Steering Exact Penalty Methods for Nonlinear Programming

Richard H. Byrd*   Jorge Nocedal†   Richard A. Waltz†

April 10, 2007

Technical Report
Optimization Technology Center
Northwestern University
Evanston, IL 60208

Abstract

This paper reviews, extends and analyzes a new class of penalty methods for nonlinear optimization. These methods adjust the penalty parameter dynamically; by controlling the degree of linear feasibility achieved at every iteration, they promote balanced progress toward optimality and feasibility. In contrast with classical approaches, the choice of the penalty parameter ceases to be a heuristic and is determined, instead, by a subproblem with clearly defined objectives. The new penalty update strategy is presented in the context of sequential quadratic programming (SQP) and sequential linear-quadratic programming (SLQP) methods that use trust regions to promote convergence. The paper concludes with a discussion of penalty parameters for merit functions used in line search methods.

1

1   Introduction

In recent years, there has been a resurgence of interest in exact penalty methods [1, 2, 10, 25, 28, 31, 37] because of their ability to handle degenerate problems and inconsistent constraint linearizations. Exact penalty methods have been used successfully to solve mathematical programs with complementarity constraints (MPCCs) [3, 30], a class of problems that do not satisfy the Mangasarian-Fromovitz constraint qualification at any feasible point. They are also used in nonlinear programming algorithms to ensure the feasibility of subproblems and to improve the robustness of the iteration [6, 24]. In this paper we discuss a new strategy for choosing the penalty parameter that overcomes (at least for a class of algorithms) the difficulties that have plagued penalty methods for many years. The new strategy *steers* penalty methods so as to ensure balanced progress toward feasibility and optimality. To select a penalty parameter that achieves this goal, one must solve an additional subproblem at some iterations.

Penalty methods have undergone three stages of development since their introduction in the 1950s. They were first seen as vehicles for solving constrained optimization problems by means of unconstrained optimization techniques. This approach has not proved to be effective, except for special classes of applications. In the second stage, the penalty problem is replaced by a sequence of linearly constrained subproblems. These formulations, which are related to the sequential quadratic programming approach, are much more effective than the unconstrained approach but they leave open the question of how to choose the penalty parameter. In the most recent stage of development, penalty methods adjust the penalty parameter at every iteration so as to achieve a prescribed level of linear feasibility. The choice of the penalty parameter then ceases to be a heuristic and becomes an integral part of the step computation.

An earlier form of the penalty update strategy discussed in this paper is presented in [6], in the context of a successive linear-quadratic programming (SLQP) algorithm. The goal of this paper is to analyze and generalize this strategy to allow its application in other methods. In sections 2 and 3 we present the classical penalty framework and the unconstrained and linearly constrained formulations. The limitations of existing penalty parameter update strategies are examined in section 4. The new penalty strategy is presented in section 5 in the context of sequential quadratic programming (SQP) methods. Later in that section we consider its application to an SLQP method and discuss some modification of the strategy presented in [6]. In section 6 we discuss the relationship between the new penalty method and other trust region approaches. We change the focus slightly in section 7, where we consider merit functions for line search methods, and in particular, how to choose the penalty parameter in this

context.

2 Classical Penalty Framework

We are concerned with the solution of nonlinear programming problems of the form

$$\underset{x}{\text{minimize}} \quad f(x) \tag{2.1a}$$

$$\text{subject to} \quad h_i(x) = 0, \quad i \in \mathcal{E}, \tag{2.1b}$$

$$g_i(x) \geq 0, \quad i \in \mathcal{I}, \tag{2.1c}$$

where the functions $f, h_i, g_i : I\!\!R^n \to I\!\!R$ are assumed to be twice continuously differentiable. We can rephrase (2.1) as the unconstrained minimization of an exact penalty function. In this paper we are interested only in nonsmooth exact penalty functions as typified by the ℓ_1 penalty function

$$\phi_\nu(x) = f(x) + \nu \sum_{i \in \mathcal{E}} |h_i(x)| + \nu \sum_{i \in \mathcal{I}} [g_i(x)]^-, \tag{2.2}$$

where

$$[y]^- = \max(0, -y).$$

As is well known, for appropriate values of the penalty parameter ν, stationary points of ϕ_ν are either KKT points of the nonlinear program (2.1) or infeasible stationary points; see e.g. [7]. This property is the most appealing feature of exact penalty methods because *one* choice of ν may be adequate for the entire minimization procedure. Exact penalty methods are therefore less dependent on the penalty parameter than the quadratic penalty method for which a sequence of subproblems with a divergent series of penalty parameters must be solved. Use of such a function was proposed by Zangwill [43] and Pietrzykowski [35] and methods using it were proposed by Conn and Pietrzykowski [12, 13].

An algorithmic framework that forms the basis for many penalty methods proposed in the literature is as follows. We present it here in the context of the ℓ_1 penalty function.

Algorithm 2.1: Classical ℓ_1 Penalty Method

    Given $\nu_0 > 0$, tolerance $\tau > 0$, starting point x_0^s;
    for $k = 0, 1, 2, \ldots$
        Find an approximate minimizer x_k of $\phi_\nu(x)$, starting at x_k^s;
        if $\sum_{i \in \mathcal{E}} |h_i(x_k)| + \sum_{i \in \mathcal{I}} [g_i(x_k)]^- \leq \tau$
          STOP with approximate solution x_k;
        else
          Choose new penalty parameter $\nu_{k+1} > \nu_k$;
          Choose new starting point x_{k+1}^s;
        end (if)
    end (for)

The minimization of the ℓ_1 penalty function $\phi_\nu(x)$ is difficult because it is non-smooth. We cannot apply algorithms for smooth unconstrained minimization, and general techniques for nondifferentiable optimization, such as bundle methods [27], are not efficient in this context, as they do not take account of the special nature of the nondifferentiabilities. As a result of these obstacles, this unconstrained approach does not translate easily into a general purpose technique for nonlinear programming. On the other hand, it is well understood how to compute minimization steps using a suitable model of $\phi_\nu(x)$, as we discuss next.

