An Interior Point Algorithm for Large Scale Nonlinear Programming*

Richard H. Byrd[†] Mary E. Hribar [‡] Jorge Nocedal[§]

December 15, 1998

Abstract

The design and implementation of a new algorithm for solving large nonlinear programming problems is described. It follows a barrier approach that employs sequential quadratic programming and trust regions to solve the subproblems occurring in the iteration. Both primal and primal-dual versions of the algorithm are developed, and their performance is illustrated in a set of numerical tests.

Key words: constrained optimization, interior point method, large-scale optimization, nonlinear programming, primal method, primal-dual method, sequential quadratic programming, barrier method, trust region method.

---

*Original manuscript: July 27, 1997.

[†]Computer Science Department, University of Colorado, Boulder CO 80309. This author was supported by ARO grant DAAH04-94-0228, and AFOSR grant F49620-94-1-0101.

[‡]CAAM Department, Rice University, Houston TX 77005. This author was supported by Department of Energy grant DE-FG02-87ER25047-A004.

[§]ECE Department, Northwestern University, Evanston Il 60208. This author was supported by National Science Foundation grant CCR-9625613 and by Department of Energy grant DE-FG02-87ER25047-A004.

1. Introduction

In this paper we discuss the design, implementation and performance of an interior point method for solving the nonlinearly constrained optimization problem

$$\begin{array}{rl} \min & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0, \end{array} \tag{1.1}$$

where $f : \mathbf{R}^n \to \mathbf{R}$, $h : \mathbf{R}^n \to \mathbf{R}^t$, and $g : \mathbf{R}^n \to \mathbf{R}^m$ are smooth functions. We are particularly interested in the case when (1.1) is not a convex program and when the number of variables n is large. We assume in this paper that first and second derivatives of the objective function and constraints are available, but our strategy can be extended so as to make use of quasi-Newton approximations.

Interior point methods provide an alternative to active set methods for the treatment of inequality constraints. Our algorithm, which is based on the framework proposed by Byrd, Gilbert and Nocedal [7], incorporates within the interior point method two powerful tools for solving nonlinear problems: sequential quadratic programming and trust region techniques. Sequential quadratic programming (SQP) ideas are used to efficiently handle nonlinearities in the constraints. Trust region strategies allow the algorithm to treat convex and non-convex problems uniformly, permit the direct use of second derivative information and provide a safeguard in the presence of nearly dependent constraint gradients.

Of crucial importance in the new algorithm is the formulation and solution of the equality constrained barrier subproblems that determine the steps of the algorithm. The formulation of the subproblems gives the iteration primal or primal-dual characteristics, and ensures that the slack variables remain safely positive. The technique used to solve the subproblems has a great impact on the efficiency and robustness of the algorithm; we use an adaptation of the trust region method of Byrd and Omojokun [6, 34] which has proved to be effective for solving large equality constrained problems [31].

Our numerical results suggest that the new algorithm holds much promise: it appears to be robust and efficient (in terms of function evaluations), and can make effective use of second derivative information. The test results also indicate that the primal-dual version of the algorithm is superior to the primal version. The new algorithm has a solid theoretical foundation, since it follows the principles of the globally convergent primal method developed in [7]. In particular, the approximate solution strategies for the subproblems solved by our algorithm are chosen to satisfy the explicit conditions for global convergence stated in that paper.

There has been much research in using interior point methods for nonlinear programming; most of it concerns line search methods. The special case when the problem is a *convex program* can be handled by line search methods that are direct extensions of interior point methods for linear programming (see e.g. [1]). In the convex case, the step generated by the solution of the primal-dual equations can be shown to be a descent direction for several merit functions, and this allows one to establish global convergence results. Other research [19, 44] has focused on the *local behavior* of interior point line search methods for nonlinear programming. Conditions have been given that guarantee superlinear and quadratic rates of convergence. These algorithms can also be viewed as a direct extension of linear programming methods, in that they do not make provisions for the case when the problems is non-convex.

Several line search algorithms designed for *non-convex* problems have recently been proposed [43, 22, 15, 23, 2, 35]. An important feature of many of these methods is a strategy for modifying the KKT system used in the computation of the search direction. This modification, which is usually based on a matrix factorization algorithm, ensures that the search direction is a descent direction for the merit function. This is an approaches are interesting, but there is not yet enough experience to fully evaluate their efficacy in general-purpose codes.

The use of trust region strategies in interior point methods for linear and nonlinear problems is not new [5, 33]. Coleman and Li [13, 14] proposed a primal method for bound constrained nonlinear optimization; see also [18]. Plantenga [36] developed an algorithm for general nonlinear programming that has some features in common with our algorithm; the main differences lie in his treatment of the trust region, in the purely primal nature of his step, and in the fact that his algorithm reverts to an active set method near the solution.

The algorithm proposed in this paper makes use of sequential quadratic programming techniques [3, 21, 24, 25], and in this sense is related to the line search algorithm of Yamashita [43]. But the way in which our algorithm combines trust region strategies, interior point approaches and sequential quadratic programming techniques leads to an iteration that is different from those proposed in the literature.

2. The New Algorithm

The algorithm is a barrier method in which the subproblems are solved approximately by an SQP iteration with trust regions. Each barrier subproblem is of the form

$$\begin{array}{ll} \min\limits_{x,s} & f(x) - \mu \sum\limits_{i=1}^{m} \ln s_i \\ \text{subject to} & h(x) = 0 \\ & g(x) + s = 0, \end{array} \tag{2.1}$$

where $\mu > 0$ is the *barrier parameter* and where the slack variable s is assumed to be positive. By letting μ converge to zero, the sequence of solutions to (2.1) should normally converge to a stationary point of the original nonlinear program (1.1). As in some interior point methods for linear programming [42], our algorithm does not require feasibility of the iterates with respect to the inequality constraints in (1.1), but only forces the slack variables in (2.1) to remain positive.

To characterize the solution of the barrier problem (2.1) we introduce its Lagrangian,

$$\mathcal{L}(x, s, \lambda_h, \lambda_g) = f(x) - \mu \sum_{i=1}^{m} \ln s_i + \lambda_h^T h(x) + \lambda_g^T (g(x) + s), \tag{2.2}$$

where λ_h and λ_g are the Lagrange multipliers. Rather than solving each barrier subproblem (2.1) accurately, we will be content with an approximate solution $(\hat{x}, \hat{s})$ satisfying $E(\hat{x}, \hat{s}; \mu) \leq \epsilon_\mu$, where E measures the optimality conditions of the barrier problem and is defined by

$$\begin{aligned} E(x, s; \mu) \quad = \quad & \max\left(\|\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g\|_\infty, \|S\lambda_g - \mu e\|_\infty, \|h(x)\|_\infty,\right. \\ & \left.\|g(x) + s\|_\infty\right). \end{aligned} \tag{2.3}$$

Here $e = [1, ..., 1]^T$, $S = \text{diag}(s^1, ..., s^m)$, with superscripts indicating components of a vector, and

$$A_h(x) = [\nabla h^1(x), \ldots, \nabla h^t(x)], \qquad A_g(x) = [\nabla g^1(x), \ldots, \nabla g^m(x)]$$

are the matrices of constraint gradients. Throughout the paper we will assume that A_h has full column rank. In the definition of the optimality measure E, the vectors λ_h, λ_g are least squares multiplier estimates (to be discussed later), and thus are functions of x, s and μ. We will show later (see (3.7)-(3.10)) that the terms in (2.3) correspond to each of the equations of the so-called perturbed KKT system upon which our primal-dual algorithm is based. The tolerance ϵ_μ, which determines the accuracy in the solution of the barrier problems, is decreased from one barrier problem to the next, and must converge to zero. In this paper we will use the simple strategy of reducing both ϵ_μ and μ by a constant factor $\theta \in (0, 1)$. We test for optimality for the nonlinear program (1.1) by means of $E(x, s; 0)$.

