints (24) may not be adequate because it may be attractive the constraints (24) to be "more complicated" than the constraints (23).

To fix ideas, suppose that the constraints (23) are defined by

$$h(x) = 0 \tag{25}$$

and the constraints (24) are

$$\ell \leq x \leq u. \tag{26}$$

Every constrained optimization problem can be expressed as the minimization of a function $f(x)$ subject to (25) and (26), using slack variables. The conventional wisdom is that, if we adopt the AL strategy, we should solve this problem by means of subproblems that minimize Augmented Lagrangians defined by f and h subject to (26). In this paper we explored the idea of solving the problem by means of successive minimizations of the Augmented Lagrangian defined by f and the functions $x - u$ and $\ell - x$, subject to the constraints (25). Of course we made our best to define a method for solving the equality-constrained subproblems by means of which the overall algorithm should be competitive. The resulting algorithm have been compared with the well-established software Algencan.

The conclusions of the numerical study depend on the definition of "satisfactory solution". If we adopt the criterion that a satisfactory solution is a point that satisfies approximately the KKT conditions with high precision, we can establish that the new algorithm is more efficient than Algencan when the number of active bounds is small, but its relative superiority in terms of efficiency decreases as the number of active constraints at the solution increases. The considerations that explain this behavior are the following:

1. In the extreme case in which there are no active inequality constraints the algorithm essentially solves a smooth nonlinear system of equations using a safeguarded Newton-like method. It is not surprising that this type of procedure should be more efficient than the possibly painful process of increasing the penalty parameter several times in order to minimize Augmented Lagrangians defined by f and h.

2. If many inequality constraints are active at the solution, even the algorithm for minimizing with equality constraints being efficient, several outer iterations are generally needed, because many bound constraints are not satisfied after each outer iteration. Moreover, the nonlinear system that represents the Lagrange conditions for that subproblem is semismooth, but not smooth, decreasing partially the efficiency of Newton-like algorithms. On the other hand, the classical implementation of Algencan deals efficiently with the active bound constraints, avoiding useless evaluations of f and h outside the feasible box.

Nevertheless, if we adopt the criterion that algorithms should be compared also with respect to the objective function value achieved at the end of execution, Algencan seems to be better than SECO. This can be explained by the opportunistic line-search strategies used by the bound-constrained solver employed by Algencan, which actively searches low values of the objective function with procedures that are not necessarily linked to worst-case convergence analysis.

We would like to finish this paper with strong statements concerning the relative efficiency or robustness of the new method with respect to existing ones. However, we are convinced that such statements are not possible in nonlinear optimization problems. There exist an infinitely large number of geometrical combinations, structures, and dimensions that make it impossible universal superiority claims. Therefore, we conclude with the cautious claim that the SECO approach may be a useful tool when the problems are formulated with the constraints "$h(x) = 0$ plus bounds" and the number of expected active bounds at the solution is moderate.

References

[1] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, On augmented Lagrangian methods with general lower-level constraints, *SIAM Journal on Optimization* 18, pp. 1286–1309, 2008.

[2] R. Andreani, G. Haeser, and J. M. Martínez. On sequential optimality conditions for smooth constrained optimization, *Optimization* 60, pp. 627–641, 2011.

[3] R. Andreani, G. Haeser, M. L. Schuverdt, and P. J. S. Silva. A relaxed constant positive linear dependence constraint qualification and applications, *Mathematical Programming* 135, pp. 255–273, 2012.

[4] R. Andreani, G. Haeser, M. L. Schuverdt, and P. J. S. Silva. Two new weak constraint qualifications and applications, *SIAM Journal on Optimization* 22, pp. 1109–1135, 2012.

[5] R. Andreani, J. M. Martínez, A. Ramos, and P. J. S. Silva. Constraint qualifications and approximate KKT-sequences, Contributed presentation at X Brazilian Workshop on Continuous Optimization, Florianópolis, Santa Catarina, Brazil, 2014.