3   Linearly Constrained Reformulation

A breakthrough in penalty methods (Fletcher [19]) was the introduction of algorithms that compute steps d based on a piecewise linear-quadratic model of ϕ_ν, in a way that resembles SQP methods. The model is given by

$$
\begin{aligned}
q_\nu(d) &= f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T W d + \nu \sum_{i \in \mathcal{E}} |h_i(x) + \nabla h_i(x)^T d| + \\
&\quad \nu \sum_{i \in \mathcal{I}} [g_i(x) + \nabla g_i(x)^T d]^-,
\end{aligned}
\tag{3.3}
$$

where W is a symmetric matrix approximating the Hessian of the Lagrangian of the nonlinear problem (2.1). The model $q_\nu(d)$ is not smooth, but we can formulate the problem of minimizing it as a smooth quadratic programming problem by introducing

artificial variables r_i, s_i, and t_i, as follows:

$$\underset{d,r,s,t}{\text{minimize}} \quad f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T W d + \nu \sum_{i \in \mathcal{E}} (r_i + s_i) + \nu \sum_{i \in \mathcal{I}} t_i \quad (3.4\text{a})$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = r_i - s_i, \quad i \in \mathcal{E}, \quad (3.4\text{b})$$

$$g_i(x) + \nabla g_i(x)^T d \geq -t_i, \quad i \in \mathcal{I}, \quad (3.4\text{c})$$

$$r, \; s, \; t \geq 0. \quad (3.4\text{d})$$

If a trust region constraint of the form $\|d\|_\infty \leq \Delta$ is added, (3.4) is still a quadratic program. We can solve (3.4) using a standard quadratic programming algorithm.

An important advantage of this approach is that by imposing (approximate) linearizations of the constraints, the step d is often able to make balanced progress toward feasibility and optimality. In fact, when the artificial variables r, s, t are small, (3.4) is closely related to the sequential quadratic programming (SQP) method which is known to be very effective in practice. The step of a classical SQP method is given by

$$\underset{d}{\text{minimize}} \quad f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T W d \quad (3.5\text{a})$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = 0, \quad i \in \mathcal{E}, \quad (3.5\text{b})$$

$$g_i(x) + \nabla g_i(x)^T d \geq 0, \quad i \in \mathcal{I}. \quad (3.5\text{c})$$

As is well known, however, the constraints $(3.5b)$-$(3.5c)$ can be inconsistent; in contrast problem (3.4) is always feasible. Thus the penalty approach that computes steps by (3.4) can be seen as a regularized SQP method in which the constraints have been relaxed.

Algorithms based on the formulation (3.4), such as the $S\ell_1 QP$ method of Fletcher [19], have been shown to possess favorable global convergence properties. They were implemented in the 1980s and early 1990s, and although they appear to have performed well on some tests, they were never incorporated into production-quality software. We conjecture that this was mainly due to the difficulties of choosing the penalty parameter. Without a reliable update procedure for ν, it is not possible to obtain uniform robustness and efficiency over a range of problems.

4 The Crucial Role of the Penalty Parameter

The strategy for choosing and updating the penalty parameter ν_k is crucial to the practical success of the classical ℓ_1 penalty method of section 2. If the initial choice ν_0 is too small, many cycles of the general framework outlined in section 2 may be needed to determine an appropriate value. In addition, the iterates may move away from the solution in these initial cycles, in which case the minimization of $\phi_{\nu_k}(x)$ should be
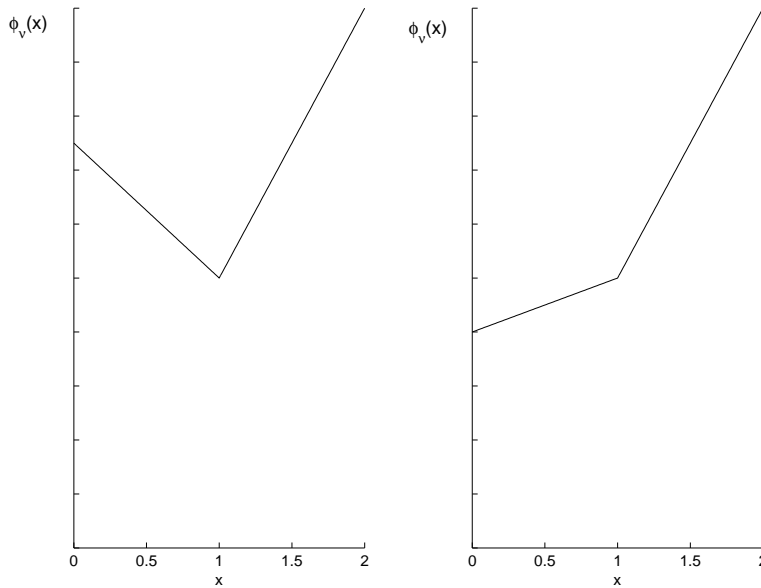
Figure 1: Penalty function for problem (4.6) with $\nu > 1$ (left) and $\nu < 1$ (right).

terminated early, and x_k^s should possibly be reset to a previous iterate. If, on the other hand, ν_k is excessively large, the penalty function may be difficult to minimize as emphasizing constraint feasibility too much may lead to small steps (or the rejection of good steps), possibly requiring a large number of iterations. The difficulties caused by an inappropriate value of ν are illustrated in the following examples.

Example 1. Consider the problem:

$$\min x \qquad \text{subject to} \quad x \geq 1, \qquad (4.6)$$

whose solution is $x_* = 1$. We have that

$$\phi_\nu(x) = \begin{cases} (1 - \nu)x + \nu & \text{if } x < 1 \\ x & \text{if } x \geq 1. \end{cases} \qquad (4.7)$$

The penalty function has a minimizer at $x_* = 1$ when $\nu > 1$, but is a monotonically increasing function when $\nu < 1$; see Figure 1. If, for example, the current iterate is $x_k = 1/2$ and $\nu < 1$, then almost any implementation of a penalty method will give a step that moves away from the solution. This behavior will be repeated, producing increasingly poorer iterates, until the penalty parameter is increased above the threshold value of 1. □

To attempt to overcome these difficulties, several strategies have been proposed to update ν_k at every iteration instead of waiting for the approximate minimization

of $\phi_\nu(x)$ to be completed. One strategy is to set ν_k to be modestly larger than $\|\lambda_k\|_\infty$ where λ_k is the current Lagrange multiplier estimate. This choice is based on the theory of penalty functions [26] which states that in a neighborhood of a solution x_* satisfying certain regularity assumptions, ν_k can be set to be any value larger than $\|\lambda_*\|_\infty$, where λ_* is a vector of optimal Lagrange multipliers for (2.1). Computational experience has shown, however, that this strategy is not generally successful because it may produce an overly large penalty parameter (if an intermediate Lagrange multiplier estimate λ_k becomes excessively large) or an underestimate that may lead to the difficulties just described.

Even if we knew an appropriate value of ν_* at the solution x_*, this value may be inadequate at the starting point or at iterates away from the solution. The following example shows that it is not possible to prescribe in advance a value of the penalty parameter that is adequate at every iteration.