Algorithm I: Barrier Algorithm for Solving the Nonlinear Problem (1.1)

Choose an initial value for the barrier parameter $\mu > 0$, and select the parameters $\epsilon_\mu > 0$, $\theta \in (0, 1)$, and the final stop tolerance ϵ_{TOL}. Choose the starting point x and $s > 0$, and evaluate the objective function, constraints, and their derivatives at x.

Repeat until $E(x, s; 0) \leq \epsilon_{\text{TOL}}$:

1. Apply an SQP method with trust regions, starting from (x, s), to find an approximate solution (x^+, s^+) of the barrier problem (2.1) satisfying $E(x^+, s^+; \mu) \leq \epsilon_\mu$.
2. Set $\mu \leftarrow \theta\mu$, $\epsilon_\mu \leftarrow \theta\epsilon_\mu$, $x \leftarrow x^+$, $s \leftarrow s^+$.

end

To obtain a rapidly convergent algorithm, it is necessary to carefully control the rate at which the barrier parameter μ and the convergence tolerance ϵ_μ are decreased [19, 44]. This question has been studied, in the context of our algorithm, in [8].

Most of the work of Algorithm I lies clearly in step 1, in the approximate solution of an equality constrained problem with an implicit lower bound on the slack variables. The challenge is to perform this step efficiently, even when μ is small, while forcing the slack variables to remain positive. To do this we apply an adaptation of the equality constrained SQP iteration with trust regions proposed by Byrd [6] and Omojokun [34] and developed by Lalee, Nocedal and Plantenga [31] for large-scale equality constrained optimization. We follow an SQP approach because, in our view, it is effective for solving equality constrained problems, even when the problem is ill-conditioned and the constraints are highly nonlinear (see also [3, 25, 21, 24]), and choose to use trust region strategies to globalize the SQP iteration because they facilitate the use of second derivative information when the problem is non-convex.

However, our numerical experience shows that a straightforward application of this SQP method to the barrier problem leads to inefficient steps that tend to violate the positivity of the slack variables, and that are thus frequently cut short by the trust region constraint. The novelty of our approach lies in the formulation of the quadratic model in the SQP iteration and in the definition of the (scaled) trust region. These are designed so as to produce steps that have some of the properties of primal-dual iterations and that avoid approaching the boundary of the feasible region too soon.

In order to describe our approach more precisely, it is instructive to briefly review the basic principles of Sequential Quadratic Programming for equality constrained optimization with trust regions [3, 9, 10, 31, 40]. Every iteration of such an SQP method begins by constructing a quadratic model of the Lagrangian function. A step d of the algorithm is computed by minimizing the quadratic model, subject to satisfying a linear approximation to the constraints, and subject to a trust region bound on this step. If the step d gives a sufficient reduction in the chosen merit function, then it is accepted; otherwise the step is rejected, the trust region is reduced and a new step is computed.

Let us apply these ideas to the barrier problem (2.1), in order to compute a step $d = (d_x, d_s)$ from the current iterate (x_k, s_k). To economize space we will often write vectors with x and s-components as

$$\begin{pmatrix} d_x \\ d_s \end{pmatrix} = (d_x, d_s).$$

After computing Lagrange multiplier estimates (λ_h, λ_g), we formulate the subproblem

$$\min_{d_x, d_s} \; \nabla f(x_k)^T d_x + \frac{1}{2} d_x^T \nabla_{xx}^2 \mathcal{L}(x_k, s_k, \lambda_h, \lambda_g) d_x - \mu e^T S_k^{-1} d_s + \frac{1}{2} d_s^T \Sigma_k d_s \tag{2.4}$$

$$\begin{aligned} \text{subject to} \qquad A_h(x_k)^T d_x + h(x_k) &= r_h && (2.5)\\ A_g(x_k)^T d_x + d_s + g(x_k) + s_k &= r_g && (2.6)\\ (d_x, d_s) &\in T_k. && (2.7) \end{aligned}$$

Here Σ_k is an $m \times m$ positive definite diagonal matrix that represents either the Hessian of the Lagrangian (2.2) with respect to s or an approximation to it. As we will see in the next section, the choice of Σ_k is of crucial importance because it determines whether the iteration has primal or primal-dual characteristics. Ideally, we would like our step to satisfy (2.5)-(2.6) with $r = (r_h, r_g) = 0$, i.e. to satisfy the linearized constraints. However, this may may be inconsistent with (2.7), so we choose the residual vector r to be the smallest vector such that (2.5)-(2.7) are consistent (with some margin). This computation is done by solving the preliminary subproblem in which we compute the *normal step*, described in §3.2. The closed and bounded set T_k defines the region around x_k where the quadratic model

(2.4) and the linearized constraints (2.5)-(2.6) can be trusted to be good approximations to the problem, and also ensures the feasibility of the slack variables. This trust region also guarantees that (2.4)-(2.7) has a finite solution even when $\nabla^2_{xx}\mathcal{L}(x_k, s_k, \lambda_h, \lambda_g)$ is not positive definite. The precise form of the trust region T_k requires careful consideration and will be described in the next section.

We compute a step $d = (d_x, d_s)$ by approximately minimizing the quadratic model (2.4) subject to the constraints (2.5)-(2.7), as will be described in §3.2. We then determine if the step is acceptable according to the reduction obtained in the following merit function

$$\phi(x, s; \nu) = f(x) - \mu \sum_{i=1}^{m} \ln s_i + \nu \left\| \begin{bmatrix} h(x) \\ g(x) + s \end{bmatrix} \right\|_2, \tag{2.8}$$

where $\nu > 0$ is a penalty parameter. This non-differentiable merit function has been successfully used in the SQP algorithm of Byrd and Omojokun [6, 34, 31], and has been analyzed in the context of interior point methods in [7]. We summarize this SQP trust region approach as follows.

Algorithm II: SQP Trust Region Algorithm for the Barrier Problem (2.1)

Input parameters $\mu > 0$ and $\epsilon_\mu > 0$ and values k, x_k and $s_k > 0$;
set trust region T_k; compute Lagrange multipliers λ_h and λ_g.
Repeat until $E(x_k, s_k; \mu) \le \epsilon_\mu$
  Compute $d = (d_x, d_s)$ by approximately solving (2.4)-(2.7).
  If the step d provides sufficient decrease in ϕ
    then set $x_{k+1} = x_k + d_x$, $s_{k+1} = s_k + d_s$,
      compute new Lagrange multiplier estimates λ_h and λ_g,
      and possibly enlarge the trust region;
    else set $x_{k+1} = x_k$, $s_{k+1} = s_k$, and shrink the trust region.
  Set $k := k + 1$.
end

Algorithm II is called at each execution of step 1 of Algorithm I. The iterates of Algorithm II are indexed by (x_k, s_k), where the index k runs continuously during Algorithm I. In the next section we present a full description of Algorithm II, which forms the core of the new interior point algorithm.