[6] R. Andreani, J. M. Martínez, and M. L. Schuverdt, On the relation between Constant Positive Linear Dependence Condition and Quasinormality Constraint Qualification, *Journal of Optimization Theory and Applications* 125, pp. 473–485, 2005.

[7] N. Banihashemi and C. Y. Kaya, Inexact Restoration for Euler Discretization of Box-Constrained Optimal Control Problems, *Journal of Optimization Theory and Applications* 156, pp. 726–760, 2013.

[8] E. G. Birgin and J. M. Martínez, Improving ultimate convergence of an Augmented Lagrangian method, *Optimization Methods and Software* 23, pp. 177–195, 2008.

[9] E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*, SIAM, Philadelphia, 2014.

[10] L. F. Bueno, G. Haeser, and J. M. Martnez, A Flexible Inexact Restoration Method for Constrained Optimization, *Journal of Optimization Theory and Applications* 165, pp. 188–208, 2015.

[11] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Lancelot: A Fortran package for large scale nonlinear optimization*, Springer-Verlag, Berlin, 1992.

[12] A. Fischer and A. Friedlander, A new line search inexact restoration approach for nonlinear programming, *Computational Optimization and Applications* 46, pp. 333–346, 2010.

[13] R. Fletcher, *Practical Methods of Optimization*, 2nd edition, Wiley, 2000.

[14] J. B. Francisco, J. M. Martínez, L. Martínez, and F. Pisnitchenko, Inexact Restoration method for minimization problems arising in electronic structure calculations, *Computational Optimization and Applications* 50, pp. 555–590, 2011.

[15] C. C. Gonzaga, E. W. Karas, and M. Vanti, A globally convergent filter method for Nonlinear Programming, *SIAM Journal on Optimization* 14, pp. 646–669, 2004.

[16] E. W. Karas, C. C. Gonzaga, and A. A. Ribeiro, Local convergence of filter methods for equality constrained non-linear programming, *Optimization* 59, pp. 1153–1171, 2010.

[17] E. W. Karas, E. A. Pilotta, and A. A. Ribeiro, Numerical comparison of merit function with filter criterion in inexact restoration algorithms using Hard-Spheres Problems, *Computational Optimization and Applications* 44, pp. 427–441, 2009.

[18] C. Y. Kaya, Inexact Restoration for Runge-Kutta discretization of Optimal Control problems, *SIAM Journal on Numerical Anlysis* 48, pp. 1492–1517, 2010.

[19] C. Y. Kaya and J. M. Martínez, Euler discretization and Inexact Restoration for Optimal Control, *Journal of Optimization Theory and Application* 134, pp. 191–206, 2007.

[20] O. L. Mangasarian and S. Fromovitz, The Fritz-John necessary optimality conditions in presence of equality and inequality constraints, *Journal of Mathematical Analysis and Applications* 17 pp. 37–47, 1967.

[21] J. M. Martínez, Inexact-Restoration method with Lagrangian tangent decrease and new merit function for nonlinear programming, *Journal of Optimization Theory and Application* 111, 39–58, 2001.

[22] J. M. Martínez and E. A. Pilotta, Inexact-Restoration algorithms for constrained optimization, *Journal of Optimization Theory and Application* 104, pp. 135–163, 2000.

[23] B. A. Murtagh and M. A. Saunders, Large-scale linearly constrained optimization, *Mathematical Programming* 14, pp. 41–72, 1978.

[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd edition, Springer, 2006.

[25] L. Qi and J. Sun, A nonsmooth version of Newton's method, *Mathematical Programming* 58, pp. 353–367, 1993.

[26] L. Qi and Z. Wei, On the constant linear dependence condition and its application to SQP methods, *SIAM Journal on Optimization* 10, pp. 963–981, 2000.

[27] A. Wächter and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming* 106, pp. 25-57, 2006.

[28] HSL(2013), *A collection of Fortran codes for large scale scientific computation*, `http://www.hsl.rl.ac.uk`.