Example 2. Consider the problem :

$$\min\ x^3 \qquad \text{subject to}\quad x \geq -1. \tag{4.8}$$

The corresponding ℓ_1 penalty function is:

$$\phi_\nu(x) = x^3 + \nu \max(0, -x - 1).$$

The solution $x_* = -1$ of (4.8) is a local minimizer of ϕ_ν provided that $\nu > \lambda_* = 3$. However, ϕ_ν is unbounded below as $x \to -\infty$, and for any value of ν, there is a starting point x_0, such that there is no decreasing path in ϕ_ν from x_0 to x_*; see Figure 2. For such a starting point, say $x_0 = -2$ in Figure 2, a local optimization algorithm cannot be expected to find x_* by minimizing ϕ_ν. □

In spite of this, one could consider the heuristic strategy of setting the penalty parameter to a very large value (say 10^{10}) and keeping it fixed throughout the optimization process. The hope is that such a value may be adequate for most problems (and starting points) at hand. As we discuss in the next section, this is not to be recommended because excessively large penalty parameters can lead to inefficient behavior, damaging roundoff errors, and failures.

The difficulties of choosing appropriate values of ν_k in penalty methods [18] caused nonsmooth penalty methods to fall out of favor during the early 1990s and stimulated the development of filter methods which do not require a penalty parameter [21]. The new approach for updating the penalty parameter discussed in section 5 promises, however, to resolve the difficulties mentioned above. By requiring that each step make progress in linear feasibility that is proportional to the optimal possible progress, the new strategy will automatically increase the penalty parameter in the examples above and overcome the undesirable behavior just described.
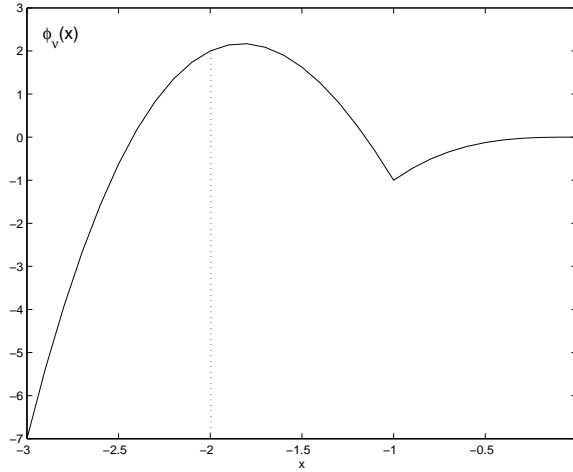
Figure 2: Penalty function for problem (4.8) with $\nu = 10$.

5 A New Penalty Parameter Update Strategy

We now propose a set of guidelines for updating the penalty parameter that can be implemented in a variety of exact penalty methods. We present them first in the context of a sequential quadratic programming method, and apply them later to a sequential linear-quadratic programming method.

We noted in section 3 that a penalty-SQP method computes steps by solving problem (3.4). In a trust region variant of this approach, the subproblem takes the form [18, 19]

$$\underset{d,r,s,t}{\text{minimize}} \quad f(x) + \nabla f(x)^T d + \frac{1}{2} d^T W d + \nu \sum_{i \in \mathcal{E}} (r_i + s_i) + \nu \sum_{i \in \mathcal{I}} t_i \quad (5.9\text{a})$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = r_i - s_i, \quad i \in \mathcal{E}, \quad (5.9\text{b})$$

$$g_i(x) + \nabla g_i(x)^T d \geq -t_i, \quad i \in \mathcal{I}, \quad (5.9\text{c})$$

$$\|d\|_\infty \leq \Delta, \quad (5.9\text{d})$$

$$r, \ s, \ t \geq 0, \quad (5.9\text{e})$$

where Δ is a trust region radius. The problem (5.9) is always feasible but the choice of the penalty parameter ν influences the quality of the step. Rather than employing heuristics, our strategy considers the optimal improvement in linear feasibility achievable inside the trust region. This will help us determine the amount of feasibility improvement that is reasonable to expect from (5.9). Thus we consider the auxiliary subproblem

$$\underset{d,r,s,t}{\text{minimize}} \quad \sum_{i \in \mathcal{E}} (r_i + s_i) + \sum_{i \in \mathcal{I}} t_i \quad (5.10\text{a})$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = r_i - s_i, \quad i \in \mathcal{E}, \tag{5.10b}$$

$$g_i(x) + \nabla g_i(x)^T d \geq -t_i, \quad i \in \mathcal{I}, \tag{5.10c}$$

$$\|d\|_\infty \leq \Delta, \tag{5.10d}$$

$$r, \ s, \ t \geq 0, \tag{5.10e}$$

which can be viewed as a specialization of (5.9) with $\nu = \infty$. Note that (5.10) is a linear program (LP).

The proposed guidelines for updating the penalty parameter are as follows.

Updating Guidelines

1. If there is a step d that lies inside the trust region (5.10d) and satisfies the linearized constraints

$$h_i(x) + \nabla h_i(x)^T d = 0, \quad i \in \mathcal{E}, \tag{5.11a}$$

$$g_i(x) + \nabla g_i(x)^T d \geq 0, \quad i \in \mathcal{I}, \tag{5.11b}$$

   we should compute such a step. In other words, the penalty parameter strategy should choose ν large enough in this case such that all artificial variables r, s, t in (5.9) are zero.

2. If there is no step inside the trust region (5.10d) that satisfies the constraints (5.11a)-(5.11b), choose ν so that the reduction in the infeasibility of the constraints (5.11a)-(5.11b) is proportional to the best possible reduction, which is defined as the reduction obtained with $\nu = \infty$ (i.e. by solving (5.10)).

3. In addition to the two previous requirements on the step d, we want the change in the penalty function to be a good measure of progress made by d. Thus when the step yields a large reduction in the linear model

$$m_x(d) = \sum_{i \in \mathcal{E}} |h_i(x) + \nabla h_i(x)^T d| + \sum_{i \in \mathcal{I}} [g_i(x) + \nabla g_i(x)^T d]^- \tag{5.12}$$

   of the constraints, ν should be chosen large enough such that the reduction in the quadratic model (3.3) of the penalty function is large also.

A more precise description of these guidelines is as follows. Let $d(\nu)$ denote the solution of the quadratic program (5.9) for a given value ν, and denote by $d(\nu_\infty)$ the solution of (5.10). (We can also view $d(\nu_\infty)$ as a minimizer of $m_x(d)$ subject to the trust region constraint (5.10d).)