3. Algorithm for Solving the Barrier Problem

Many details of the SQP trust region method outlined in Algorithm II need to be developed. We first give a precise description of the subproblem (2.4)-(2.7), including the choice of the diagonal matrix Σ_k which gives rise to primal or primal-dual iterations. Further, we define the right hand side vectors (r_h, r_g), the form of the trust region constraint T_k, and the choice of Lagrange multiplier estimates. Once a complete description of the subproblem (2.4)-(2.7) has been given, we will present our procedure for finding an approximate solution of it. We will conclude this section with a discussion of various other details of implementation of the new algorithm.

3.1. Formulation of the Subproblem

Let us begin by considering the quadratic model (2.4). We have mentioned that SQP methods choose the Hessian of this model to be the Hessian of the Lagrangian of the problem under consideration, or an approximation to it. Since the problem being solved by Algorithm II is the barrier problem (2.1), which has a separable objective function in the variables x and s, its Hessian consists of two blocks. As indicated in (2.4), we choose the Hessian of the quadratic model with respect to d_x to be $\nabla^2_{xx}\mathcal{L}(x_k, s_k, \lambda_h, \lambda_g)$ (which we abbreviate as $\nabla^2_{xx}\mathcal{L}_k$) but consider several choices for the Hessian Σ_k of the model with respect to d_s. The first choice is to define $\Sigma_k = \nabla^2_{ss}\mathcal{L}_k$, which gives

$$\Sigma_k = \mu S_k^{-2}. \tag{3.1}$$

The general algorithm studied in Byrd, Gilbert and Nocedal [7] defines Σ_k in this manner.

To study the effect of Σ_k in the step computation, let us analyze the simple case when the matrix $\nabla^2_{xx}\mathcal{L}_k$ is positive definite on the null space of the constraint gradients, when the residual (r_h, r_g) is zero, and when the step generated by (2.4)-(2.7) lies strictly inside the trust region. In this case the subproblem (2.4)-(2.6) has a unique solution $d = (d_x, d_s)$ which satisfies the linear system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & 0 & A_h(x_k) & A_g(x_k) \\ 0 & \Sigma_k & 0 & I \\ A_h^T(x_k) & 0 & 0 & 0 \\ A_g^T(x_k) & I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ \lambda_h^+ \\ \lambda_g^+ \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ \mu S_k^{-1} e \\ -h(x_k) \\ -g(x_k) - s_k \end{bmatrix}. \tag{3.2}$$

If Σ_k is defined by (3.1), we call this approach a *primal method*. In this case, it is easy to verify (see e.g. [19, 42, 7]) that the system (3.2) is equivalent to a Newton iteration on the KKT conditions of the barrier problem (2.1), which are given by

$$\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g = 0 \tag{3.3}$$
$$-\mu S^{-1} e + \lambda_g = 0 \tag{3.4}$$
$$h(x) = 0 \tag{3.5}$$
$$g(x) + s = 0. \tag{3.6}$$

Several authors, including Jarre and S. Wright [30], M. Wright [41] and Conn, Gould and Toint [16] have given arguments suggesting that the primal search direction will often cause the slack variables to become negative, and can be inefficient. Although those papers consider a different formulation of the problem, it is easy to see [29] that the arguments apply in our case.

Research in linear programming [42] has shown that a more effective interior point method is obtained by considering the *perturbed KKT system*

$$\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g = 0 \tag{3.7}$$
$$S\lambda_g - \mu e = 0 \tag{3.8}$$
$$h(x) = 0 \tag{3.9}$$
$$g(x) + s = 0, \tag{3.10}$$

which is obtained by multiplying (3.4) by S. Even though (3.4)-(3.6) and (3.8)-(3.10) have the same solutions, applying Newton's method to them will produce different iterates. It is well-known, and also easy to verify, that a Newton step on (3.8)-(3.10) is given by the solution to (3.2), with

$$\Sigma_k = S_k^{-1}\Lambda_g. \tag{3.11}$$

Here $\Lambda_g = \mathrm{diag}(\lambda_g^1, \ldots, \lambda_g^m)$ contains the Lagrange multiplier estimates corresponding to the inequality constraints. The system (3.2) with Σ_k defined by (3.11) is called the *primal-dual* system. This choice of Σ_k may be viewed as an approximation to $\nabla^2_{ss}\mathcal{L}_k$ since, by (3.4), at the solution (x, s, λ) of the barrier problem the equation $\mu S^{-1} = \Lambda_g$ is satisfied. Substituting this equation in (3.1) gives (3.11).

The system (3.7)-(3.10) has the advantage that the derivatives of (3.8) are bounded as any slack variables approach zero, which is not the case with (3.4). In fact, analysis of the primal-dual step, as well as computational experience with linear programs, has shown that it overcomes the drawbacks of the primal step: it does not tend to violate the constraints on the slacks, and usually makes excellent progress towards the solution (see e.g. [30, 41, 42, 39]). These observations suggest that the primal-dual model in which Σ_k is given by (3.11) is likely to perform better than the primal choice (3.1). Of course, these arguments do not apply directly to our algorithm which solves the SQP subproblem inexactly, and whose trust region constraint may be active. Nevertheless, as the iterates approach a solution point, the algorithm will resemble more and more an interior point method in which a Newton step on some form of the KKT conditions of the barrier problem is taken at each step.

Lagrange multiplier estimates are needed both in the primal-dual choice (3.11) of Σ_k and in the Hessian $\nabla^2\mathcal{L}_{xx}(x_k, s_k, \lambda_h, \lambda_g)$. To complete our description of the quadratic model (2.4) we must discuss how these multipliers are computed.

Lagrange Multipliers

Since the method we will use for finding an approximate solution to the subproblem (2.4)-(2.7) does not always provide Lagrange multiplier estimates as a side computation, we will obtain them using a least squares approach. As is done in some SQP methods [21, 3], which compute least squares estimates based on the stationarity conditions at the current iterate, we will choose the vector $\lambda = (\lambda_h, \lambda_g)$ that minimizes the Euclidean norm of (3.7)-(3.8). This gives the formula

$$\lambda_k = \begin{bmatrix} \lambda_h \\ \lambda_g \end{bmatrix} = \lambda^{\mathrm{LS}}(x_k, s_k, \mu) = \left(\hat{A}_k^T \hat{A}_k\right)^{-1} \hat{A}_k^T \begin{bmatrix} -\nabla f(x_k) \\ \mu e \end{bmatrix}, \tag{3.12}$$

where

$$\hat{A}_k = \begin{bmatrix} A_h(x_k) & A_g(x_k) \\ 0 & S_k \end{bmatrix}. \tag{3.13}$$

The computation of (3.12) will be performed by solving an augmented system, instead of factoring $\hat{A}_k^T \hat{A}_k$, as will be discussed in §3.4.