The first guideline asks that if $m_x(d(\nu_\infty)) = 0$, then the penalty parameter ν_+ chosen at the current iteration should be large enough such that $m_x(d(\nu_+)) = 0$. The

second requirement stipulates that, when the linearized constraints are not feasible, ν_+ should ensure that

$$m_x(0) - m_x(d(\nu_+)) \geq \epsilon_1[m_x(0) - m_x(d(\nu_\infty))], \qquad (5.13)$$

where $\epsilon_1 \in (0,1)$ is a prescribed parameter.

Beyond these two conditions, the third condition requires that the penalty function give sufficient recognition to improvement in feasibility in order to promote acceptance of a good step. Suppose the step $d(\nu_+)$ that minimizes q_{ν_+} makes good progress on feasibility, so that the first two conditions are satisfied, but decreases q_{ν_+} by only a small amount. Then, nonlinearities in the objective f and constraints h, g, could overwhelm this small improvement, and the step $d(\nu_+)$ could cause the actual value of the penalty function ϕ_{ν_+} to increase, and thus be rejected. To prevent this undesirable behavior, after computing a trial value ν_+ of the penalty parameter so that the first two requirements are satisfied, we increase it further if necessary to ensure that

$$q_{\nu_+}(0) - q_{\nu_+}((d(\nu_+))) \geq \epsilon_2\nu_+[m_x(0) - m_x(d(\nu_+))], \qquad (5.14)$$

where $\epsilon_2 \in (0,1)$. We note that a condition similar to (5.14) has been used for the merit function in a variety of trust region methods [8, 14, 16, 29, 34] that compute steps d without reference to a penalty function. Condition (5.14) has some resemblance to a condition proposed by Yuan [42] for adjusting the penalty parameter in an exact penalty method. However, Yuan's condition compares the actual decrease in the penalty function to infeasibility, and he uses the condition to update the penalty parameter for use in the next step.

We summarize the discussion by providing a concrete strategy that implements the guidelines described above.

Algorithm 5.1: Penalty Parameter Update Strategy

Initial data: $(x_k, \nu_{k-1}, \Delta_k)$ and the parameters $\epsilon_1, \epsilon_2 \in (0,1)$.

1. Solve quadratic program (5.9) with $(x_k, \nu_{k-1}, \Delta_k)$ to get $d(\nu_{k-1})$. If $d(\nu_{k-1})$ is linearly feasible (i.e., $m_{x_k}(d(\nu_{k-1})) = 0$), set $\nu_+ \leftarrow \nu_{k-1}$ and proceed to step 4.

2. Solve the linear program (5.10) to get $d(\nu_\infty)$. If $d(\nu_\infty)$ is linearly feasible, choose some $\nu_+ > \nu_{k-1}$ such that $m_{x_k}(d(\nu_+)) = 0$ and proceed to step 4.

3. Choose some $\nu_+ > \nu_{k-1}$ such that the solution $d(\nu_+)$ of (5.9) satisfies (5.13).

4. If ν_+ satisfies (5.14), set $\nu_k \leftarrow \nu_+$; else choose, $\nu_k > \nu_+$ such that $d(\nu_k)$ satisfies (5.14).

A variant of this algorithm would treat the linear feasible and infeasible cases equally. Instead of insisting that $m_x(d(\nu_+)) = 0$ whenever $d(\nu_\infty)$ is linearly feasible, one can accept any trial values of ν that satisfy (5.13). We expect this approach to be effective in practice, but to ensure a fast rate of convergence, the parameter ϵ_1 must vary, and as the iterates approach a solution, it must converge to 1.

Another variant consists of not recomputing the vector $d(\nu_k)$ in step 4 of Algorithm 5.1. Instead, we can keep the vector $d(\nu_+)$ from the previous steps of Algorithm 5.1 (but during the same iteration) and increase ν as necessary to satisfy condition (5.14).

Algorithm 5.1 may call for additional solves of the linear and quadratic programs given by (5.10) and (5.9) to determine an appropriate value of ν. These extra solves are potentially expensive, but the hope is that the additional expense is offset by a reduction in the number of iterations of the SQP method. We have not yet developed a software implementation of the penalty SQP method just outlined, and therefore, cannot evaluate the computational tradeoffs of the penalty update strategy. However, we have produced a software implementation for the SLQP method discussed next, and the results are highly encouraging.

5.1 Application to a Sequential Linear-Quadratic Programming Method

SLQP methods [6, 11, 22] compute a step in two stages. First, a linear program (LP) is solved to identify a working set $\mathcal{W}$. Then an equality constrained quadratic program (EQP) is solved in which the constraints in the working set $\mathcal{W}$ are imposed as equalities, while the remaining constraints are temporarily ignored.

The need for a penalty function arises in the linear programming phase which has the form

$$\underset{d}{\text{minimize}} \quad f(x) + \nabla f(x)^T d \tag{5.15a}$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = 0, \quad i \in \mathcal{E} \tag{5.15b}$$

$$g_i(x) + \nabla g_i(x)^T d \geq 0, \quad i \in \mathcal{I} \tag{5.15c}$$

$$\|d\|_\infty \leq \Delta. \tag{5.15d}$$

A trust region constraint is necessary in (5.15) to ensure that the LP is not unbounded; it also encourages that only locally active constraints are added to the working set. As before, we can ensure the feasibility of the constraints (5.15b)-(5.15d) by following an ℓ_1 penalty approach. Thus we reformulate the LP phase as

$$\underset{d,r,s,t}{\text{minimize}} \quad f(x) + \nabla f(x)^T d + \nu \sum_{i \in \mathcal{E}} (r_i + s_i) + \nu \sum_{i \in \mathcal{I}} t_i \tag{5.16a}$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = r_i - s_i, \quad i \in \mathcal{E} \tag{5.16b}$$

$$g_i(x) + \nabla g_i(x)^T d \geq -t_i, \quad i \in \mathcal{I} \tag{5.16c}$$

$$\|d\|_\infty \leq \Delta \tag{5.16d}$$

$$r, \ s, \ t \geq 0. \tag{5.16e}$$

We denote the solution of this problem as $d^{\text{LP}}(\nu)$. The penalty update guidelines will be applied to choose a suitable penalty parameter value for problem (5.16). The constraints active at the solution of this LP are used to determine a working set $\mathcal{W}$ of equality constraints, subject to which a quadratic model of the penalty function is minimized. The total step of the algorithm is a combination of the steps obtained in the linear programming and equality constrained phases; see [6, 22].