We should note that the multiplier estimates λ_g obtained in this manner may not always be positive, and it may be questionable to use them in this case in the primal-dual choice of Σ_k given by (3.11). In particular, since the Hessian of the barrier term $-\mu \sum \ln s_i$ is known to be positive definite, it seems undesirable to create an indefinite approximation Σ_k to it. On the other hand, one could argue that trust region methods can handle indefinite approximations and therefore that the multipliers need not be modified. We cannot see a compelling argument in favor of either strategy. In primal-dual interior point methods for linear programming, the initial Lagrange multiplier estimate is chosen to be positive, and in subsequent iterations a backtracking line search ensures that all new multiplier estimates remain safely positive (see e.g. [42]). Here we follow a different approach, not enforcing the positivity of the multipliers λ_g, but ensuring that the quadratic model remains convex in the slack variables. To do so, in the primal-dual version of the algorithm we define the i-th diagonal element of Σ_k as

$$\sigma_k^i = \begin{cases} \lambda_g^i / s^i & \text{if } \lambda_g^i > 0 \\ \mu/(s^i)^2 & \text{otherwise.} \end{cases} \tag{3.14}$$

This means, in particular, that when a multiplier λ_g^i given by (3.12) is negative, the corresponding entry in the primal-dual matrix Σ_k coincides with the corresponding entry in the primal Hessian.

To avoid an abrupt change in Σ_k when μ is decreased, we modify the definition of λ_k slightly in the primal-dual version of the algorithm. If (x_k, s_k) is the starting point for a new barrier subproblem (i.e. the input in Algorithm II), then in the formula (3.14) λ_g is the multiplier from the last iterate of the previous barrier problem.

Thus the definition of the multipliers is

$$\lambda_k = \begin{cases} \lambda^{\mathrm{LS}}(x_k, s_k, \mu) & \text{in primal version} \\ \lambda^{\mathrm{LS}}(x_k, s_k, \bar{\mu}) & \text{in primal-dual version,} \end{cases} \tag{3.15}$$

where $\bar{\mu}$ is the value of the barrier parameter used in the computation of (x_k, s_k). As mentioned earlier, other strategies for computing multiplier estimates can be used, and we do not yet know which choice might be preferable in practice.

This approach could just barely be considered a primal-dual method, as other primal-dual methods treat the multipliers λ_h, λ_g as independent variables. In that respect our approach is much closer to those SQP methods where the multipliers have a subordinate role, being estimated as a function of the primal variables, and not appearing explicitly in the merit function.

The Trust Region

Algorithm II stipulates that the step (d_x, d_s) must be restricted to a set T_k, called the trust region. We will define T_k so as to accomplish two goals. First of all it should restrict the step to a region where the quadratic model (2.4) is a good approximation of the Lagrangian (2.2), and where the linear equations (2.5)-(2.6) are good approximations to the constraints. This is the basic philosophy of trust regions and is normally achieved by imposing a bound of the form $\|(d_x, d_s)\| \leq \Delta_k$, where the trust region radius Δ_k is updated at every iteration according to how successful the step has been.

We will impose such a bound on the step, but the shape of the trust region must also take into account other requirements of Algorithm II. Since the slack variables should not approach zero prematurely, we introduce the scaling S_k^{-1} that penalizes steps d_s near the boundary of the feasible region. This scaled trust region will be defined as

$$\|(d_x, S_k^{-1} d_s)\|_2 \leq \Delta_k, \tag{3.16}$$

and we will allow Δ_k to be greater than 1. The second objective of our trust region is to ensure that the slack variables remain positive. For this purpose we impose the well-known [42, 39] fraction to the boundary rule

$$s_k + d_s \geq (1 - \tau) s_k, \tag{3.17}$$

where $\tau \in (0, 1)$; in our tests we use $\tau = 0.995$. Combining this inequality, which can be rephrased as $d_s \geq -\tau s_k$, with (3.16) we obtain the final form of the trust region,

$$\|(d_x, S_k^{-1} d_s)\|_2 \leq \Delta_k, \quad \text{and} \quad d_s \geq -\tau s_k. \tag{3.18}$$

We have experimented with other forms of the trust region, in particular with box-shaped trust regions defined by an ℓ_∞ norm, but so far (3.18) appears to be the most appropriate for our algorithm.

Now that the quadratic model (2.4) and the trust region (2.7) have been defined, it only remains to specify the choice of the residual vector $r = (r_h, r_g)$ in (2.5)-(2.6). This vector will be determined during the course of solving the subproblem, as discussed next.

3.2. Solution of the Quadratic Subproblem

We will use the decomposition proposed by Byrd and Omojokun [6, 34] to find an approximate solution of the subproblem (2.4)-(2.7). In this approach the step d is a combination of a *normal step* that attempts to satisfy the linear constraints (2.5)-(2.6) as well as possible, and a *tangential step* that lies on the tangent space of the constraints and that tries to achieve optimality. The efficiency of the new algorithm depends, to a great extent, on how these two components of the step are computed.

Throughout this section we omit the iteration subscript, and write s_k as s, $A_h(x_k)$ as A_h, etc.

Normal Step

It is clear [40] that restricting the size of the step d by means of the trust region bounds (3.18) may preclude d from satisfying the linearized constraints (2.5)-(2.6) with $r = 0$. To find a value of r that makes the quadratic subproblem feasible, we first compute the normal step v, that lies well within the trust region and that approximately satisfies (2.5)-(2.6), in the least squares sense. To do this, we choose a parameter $\zeta \in (0, 1)$ (in our code we use the value $\zeta = 0.8$) and formulate the following subproblem in the variable $v = (v_x, v_s)$

$$\begin{aligned} &\min_v \|A_h^T v_x + h\|_2^2 + \|A_g^T v_x + v_s + g + s\|_2^2 \\ &\text{subject to} \quad \begin{array}{rcl} \|(v_x, S^{-1} v_s)\|_2 & \leq & \zeta\Delta \\ v_s & \geq & -\tau s/2. \end{array} \end{aligned} \tag{3.19}$$

To simplify the constraints we define

$$\tilde{v} = (v_x, \tilde{v}_s) = (v_x, S^{-1} v_s).$$

Performing this transformation, recalling the definition (3.13) of $\hat{A}$, squaring and expanding the quadratic objective and ignoring constant terms, we obtain

$$\min_{\tilde{v}} \ m(\tilde{v}) \ \equiv 2 \begin{bmatrix} h^T & (g+s)^T \end{bmatrix} \hat{A}^T \begin{bmatrix} v_x \\ \tilde{v}_s \end{bmatrix} + \begin{bmatrix} v_x^T & \tilde{v}_s^T \end{bmatrix} \hat{A}\hat{A}^T \begin{bmatrix} v_x \\ \tilde{v}_s \end{bmatrix} \tag{3.20}$$

$$\text{subject to} \quad \|\tilde{v}\|_2 \leq \quad \zeta\Delta \tag{3.21}$$

$$\tilde{v}_s \geq \quad -\tau/2. \tag{3.22}$$

We compute an approximate solution of this problem by means of an adaptation of Powell's *dogleg method* [37], which provides a relatively inexpensive solution that is good enough to allow our algorithm to be robust and rapidly convergent. Like Powell's dogleg, it provides as least as much decrease on (3.19) as a truncated steepest descent step, and it equals the unconstrained minimizer of (3.19) if that vector satisfies the constraints of the subproblem. This first property, together with the fact that it lies in the range space of $\hat{A}$, implies that it satisfies the conditions placed on the normal step for the global convergence theory in [7].