The subproblem (5.16) is identical to (5.9) except for the absence of the quadratic term $\frac{1}{2}d^T W d$ in (5.9a), but this does not alter in any way the goals of our updating strategy. Instead of the quadratic model $q_\nu(d)$ given by (3.3), we now consider the following piecewise linear model of ϕ_ν:

$$\ell_\nu(d) = f(x) + \nabla f(x)^T d + \nu \sum_{i \in \mathcal{E}} |h_i(x) + \nabla h_i(x)^T d| + \nu \sum_{i \in \mathcal{I}} [g_i(x) + \nabla g_i(x)^T d]^-. \tag{5.17}$$

Algorithm 5.1 is applicable in the SLQP context, with the following changes. We denote by $d^{\text{LP}}(\nu_\infty)$ the solution of (5.10). Throughout Algorithm 5.1, we make the replacement $d \leftarrow d^{\text{LP}}$, and condition (5.14) takes the form

$$\ell_{\nu_+}(0) - \ell_{\nu_+}(d^{\text{LP}}(\nu_+)) \geq \epsilon_2 \nu_+ [m_x(0) - m_x(d^{\text{LP}}(\nu_+))]. \qquad (5.18)$$

The strategy given in Algorithm 5.1 has been implemented in the SLQP method that forms part of the KNITRO 4.0 software package [40]; it is available under the option KNITRO-Active. It initializes $\nu_0 = 10$ at the beginning of the algorithm, and sets $\epsilon_1 = 0.1$ in (5.13) and $\epsilon_2 = 0.5$ in (5.14). The implementation of Steps 2, 3 and 4 is achieved by increasing ν_{k-1} by a factor of 10 and re-solving (5.16) until the desired condition is satisfied. (An earlier version of this penalty update strategy is described in [6].)

Numerical tests with KNITRO-Active indicate that Algorithm 5.1 is substantially more efficient and robust than classical strategies that use a more static penalty update strategy. Although Algorithm 5.1 calls for potentially many extra linear programs to be solved in selecting the penalty parameter, the cost of these extra LPs has proved to be fairly small in practice for two reasons. Firstly, we have noticed that the strategy described in Algorithm 5.1 typically finds an adequate value of the penalty parameter quickly, after which it stabilizes resulting in a small number of extra LPs and a reduction in the overall number of outer iterations compared with other heuristic techniques for modifying the penalty parameter. Secondly, when using a simplex method, the extra LPs with varying values of ν which need to be solved are typically solved very quickly using warm starts. In our tests on a large set of test problems, we found that the number of simplex iterations used in solving the extra LPs was less than three percent of the total number of simplex iterations used in computing the steps d actually taken.

A global convergence analysis of a penalty SLQP method is given in [7]. In that study, condition (5.14) is replaced by the condition

$$\ell_{\nu_+}(0) - \ell_{\nu_+}((d^{\text{LP}}(\nu_+)) \geq \epsilon_2 \nu_+ [m_x(0) - m_x(d^{\text{LP}}(\nu_\infty))],$$

which is less restrictive than (5.14) because $m_x(d^{\text{LP}}(\nu)) \geq m_x(d^{\text{LP}}(\nu_\infty))$. Therefore the analysis in [7] is of broad applicability.

5.2   Behavior on Three Examples

Let us examine the behavior of the penalty update strategy of Algorithm 5.1 on the two examples given in section 4 and on a linear programming problem studied by Fletcher [20].
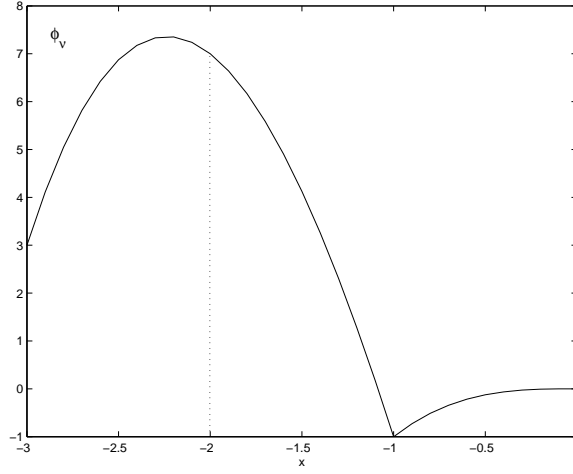
Example 1, Revisited.

Figure 3: Penalty function for problem (4.8) after adjustment of penalty parameter.

Suppose that $x_k = 1/2$, $\nu_{k-1} = 0.1$ and that $W = 0$ in (5.9). Let us denote the current trust region radius by Δ_k. For this value of ν, the artificial variable t is not zero and the step generated by (5.9) would move away from the solution. For $\nu = \infty$ we see that if $\Delta_k \geq \frac{1}{2}$ then the artificial variable t will be set to zero and the step d will be feasible. Algorithm 5.1 would then increase ν until t is driven to zero, that is, it will choose $\nu_k \geq 1$. On the other hand, if $\Delta_k < \frac{1}{2}$ then by setting $\nu = \infty$ the artificial variable t is not driven to zero, but takes the value $\frac{1}{2} - \Delta_k$. Algorithm 5.1 will choose ν_k to satisfy (5.13), which in this case requires $\nu > 1$ for any $\epsilon_1 \in (0,1)$. In either case the algorithm will perform as desired.

Example 2, Revisited.

For simplicity, we assume again that $W = 0$ in (5.9). At an iterate $x_k < -1$ we have that $m_k(d) = \max(-x_k - 1 - d, 0)$, and $d(\nu_\infty) = \min(\Delta_k, -1 - x_k)$. The linearized model of the penalty function is given by $\ell_\nu(d) = 3x_k^2 d + \nu \max(-x_k - 1 - d, 0)$. If we want $d(\nu)$ to make any progress on feasibility, we must have $d(\nu) > 0$, and this can only occur if $\nu > 3x_k^2$. If ν_+ is chosen that large, then $d(\nu_+) = \min(\Delta_k, -1 - x_k) = d(\nu_\infty)$, and (5.13) is satisfied. If we also impose (5.14), then we must have $\nu > 3x_k^2/(1 - \epsilon_2)$. Now if ν_+ is chosen to have such a large value, then it is still the case that $\phi_{\nu_+}(x)$ is unbounded below but $\phi'_{\nu_+}(x) < 0$ on the interval $[x_k, -1]$; see Figure 3 where we use the value $\nu = 15$, which is sufficiently large when $x_k = -2$. By increasing ν we have enlarged the basin of attraction of the solution to include the current iterate; thus a minimization method on ϕ_ν should be expected to move toward the solution $x_* = -1$, indicating that ϕ_ν is a useful penalty function.

Example 3: ADLITTLE

The problem `ADLITTLE` from the Netlib LP collection [23] was used by Fletcher [20] to illustrate the difficulties of designing a robust penalty method. For the ℓ_1 penalty function to have a minimizer at the solution of this linear program, ν must be at least as large as roughly 3.31×10^3.

Consider a penalty method that computes a step by minimizing (3.3), i.e. by solving (3.4). We want to keep the penalty parameter ν as small as possible to avoid ill-conditioning and the desired threshold value of 3.31×10^3 is unknown to the algorithm at the outset. We expect that if we choose a value of ν less than the threshold, then the minimization of the penalty function will terminate without driving the constraints to zero and we will be alerted to increase the penalty parameter. However, Fletcher points out that if ν is slightly less than 3.31×10^3, the function (3.3) becomes unbounded. Therefore a penalty method which does not dynamically update the penalty parameter based on progress in linear feasibility will generate a sequence of iterates that moves away from the feasible region. (We confirmed this behavior experimentally.)

We applied the exact penalty method implemented in KNITRO-Active which uses Algorithm 5.1 to select the penalty parameter. Choosing $\nu_0 = 10$, setting the initial trust region radius to a very large number ($\Delta_0 = 10^{10}$), and turning off the scaling feature of KNITRO, we observed that the algorithm immediately increases the penalty parameter to 10^4 which is sufficient to achieve feasibility, and solves the problem in one iteration. This is expected, because for a large initial trust region, the constraints (5.15b)-(5.15d) are consistent and Algorithm 5.1 forces the penalty parameter large enough to achieve feasibility such that the solution of (5.10) corresponds to the solution of the original LP defined by `ADLITTLE`.

We repeated the experiment with $\Delta_0 = 10$. This initial Δ_0 is small enough such that the algorithm cannot achieve linear feasibility in one step. Although `ADLITTLE` is a linear program, this information is not given to KNITRO-Active (which assumes that the problem is a general nonlinear program). As a result, it solves a sequence of penalty problems of the form (5.10), adjusting ν and Δ as it progresses. The problem was solved in 6 iterations, with a final value of the penalty parameter (achieved at iteration 3) of 10^4. Therefore the algorithm behaved as expected, both with large and small initial trust regions. $\qquad\square$

We conclude this section with some remarks about alternative strategies for updating the penalty parameter. One simple strategy is to choose a very large and fixed value of ν for all problems. The hope is that if a scale-invariant Newton-type method is used to generate steps, the ill-conditioning introduced by a large penalty parameter may not be detrimental.

We experimented with running KNITRO-Active using an initial penalty parameter value of $\nu = 10^5$ and observed a degradation of performance. The number of problems (from a subset of the CUTEr test set) solved successfully went from 485

out of 616 to 449. When we tested an initial value of $\nu = 10^{10}$, the number of problems successfully solved dropped to only 321. Many of the failures caused by large values of ν seemed to occur because, near the feasible region, there are often small increases in infeasibility due to nonlinearities in constraints or roundoff error in even linear constraints. Because of the large value of ν, these increases dominated the objective function improvement, and forced the method to take very small steps, and sometimes completely prevented further progress. We conclude that the risks of using excessively large penalty parameters are real indeed and that an adaptive strategy, such as the one described above is valuable in practice.

Other, more sophisticated, penalty update strategies have been proposed recently. Chen and Goldfarb [10] propose rules that update the penalty parameter as optimality of the penalty problem is approached; they are based in part on feasibility and the size of the multipliers. Leyffer et al. [30] consider penalty methods for MPCCs and describe dynamic criteria for updating the penalty parameter based on the average decrease in the penalized constraints. The methods proposed in section 5 differ from these strategies in that they assess the effect of the penalty parameter on the step to be taken, based on the current model of the problem.

6   Balancing Feasibility and Optimality in Other Trust Region Methods

Algorithm 5.1 provides a strategy for balancing progress on optimality and on feasibility in an exact penalty method. In trust region SQP methods that compute steps without regard to a penalty function there is also a need to balance these two goals, and some of the proposed methods for doing this are analogous to the procedure described in the previous section.

Consider a problem with equality constraints only. Trust region SQP methods aim to compute a step d by solving the subproblem

$$\underset{d}{\text{minimize}} \quad f(x) + \nabla f(x)^T d + \tfrac{1}{2}d^T W d \qquad (6.19a)$$

$$\text{subject to} \quad h_i(x) + \nabla h_i(x)^T d = 0, \quad i \in \mathcal{E}, \qquad (6.19b)$$

$$\|d\|_\infty \leq \Delta. \qquad (6.19c)$$

One way to ensure that the constraints (6.19b)-(6.19c) are always compatible, is to relax the linear constraints (6.19b). The question of how much to relax them is delicate and is analogous to the question of what value to give a penalty parameter. A number of proposed methods adopt strategies that, like Algorithm 5.1, proceed by working on minimizing a measure of feasibility alone. Both Byrd and Omojokun

[5, 34], and Powell and Yuan [36] solve a problem of the form

$$\text{minimize}_{v} \quad \|h(x) + \nabla h(x)^T v\|_2 \tag{6.20a}$$

$$\text{subject to} \quad \|v\|_2 \leq \beta\Delta, \tag{6.20b}$$

where $\beta \in (0, 1)$ and where h denotes the vector with components $h_i(x)$, $i \in \mathcal{E}$. The resulting step v is usually referred to as the *normal* step. Byrd and Omojokun suggest the value $\beta = 0.8$ in (6.20b) and compute a total step d by solving (6.19) with (6.19b) replaced by the condition

$$h_i(x) + \nabla h_i(x)^T d = h_i(x) + \nabla h_i(x)^T v, \quad i \in \mathcal{E}.$$

This is very similar to solving the subproblem with $\nu = \infty$ in step 2 of Algorithm 5.1. Thus the normal step v, and hence the total step d, provides *exactly* the best possible decrease in the linearized constraints within the (shortened) trust region (6.20b).

Powell and Yuan replace (6.19b) with

$$\|h(x) + \nabla h(x)^T v\|_2 \leq \zeta_k, \tag{6.21}$$

where ζ_k is the optimal value of (6.20a) for some value of $\beta \in (0, 1)$. This has the flavor of condition (5.13). Celis, Dennis and Tapia [9] also impose a constraint of the form (6.21), but define ζ as the value $\|h(x) + \nabla h(x)^T d_\mathrm{C}\|_2$, where d_C is a Cauchy step for problem (6.20). In this respect it is different from the approaches considered earlier in the paper which aim for at least a fraction of the optimal linear decrease, whereas Celis, Dennis and Tapia are satisfied with a more modest decrease. Additionally, Burke [4] describes a trust region method that is similar to these.