We first calculate the Cauchy point $\tilde{v}^{\mathrm{CP}}$ for problem (3.20)-(3.21), which is obtained by minimizing the quadratic (3.20) along the steepest descent direction, starting from $\tilde{v} = 0$. A simple computation shows that

$$\tilde{v}^{\mathrm{CP}} = \begin{bmatrix} v_x^{\mathrm{CP}} \\ \tilde{v}_s^{\mathrm{CP}} \end{bmatrix} = -\alpha \hat{A} \begin{bmatrix} h \\ g+s \end{bmatrix}, \tag{3.23}$$

where α is given by

$$\alpha = \frac{\left\| \hat{A} \begin{bmatrix} h \\ g+s \end{bmatrix} \right\|_2^2}{\begin{bmatrix} h^T & g^T + s^T \end{bmatrix} (\hat{A}^T \hat{A})^2 \begin{bmatrix} h \\ g+s \end{bmatrix}}.$$

Note that this computation is inexpensive, requiring only matrix-vector multiplications and no matrix factorizations.

We then compute the Newton step $\tilde{v}^{\mathrm{N}}$, which in our case is defined as the minimum norm minimizer of (3.20). It is given by

$$\tilde{v}^{\mathrm{N}} = \begin{bmatrix} v_x^{\mathrm{N}} \\ \tilde{v}_s^{\mathrm{N}} \end{bmatrix} = -\hat{A}(\hat{A}^T \hat{A})^{-1} \begin{bmatrix} h \\ g+s \end{bmatrix}. \tag{3.24}$$

The computation of $\tilde{v}^{\mathrm{N}}$ will be done by solving an augmented system, instead of factoring $\hat{A}^T \hat{A}$, as will be discussed in §3.4.

The Cauchy and Newton steps define the dogleg path, which consists of the two line segments from $\tilde{v} = 0$ to $\tilde{v} = \tilde{v}^{\mathrm{CP}}$, and from $\tilde{v} = \tilde{v}^{\mathrm{CP}}$ to $\tilde{v} = \tilde{v}^{\mathrm{N}}$. We compute the normal step by minimizing $m(\tilde{v})$ subject to (3.21) and (3.22) along this path, and along the Newton direction, as described below.

Dogleg Procedure.

Compute $\tilde{v}^{CP}$ and $\tilde{v}^{N}$.
$\theta_1 = \max\{\theta \in (0,1] | \theta\tilde{v}^{\mathrm{N}} \text{ is feasible}\}$
If $\theta_1 = 1$ **then**
$\tilde{v} = \tilde{v}^{\mathrm{N}}$
Else
$\theta_2 = \max\{\theta \in (0,1] | (1-\theta)\tilde{v}^{\mathrm{CP}} + \theta\tilde{v}^{\mathrm{N}} \text{ is feasible for (3.21) and (3.22) }\}$
If no such value θ_2 exists **then**
$\theta_3 = \max\{\theta \in (0,1] | \theta\tilde{v}^{\mathrm{CP}} \text{ is feasible}\}$
$\tilde{v}^{\mathrm{DL}} = \theta_3\tilde{v}^{\mathrm{CP}}$
Else
$\tilde{v}^{\mathrm{DL}} = (1-\theta_2)\tilde{v}^{\mathrm{CP}} + \theta_2\tilde{v}^{\mathrm{N}}$
Endif
If $m(\tilde{v}^{\mathrm{DL}}) < m(\theta_1\tilde{v}^{\mathrm{N}})$ **then**
$\tilde{v} = \tilde{v}^{\mathrm{DL}}$
Else
$\tilde{v} = \theta_1\tilde{v}^{\mathrm{N}}$
Endif
Endif
$v = (v_x, S\tilde{v}_s)$

Since the model function m is convex, it decreases along the dogleg path, and thus the dogleg point $\tilde{v}^{\mathrm{DL}}$ minimizes m along that path, subject to (3.21) and (3.22). Note that even if $\tilde{v}^{\mathrm{CP}}$ and $\tilde{v}^{\mathrm{N}}$ are infeasible, the line from $\tilde{v}^{\mathrm{CP}}$ to $\tilde{v}^{\mathrm{N}}$ may still contain a feasible segment. Also, to try to achieve a greater reduction in the model function, we compare the dogleg step with the Newton step truncated to the feasible region, and choose whichever of these two points gives a lower value of m. Finally, we obtain the normal step by transforming $\tilde{v}$ into the original space of variables.

For future reference we note that the step $\tilde{v}$ lies in the range space of $\hat{A}$; see (3.23) and (3.24).

An alternative to the dogleg method is to compute the normal step by means of Steihaug's implementation of the conjugate gradient method [38]. This is described in detail in [29] (see also [31]), and is certainly a viable option. We prefer the dogleg method in this study because it allows us to compute the normal step using a direct linear algebra solver, thereby avoiding the difficulties that can arise when applying the conjugate gradient method to ill-conditioned systems. In addition, the matrix factorization performed during the computation of the Lagrange multipliers can be saved and used to compute the normal step, giving significant savings in computation. We will return to this in §3.4.

Tangential Problem

Once the normal step v is computed, we define the vectors r_h and r_g in (2.5)-(2.6) as the residuals in the normal step computation, i.e.

$$r_h = A_h^T v_x + h, \qquad r_g = A_g^T v_x + v_s + g + s.$$

The subproblem (2.4)-(2.7) therefore takes the form

$$\min \nabla f^T d_x - \mu e^T S^{-1} d_s + \frac{1}{2}(d_x^T \nabla_{xx}^2 \mathcal{L} d_x + d_s^T \Sigma d_s) \tag{3.25}$$

$$\begin{aligned} \text{subject to} \quad A_h^T d_x &= A_h^T v_x && (3.26)\\ A_g^T d_x + d_s &= A_g^T v_x + v_s && (3.27)\\ \|(d_x, S^{-1} d_s)\|_2 &\le \Delta && (3.28)\\ d_s &\ge -\tau s. && (3.29)\end{aligned}$$

We will devote much attention to this subproblem, whose solution represents the most complex and time consuming part of the new algorithm.

Let us motivate our choice of the residual vectors r_h and r_g. First, the constraints (3.26)-(3.29) are now feasible since $d = v$ clearly satisfies them (recall that $\zeta < 1$ in (3.19)). Second, we are demanding that the total step d makes as much progress towards satisfying the constraints (3.26)-(3.27) as the normal step v.

To find an approximate solution of (3.25)-(3.29), we write $d = v + w$, where v is the normal step and w, which is to be determined, is tangent to the (scaled) constraint gradients. Introducing the same change of variables as in the normal step computation, we define

$$\tilde{d} = \begin{pmatrix} \tilde{d}_x \\ \tilde{d}_s \end{pmatrix} = \begin{pmatrix} d_x \\ S^{-1} d_s \end{pmatrix} = \begin{pmatrix} v_x \\ \tilde{v}_s \end{pmatrix} + \begin{pmatrix} w_x \\ \tilde{w}_s \end{pmatrix} = \tilde{v} + \tilde{w}. \tag{3.30}$$

Using this and defining

$$G = \begin{bmatrix} \nabla_{xx}^2 \mathcal{L} & 0 \\ 0 & S \Sigma S \end{bmatrix}, \tag{3.31}$$

the objective of (3.25) can be expressed as

$$q(\tilde{v} + \tilde{w}) \equiv (\nabla f^T, \; -\mu e^T)(\tilde{v} + \tilde{w}) + \frac{1}{2}(\tilde{v} + \tilde{w})^T G (\tilde{v} + \tilde{w}). \tag{3.32}$$

The constraint (3.28) can be rewritten as

$$\|\tilde{d}\|_2^2 = \|\tilde{v} + \tilde{w}\|_2^2 \le \Delta^2. \tag{3.33}$$

We have noted in §3.2 that the (scaled) normal step $\tilde{v}$ lies in the range space of $\hat{A}$, and we will require that w satisfies $\hat{A}^T \tilde{w} = 0$. Thus $\tilde{w}^T \tilde{v} = 0$, and (3.28) can be expressed as

$$\|\tilde{w}\|_2^2 \le \Delta^2 - \|\tilde{v}\|_2^2.$$

Using this, (3.32) and the definitions (3.30), we can rewrite (3.25)-(3.29) as

$$\min_{\tilde{w}} \; q(\tilde{v} + \tilde{w}) \equiv q(\tilde{v}) + \nabla f^T w_x - \mu e^T \tilde{w}_s + (G\tilde{v})^T \tilde{w} + \frac{1}{2}(\tilde{w}^T G \tilde{w}) \tag{3.34}$$

$$\begin{aligned} \text{subject to} \quad A_h^T w_x &= 0 && (3.35)\\ A_g^T w_x + S\tilde{w}_s &= 0 && (3.36)\\ \|\tilde{w}\|_2^2 &\le \Delta^2 - \|\tilde{v}\|_2^2, && (3.37)\\ \tilde{w}_s &\ge -\tau e - \tilde{v}_s. && (3.38)\end{aligned}$$

We call this the *tangential* subproblem. Clearly this subproblem can be very expensive to solve. However, the shape of the feasible region for this problem resembles a trust region in that the boundaries of the feasible region are never close to the origin ($\tilde{w} = 0$) in the scaled coordinates. So it is reasonable to expect that an adaptation of a method for computing an approximate solution of a trust region problem, such as the CG iteration proposed by Steihaug, will be efficient in this context. We will follow this approach and apply the conjugate gradient (CG) method to the quadratic objective (3.34), while forcing the CG iterates to satisfy the constraints (3.35)-(3.36). To take into account the trust region and the possibility of indefiniteness in the model, we will terminate the CG iteration using the stopping tests of Steihaug [38]. We will also precondition the CG iteration.

Rather than simply presenting this CG iteration, we will now describe in detail the steps that lead to it, and will motivate our preconditioning strategy.

Since $\tilde{w}$ is assumed to lie in the null space of $\hat{A}^T$, it can be expressed as

$$\tilde{w} = \tilde{Z}u \equiv \begin{pmatrix} Z_x \\ \tilde{Z}_s \end{pmatrix} u, \tag{3.39}$$

for some vector $u \in \mathbf{R}^{n-t}$, and where $\tilde{Z}$ is a basis for the null space of $\hat{A}^T$. The constraints (3.35)-(3.36) can be written as $\hat{A}^T\tilde{w} = 0$, and are therefore satisfied by any $\tilde{w}$ of the form (3.39). Therefore the tangential problem (3.34)-(3.38) can be stated as

$$\min_u \; q(\tilde{v} + \tilde{Z}u) \tag{3.40}$$

$$\begin{aligned} \text{subject to} \quad \|\tilde{Z}u\|_2^2 \;&\leq\; \Delta^2 - \|\tilde{v}\|_2^2, \\ \tilde{Z}_s u \;&\geq\; -\tau e - \tilde{v}_s. \end{aligned} \tag{3.41}$$

We will precondition the CG iteration so as to eliminate the inefficiencies that can arise from an ill-conditioned null space basis Z. Note that if we were to apply the unpreconditioned CG iteration for minimizing (3.40), since the Hessian of (3.40) is

$$\tilde{Z}^T G \tilde{Z},$$

a poor choice of Z could make this matrix unnecessarily ill-conditioned, causing the CG iteration to be very slow. Such a poor choice of null space basis could occur, for example, when using the easily computable basis

$$\tilde{Z} = \begin{bmatrix} \hat{A}_1^{-1}\hat{A}_2 \\ -I \end{bmatrix}$$

based on the basic-nonbasic partition $\hat{A}^T = [\hat{A}_1 \; \hat{A}_2]$. This problem could be avoided by preconditioning the CG iteration for minimizing (3.40) by the matrix

$$\tilde{Z}^T\tilde{Z}, \tag{3.42}$$

in which case the rate of convergence is governed by the spectrum of

$$(\tilde{Z}^T\tilde{Z})^{-\frac{1}{2}}\tilde{Z}^T G\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-\frac{1}{2}}. \tag{3.43}$$

Since the matrix $\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-\frac{1}{2}}$ has orthonormal columns, the behavior of the CG iteration will now be identical to that obtained when $\tilde{Z}$ is a basis with orthonormal columns. Note also from (3.4) that $\mu S^{-1} \approx \Lambda_g$ near the solution of the barrier problem, and thus by (3.11) $S\Sigma S$ is close to μI. From (3.31) we see that (3.43) does become increasingly ill-conditioned as $\mu \to 0$, but this ill-conditioning does not greatly degrade the performance of the CG method since it results in one tight cluster of small eigenvalues. The numerical tests described in §4 confirm that the solution by the CG method does not become significantly more difficult as μ tends to zero.

The conjugate gradient iteration computes estimates of the minimizer of (3.40) by the recursion (see e.g. [21])

$$u^+ = u + \alpha\delta, \tag{3.44}$$

where the parameter α is chosen to minimize the quadratic objective q along the direction δ. Since the gradient of q with respect to u is $\tilde{Z}^T\nabla q(\tilde{v} + \tilde{Z}u)$, and since our preconditioner is given by (3.42), the conjugate directions δ are recurred by

$$\delta^+ = -(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T\nabla q(\tilde{v} + \tilde{Z}u) + \beta\delta, \tag{3.45}$$

where the parameter β is initially zero and is chosen at subsequent steps to maintain conjugacy.

However, because of the computational cost of manipulations with the preconditioner (3.42), it is preferable to perform the CG iteration in the full space rather than the reduced space. More specifically, by applying the transformation (3.39) to (3.44)-(3.45), we obtain the following iteration in the variable $\tilde{w}$ of problem (3.34),

$$\begin{aligned} \tilde{w}^+ &= \tilde{w} + \alpha p \qquad\qquad (p \equiv \tilde{Z}\delta) && (3.46)\\ p^+ &= -\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T\nabla q(\tilde{v} + \tilde{w}) + \beta p. && (3.47)\end{aligned}$$

We have therefore obtained a CG iteration to minimize the objective (3.34) of the tangential subproblem that, by construction, satisfies the constraints (3.35)-(3.36). Note that the matrix $\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T$ is actually the orthogonal projection onto the null space of $\hat{A}^T$ and thus can be expressed as

$$P = \tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T = I - \hat{A}(\hat{A}^T\hat{A})^{-1}\hat{A}^T. \tag{3.48}$$

We compute projections of the form Pr by solving an augmented system whose coefficient matrix coincides with that used in the normal step and Lagrange multiplier computations, as will be discussed in §3.4. The resulting iteration is equivalent to the preconditioned CG iteration in the null space, described above, but allows us to totally bypass the computation of the null space matrix Z. The computation of the projected residual Pr corresponds to the preconditioning step in the null space iteration.