The approaches of Powell and Yuan and of Celis, Dennis and Tapia allow more flexibility in reducing the objective function, while the Byrd-Omojokun approach involves a subproblem that is easier to solve. (The Byrd-Omojokun technique is at the core of the interior-point option KNITRO-Interior/CG.) All of these approaches share the property of requiring somewhat less than the level of linearized feasibility that is attainable within the current trust region Δ_k. Thus the β of (6.20b), the single steepest descent step of Celis, Dennis and Tapia, and the ϵ_1 of condition (5.13) used in Algorithm 5.1 all play similar roles.

7   Penalty Parameters in Merit Functions

The main goal of this paper has been to describe guidelines for dynamically updating the penalty parameter in penalty methods for constrained optimization. However, penalty parameter update strategies are also important in methods that use a nonsmooth penalty function only as a *merit function*, i.e., to judge the acceptability of the

step but not to determine the search direction. As noted previously, a condition like (5.14) is also known to be effective for merit functions used with trust-region methods. In this section, we point out that this condition is also relevant for choosing the penalty parameter value in merit functions with line search methods.

Let us begin by considering trust region methods that use a nondifferentiable function like (2.2) as a merit function. After a step d is computed, these methods adjust the penalty parameter so that the merit function is compatible with the step. We have argued in the paragraph that contains condition (5.14) that, for penalty methods, it is reasonable to select the penalty parameter ν_+ so the decrease in the model that produced the step is proportional to the product of ν and the decrease in the linearized constraints. This requirement was expressed as condition (5.14) for an SQP-based method. The same reasoning is valid for (non penalty) SQP methods of the type described in the previous section. Therefore, the condition (5.14) is still appropriate, and we restate it here:

$$ q_\nu(0) - q_\nu(d(\nu)) \geq \epsilon_2 \nu [m_x(0) - m_x(d(\nu))]. \tag{7.22} $$

Several trust region algorithms [8, 14, 16, 29, 34] select the penalty parameter ν_+ at every iteration so that a condition like (7.22) is satisfied. (Some of these methods omit the factor ν from the right side of (7.22). However, our numerical experience indicates that when ν is large, step acceptance is more likely when ν is included.) This strategy has proved to be effective in the trust region algorithm implemented in KNITRO-Interior/CG.

Line search methods, however, have generally had less success using nonsmooth merit functions. Several authors, most recently Wächter [38], report that ℓ_1 and ℓ_2 merit functions interfere unduly with good Newton steps, even if a mechanism to overcome the Maratos effect is employed. The perception is that it is difficult to find rules for selecting the penalty parameter that are effective over a wide range of problems. As a result, nonsmooth merit functions have often been discarded in favor of smooth merit functions, such as augmented Lagrangians, or filters [15, 21, 24, 32, 39].

It should be noted that historically line search methods have not enforced a condition like (7.22), but rather have typically required that the penalty parameter be chosen to ensure the computed step is a descent direction for the merit function. Recently, however, Waltz et. al. [41] have reported good computational results for a line search interior-point method that uses a nonsmooth merit function in which the penalty parameter is selected by condition (7.22). Their strategy was motivated by computational efficiency; we now analyze it in the context of the updating guidelines of section 5. For simplicity, we restrict our attention to equality constrained optimization.

Let us consider a method that computes the search direction d by solving the

Newton-KKT system

$$\begin{bmatrix} W & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix} \begin{bmatrix} d \\ d_\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ h(x) \end{bmatrix}, \tag{7.23}$$

where $\mathcal{L}$ is the Lagrangian of the nonlinear program (2.1), and W is an approximation to $\nabla_{xx}^2 \mathcal{L}$ that is positive definite on the null space of the Jacobian matrix, $\nabla h(x)^T$. The method then performs a line search to find a steplength $\alpha > 0$ such that $\phi_\nu(x+\alpha d)$ is sufficiently smaller than $\phi_\nu(x)$, where

$$\phi_\nu(x) = f(x) + \nu \|h(x)\|,$$

and $\| \cdot \|$ is a vector norm.

It follows from (7.23) that the step d satisfies

$$m(d) = \|h(x) + \nabla h(x)^T d\| = 0, \tag{7.24}$$

i.e., d has achieved the best possible decrease in linear feasibility. Therefore, the updating guidelines of section 5 would simply require that inequality (7.22) be satisfied for some appropriate model q_ν. This model will be chosen differently than in the trust region case. Following El Hallabi [17], Waltz et al. [41] define

$$q_\nu(d) = f(x) + \nabla f(x)^T d + \frac{\sigma}{2} d^T W d + \nu \|h(x) + \nabla h(x)^T d\|, \tag{7.25}$$

with

$$\sigma = \begin{cases} 1 & \text{if } d^T W d > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7.26}$$

Note that (7.25) differs from (6.19) only in the parameter σ. The new penalty parameter ν_+ is required to satisfy condition (7.22), which by (7.24) can be written as

$$q_\nu(0) - q_\nu(d) \geq \epsilon_2 \nu \|h(x)\| \tag{7.27}$$

(we removed the dependence of d on ν). It follows from (7.25) that condition (7.27) implies

$$\nu \geq \frac{\nabla f(x)^T d + \frac{\sigma}{2} d^T W d}{(1 - \epsilon_2) \|h(x)\|} \equiv \nu_{\text{TRIAL}}. \tag{7.28}$$

The update rule for the penalty parameter is as follows. If ν denotes the penalty parameter from the previous iteration, we define the new parameter by

$$\nu_+ = \begin{cases} \nu & \text{if } \nu \geq \nu_{\text{TRIAL}} \\ \nu_{\text{TRIAL}} + 1 & \text{otherwise.} \end{cases} \tag{7.29}$$

The role of the term σ in (7.25) is to ensure that the direction computed by (7.23) is a descent direction for ϕ_{ν_+}. It is not difficult to show (see e.g. [33, p. 545]) that the directional derivative of ϕ_ν in the direction d is given by

$$D\phi_\nu(x;d) = \nabla f(x)^T d - \nu\|h(x)\|. \tag{7.30}$$

If ν_+ satisfies (7.28) we have

$$D\phi_{\nu_+}(x;d) \leq -\epsilon_2\nu_+\|h(x)\|, \tag{7.31}$$

and therefore d is a descent direction for the merit function ϕ_{ν_+}. This inequality is, however, not always valid if $\sigma = 1$ and $d^T W d < 0$.

The numerical tests reported in [39, 41] indicate that the strategy (7.28)-(7.29) is effective in practice. Wächter [39] made a controlled comparison with a filter line search method and found that the merit function strategy just described appears to be as effective as a filter approach in its ability to accept good steps. Our experience with KNITRO-Interior/Direct also indicates that the strategy (7.28)-(7.29) is significantly more effective than choosing ν simply to ensure that (7.31) holds. The latter condition implies

$$\nu \geq \frac{\nabla f(x)^T d}{(1 - \epsilon_2)\|h(x)\|}. \tag{7.32}$$

By comparing (7.32) and (7.28) we see that, when $\sigma > 0$, the new strategy selects a larger penalty parameter, placing more weight on the reduction of the constraints. As a consequence, if the step d decreases the constraints but increases the objective, it has better chances of being accepted by the merit function.