Because of the trust region constraint (3.37), and due to the possibility of indefiniteness in the quadratic model, we use Steihaug's stopping tests in the iteration (3.46)-(3.47): we terminate if the projected gradient of q is smaller than a prescribed tolerance, if the direction p^+ is one of negative curvature, or if the iterates violate the trust region norm constraint (3.37). We include an additional step truncation to satisfy the bound constraint (3.38).

PCG Procedure: Projected CG Method for the Tangential Subproblem (3.34)-(3.38).

Set $\tilde{w} = 0$, $r = (r_x, r_s) = (\nabla f, -\mu e) + G\tilde{v}$, $g = Pr$, $p = -g$, tol $= 0.01\sqrt{g^T r}$.

Repeat at most $2(n - t)$ times, or until a stopping test is satisfied.

If $p^T G p \leq 0$

then $\tilde{w}^+ = \tilde{w} + \theta p$, where $\theta > 0$ is such that $\|\tilde{w}^+\|_2 = \Delta$; STOP

$\alpha = r^T g / p^T G p$

$\tilde{w}^+ = \tilde{w} + \alpha p$

If $\|\tilde{w}^+\|_2 > \Delta$

then $\tilde{w}^+ = \tilde{w} + \theta p$, where $\theta > 0$ is such that $\|\tilde{w}^+\|_2 = \Delta$; STOP

$r^+ = r + \alpha G p$

$g^+ = P r^+$

If $(g^+)^T r^+ <$ tol, STOP

$\beta = (r^+)^T g^+ / r^T g$

$p^+ = -g^+ + \beta p$

$\tilde{w} \leftarrow \tilde{w}^+, \quad r \leftarrow r^+, \quad p \leftarrow p^+$

End repeat

If $\tilde{w}^+$ does not satisfy the slack variable bound (3.38), restore the last feasible iterate $\tilde{w}$ and the direction p computed at that point. Set $\tilde{w}^+ = \tilde{w} + \theta p$, where $\theta > 0$ is is the largest value such that $\tilde{w} + \theta p$ is feasible. Set $w = (w_x, w_s) = (\tilde{w}_x^+, S\tilde{w}_s^+)$.

Note that during the **Repeat** loop we only test whether the trust region norm constraint (3.37) is satisfied, and ignore the slack variable bound (3.38). The reason for this is that it can be shown [38] that the norm of the iterates $\|\tilde{w}\|_2$ increases during the conjugate gradient iteration, so that once an iterate violates (3.37), all subsequent iterates will also violate this constraint. It is therefore sensible to stop iterating when (3.37) is violated. However, the slack bounds (3.38) could be crossed several times, so we do not check feasibility with respect to the bound until we have gone as far as possible subject to the norm constraint. Thus, at the end of the **Repeat** loop the point $\tilde{w}^+$ may not satisfy the slack variable bounds (3.38). In this case we select the last intersection point of the path generated by the iterates $\tilde{w}$ with the bounds (3.38). This strategy has the potential of being wasteful, because we could generate a series of iterates that violate the slack variable bounds and never return to the feasible region. To control this cost we include a limit of $2(n - t)$ CG iterations in the tangential step computation. In the tests described in §4, the infeasible CG steps accounted for about 2% of the total, and our strategy appears to pay off because, in our experience, when the iterates did return to the feasible region they usually generated a much better step than the one obtained when the bounds were first encountered.

In §3.4 we will show how the projection Pr^+ can be computed by solving an augmented system whose coefficient matrix is the same as that needed in the normal step and Lagrange multiplier computations.

3.3. Merit Function, Trust Region, and Second-Order Correction

The merit function $\phi(x, s; \nu)$, defined by (2.8), is used to determine whether the total step $d = v + w$ is acceptable, and also provides information on how to update the trust region radius Δ. The penalty parameter ν (not to be confused with the barrier parameter μ) balances the relative contribution of the objective function and constraints, and needs to be selected at every iteration so that the step d and the merit function ϕ are compatible. By this we mean that if the trust region is sufficiently small, then the step d must give a reduction in ϕ.

We approximate the change in the merit function due to the step d by the *predicted reduction* defined as

$$\text{pred}(d) = -q(\tilde{v} + \tilde{w}) + \nu \text{vpred}, \tag{3.49}$$

where q is the objective in the tangential subproblem (3.34) and vpred is the reduction provided by the normal step,

$$\text{vpred} = \left\| \begin{bmatrix} h \\ g + s \end{bmatrix} \right\| - \left\| \begin{bmatrix} h \\ g + s \end{bmatrix} + \hat{A}^T \tilde{v} \right\|. \tag{3.50}$$

The definition (3.49) is motivated and analyzed in [7], and is similar to the measures used in other trust region algorithms for constrained optimization. We demand that ν be large enough that pred(d) be positive and proportional to vpred, i.e.

$$\text{pred}(d) \geq \rho \nu \text{vpred}, \tag{3.51}$$

where $0 < \rho < 1$ (in our code we use the value $\rho = 0.3$).

We see from (3.49) that we can enforce inequality (3.51) by choosing the penalty parameter ν so that

$$\nu \geq \frac{q(\tilde{v} + \tilde{w})}{(1 - \rho)\text{vpred}}. \tag{3.52}$$

As has been argued in [7], if $m(\tilde{v}) = 0$, then $\tilde{v} = 0$, which implies $q(\tilde{v} + \tilde{w}) \leq 0$, and so (3.51) is satisfied for any value of ν. In this case ν can be defined as its value in the previous iteration of Algorithm II, ν^-. Thus we update ν as follows.

Penalty Parameter Procedure.

If $m(\tilde{v}) = 0$ **then**

$\nu = \nu^-$

Else

$\nu = \max\left\{\nu^-, \frac{q(\tilde{v}+\tilde{w})}{(1-\rho)\text{vpred}}\right\}.$

End

This procedure is applied while the barrier parameter μ is fixed. Thus, for a fixed barrier problem the penalty parameter ν is monotonically increasing as the iterations progress, which is an important property for the global convergence analysis of the algorithm [7]. If the value of the barrier parameter was just changed at the beginning of the current iteration, the value of ν^- to be used in the Penalty Parameter Procedure is reset to a default initial value.

Now that the merit function has been completely specified, let us consider how to use it to determine if a step d is to be accepted by Algorithm II. As is common in trust region methods, we compute the *actual reduction* in the merit function,

$$\text{ared}(d) = \phi(x, s; \nu) - \phi(x + d_x, s + d_s; \nu), \tag{3.53}$$

and accept d only if it gives a sufficient reduction in ϕ, in the sense that

$$\gamma \equiv \frac{\text{ared}(d)}{\text{pred}(d)} \geq \eta, \tag{3.54}$$

where $0 < \eta < 1$ (in our code we use $\eta = 10^{-8}$). Using essentially the same argument as in [7] it can be shown that (3.54) will be satisfied if the trust region radius Δ is sufficiently small.