In summary, the penalty update strategy (7.28)-(7.29) can be justified for (non penalty) line search methods, and has proved to be effective in practice. It is interesting, that this strategy is consistent, and could have been derived, from the updating guidelines proposed in this paper. It remains an open question, however, how to extend our guidelines to penalty line search methods.

8 Final Remarks

The penalty update strategy presented in this paper is significantly different from most of the approaches described in the literature. We take the view that choosing an appropriate value of ν is not a simple task and requires, in some cases, the solution of an auxiliary subproblem. Fortunately, in several algorithmic contexts this subproblem adds little computational cost to the iteration.

References

[1] M. Anitescu. On Solving Mathematical Programs with Complementarity Constraints as Nonlinear Programs. Preprint ANL/MCS-P864-1200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA, 2000.

[2] M. Anitescu. Global convergence of an elastic mode approach for a class of Mathematical Programs with Complementarity Constraints. *SIAM Journal on Optimization*, 16(1):120–145, 2005.

[3] H.Y. Benson, A. Sen, D.F. Shanno, and R.J. Vanderbei. Interior-point algorithms, penalty methods and equilibrium problems. Report ORFE-03-02, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA, 2003.

[4] J. V. Burke. A robust trust region method for constrained nonlinear programming problems. *SIAM Journal on Optimization*, 2(2):324–347, 1992.

[5] R. H. Byrd. Robust trust region methods for constrained optimization. Third SIAM Conference on Optimization, Houston, Texas, May 1987.

[6] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming, Series B*, 100(1):27–48, 2004.

[7] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. On the convergence of successive linear-quadratic programming algorithms. *SIAM Journal on Optimization*, 16(2):471–489, 2006.

[8] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[9] M. R. Celis, J. E. Dennis, and R. A. Tapia. A trust region strategy for nonlinear equality constrained optimization. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, Philadelphia, USA, 1985. SIAM.

[10] L. Chen and D. Goldfarb. Interior-point ℓ_2 penalty methods for nonlinear programming with strong global convergence properties. *Mathematical Programming*, 108(1):1–36, 2006.

[11] C. M. Chin and R. Fletcher. On the global convergence of an SLP-filter algorithm that takes EQP steps. *Mathematical Programming, Series A*, 96(1):161–177, 2003.

[12] A. R. Conn. Constrained optimization via a nondifferentiable penalty function. *SIAM Journal on Numerical Analysis*, 10(4):760–779, 1973.

[13] A. R. Conn and T. Pietrzykowski. A penalty function method converging directly to a constrained optimum. *SIAM Journal on Numerical Analysis*, 14(2):348–375, 1977.

[14] J. E. Dennis, M. El-Alem, and M. C. Maciel. A global convergence theory for general trust-region based algorithms for equality constrained optimization. *SIAM Journal on Optimization*, 7(1):177–207, 1997.

[15] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.

[16] M. El-Alem. A global convergence theory for the Dennis-Celis-Tapia trust-region algorithm for constrained optimization. *SIAM Journal on Numerical Analysis*, 28(1):266–290, 1991.

[17] M. El-Hallabi. A hybrid algorithm for nonlinear equality constrained optimization problems: global and local convergence theory. Technical Report TR4-99, Mathematics and Computer Science Department, Institut National des Postes et Télécommunications, Rabat, Morocco, 1999.

[18] R. Fletcher. An ℓ_1 penalty method for nonlinear constraints. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 26–40, Philadelphia, USA, 1985. SIAM.

[19] R. Fletcher. *Practical Methods of Optimization*. J. Wiley and Sons, Chichester, England, second edition, 1987.

[20] R. Fletcher. Presentation. SIAM Conference on Optimization, Victoria, Canada, May 1996.

[21] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–269, 2002.

[22] R. Fletcher and E. Sainz de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43(3):235–256, 1989.

[23] D.M. Gay. Electronic mail distribution of linear programming test problems. *COAL Newsletter*, 13(10):10–12, 1985.

[24] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[25] N. I. M. Gould, D. Orban, and Ph. Toint. An interior-point L1-penalty method for nonlinear optimization. Technical Report RAL-TR-2003-022, Rutherford Appleton Laboratory Chilton, Oxfordshire, UK, 2003.

[26] S. P. Han and O. L. Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical Programming*, 17(3):251–269, 1979.

[27] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms. Part 1: Fundamentals.* Springer Verlag, Heidelberg, Berlin, New York, 1993.

[28] X.M. Hu and D. Ralph. Convergence of a penalty method for mathematical programming with complementarity constraints. *Journal of Optimization Theory and Applications*, 123(2):365–390, 2004.

[29] M. Lalee, J. Nocedal, and T. D. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, 1998.

[30] S. Leyffer, G. López-Calva, and J. Nocedal. Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 17(1):52–77, 2006.

[31] M. Mongeau and A. Sartenaer. Automatic decrease of the penalty parameter in exact penalty function methods. *European Journal of Operational Research*, 83(3):686–699, 1995.

[32] B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical report, SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1983. Revised 1995.

[33] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, 1999.

[34] E. O. Omojokun. *Trust region algorithms for optimization with nonlinear equality and inequality constraints.* PhD thesis, University of Colorado, Boulder, Colorado, USA, 1989.

[35] T. Pietrzykowski. An exact potential method for constrained maxima. *SIAM Journal on Numerical Analysis*, 6(2):299–304, 1969.

[36] M. J. D. Powell and Y. Yuan. A trust region algorithm for equality constrained optimization. *Mathematical Programming*, 49(2):189–213, 1990.

[37] H. Scheel and S. Scholtes. Mathematical programs with complementarity constraints: Stationarity, optimality and sensitivity. *Mathematics of Operations Research*, 25(1):1–22, 2000.

[38] A. Wächter. *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering.* PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2002.

[39] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[40] R. A. Waltz. Knitro 4.0 User's Manual. Technical report, Ziena Optimization, Inc., Evanston, IL, USA, October 2004.

[41] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming, Series A*, 107:391–408, 2006.

[42] Y. Yuan. On the convergence of a new trust region algorithm. *Numerische Mathematik*, 70(4):515–539, 1995.

[43] W.I. Zangwill. Non-linaer programming via penalty functions. *Management Science*, 13:344–358, 1967.