If a step is accepted then the trust region is increased as follows:

$$\Delta^+ = \begin{cases} \max\{7\|d\|, \Delta\} & \text{if} \quad \gamma \geq 0.9 \\ \max\{2\|d\|, \Delta\} & \text{if} \quad 0.3 \leq \gamma < 0.9 \\ \Delta & \text{if} \quad \eta \leq \gamma < 0.3 \end{cases}. \tag{3.55}$$

When a step is rejected, the new trust region radius is at most one half, but not less than one tenth, of the length of the step. To determine the exact fraction of contraction in Δ we use linear or quadratic interpolation; the details are given in [36]. We also adjust Δ when the barrier parameter μ is reduced using the rule $\Delta \leftarrow \max(5\Delta, 1)$.

In order to achieve fast convergence, it is important that near the solution the trust region be inactive so that the algorithm can take full Newton steps. However, because of the non-differentiability of the merit function, it can occur that a step that approaches the solution point does not satisfy (3.54) and is rejected. (This is sometimes referred to as the Maratos effect; see e.g. [32, 11].) Since this problem is caused by an increase in the norm of the constraints due to their nonlinearity, one way to rectify the situation is to add a *second order correction* step y when (3.54) fails. (See section 14.4 in [21].) This is a Newton-like step on the constraints, and amounts to computing (3.24) at the point $x + d$. In our implementation the second order correction is applied only when the normal component is small relative to the tangential component of the step.

Procedure SOC. Second Order Correction.

If $\|\tilde{v}\| \leq 0.1\|\tilde{w}\|$ **then**

$$y = \hat{A}\left(\hat{A}^T\hat{A}\right)^{-1}\begin{bmatrix} h(x + d_x) \\ g(x + d_x) + s + d_s \end{bmatrix}$$

Else

$y = 0$

End

The total step of Algorithm II, when a second order correction is needed, is given by $d + y$.

3.4. Solution of Linear Systems

The algorithm requires the solution of three linear systems per iteration. They occur in the computation of the Lagrange multiplier estimates (3.12), in the Newton component (3.24) of the normal step, and in the projection Pr^+ required by the PCG Procedure, where P is defined by (3.48). We now show that these three systems can be solved using only one matrix factorization.

Note that the normal step (3.24) requires the solution of a system of the form

$$\hat{A}^T \hat{A} x = b,$$

where $\hat{A}$ is defined by (3.13). We compute the solution by solving the *augmented system*

$$\begin{bmatrix} I & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ -b \end{bmatrix}. \tag{3.56}$$

Similarly, the computation $g = Pr$, where P is expressed in terms of $\hat{A}$ (3.48), can be performed by solving

$$\begin{bmatrix} I & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} g \\ l \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}. \tag{3.57}$$

Moreover, if we solve the system (3.57) with r replaced by $(-\nabla f, \mu e)^T$ then, by (3.12), the vector l contains the least-squares multiplier estimates.

We use routine MA27 [27] to factor the coefficient matrix in (3.56) and (3.57). We prefer working with this augmented system, rather than factoring the normal equations matrix $\hat{A}^T \hat{A}$, because our numerical experience and the analysis given by Gould, Hribar and Nocedal [26] indicates that it is usually more accurate. Our code includes an option for detecting errors in the solution of the linear systems, and applying iterative refinement, when necessary. A detailed description of this procedure is given in [26].

3.5. Full Description of the New Interior Point Method

Having gone over all the details of our approach we can now present a complete description of the new algorithm for solving the nonlinear programming problem (1.1). We will refer to this algorithm as NITRO, for Nonlinear Interior point Trust Region Optimizer. There are primal and primal-dual versions of the algorithm, depending on how Σ_k, (3.1) and (3.11), and the Lagrange multipliers λ_k (3.15) are defined.

The stopping conditions for each barrier subproblem, and for the entire algorithm, are based on the function $E(x, s; \mu)$, which is defined by (2.3) where $(\lambda_h, \lambda_g) = \lambda^{\mathrm{LS}}(x, s, \mu)$, is defined by (3.12).

Algorithm III. Complete NITRO Algorithm

Choose a value for the parameters $\eta > 0$, $\tau \in (0,1)$, $\theta \in (0,1)$, and $\zeta \in (0,1)$, and select the stopping tolerances ϵ_μ and ϵ_{TOL}. Choose an initial value for μ, x_0, $s_0 > 0$ and Δ_0. Set $k = 0$.

Repeat until $E(x_k, s_k; 0) \leq \epsilon_{\text{TOL}}$:

    Repeat until $E(x_k, s_k; \mu) \leq \epsilon_\mu$:

        Compute the normal step $v_k = (v_x, v_s)$ by the Dogleg Procedure, described in §3.2.

        Compute Lagrange multipliers from (3.15).

        Compute $\nabla^2_{xx}\mathcal{L}(x_k, s_k, \lambda_h, \lambda_g)$ and Σ_k, using (3.1) or (3.14).

        Compute the tangential step w_k by the PCG Procedure.

        Compute the total step $d_k = v_k + w_k$.

        Update ν_k by Penalty Parameter Procedure in §3.3.

        Compute $\text{pred}_k(d_k)$ by (3.49), and $\text{ared}_k(d_k)$ by (3.53).

        If $\text{ared}_k(d_k) \geq \eta\,\text{pred}_k(d_k)$

            Then set $x_{k+1} = x_k + d_x$, $s_{k+1} = s_k + d_s$, and update Δ_{k+1} by (3.55).

        Else perform Procedure SOC to obtain $y_k = (y_x, y_s)$.

            If $y_k \neq 0$, if $\text{ared}_k(d_k + y_k) \geq \eta\,\text{pred}_k(d_k)$,

              and if $s_k + d_s + y_s \geq (1-\tau)s_k$

            then set $x_{k+1} = x_k + d_x + y_x$, $s_{k+1} = s_k + d_s + y_s$,

              and $\Delta_{k+1} = \Delta_k$.

            else set $x_{k+1} = x_k$, $s_{k+1} = s_k$, $\Delta_{k+1} \in [0.1\Delta_k, 0.5\Delta_k]$.

            Endif

        Endif

        Set $k \leftarrow k+1$.

    End

    $\mu \leftarrow \theta\mu$, $\epsilon_\mu \leftarrow \theta\epsilon_\mu$.

    Reset ν_{k-1} and Δ_k.

End

In our code we assign the following values to the parameters in the algorithm: $\eta = 10^{-8}$, $\tau = 0.995$, $\theta = 0.2$, $\zeta = 0.8$, and $\epsilon_{\text{TOL}} = 10^{-7}$. We use the following initial values: $\epsilon_\mu = 0.1$, $\mu = 0.1$, $\nu_0 = 1$, and $\Delta_0 = 1$.

Byrd, Gilbert and Nocedal [7] present a global convergence analysis for an algorithm that is very similar to the one just given. Perhaps the only significant difference is that [7] studies only the primal method where Σ_k is given by (3.1); here we are interested also in the primal-dual formulation. We expect, however, that the results of [7] can be extended without great difficulty to the primal-dual case.

4. Numerical Tests

We have tested our algorithm on a set of problems from the CUTE collection [4] whose characteristics are described in Table 1. For each problem, we give the number of variables and the total number of constraints, including equalities and general inequalities (but not bounds on the variables). We also state what kinds of conditions are imposed on the variables (fixed, free, bounds). For example in problem CORKSCRW some variables are fixed, some are free and some contain bounds. We also specify what kind of general constraints occur in the problem (equalities, inequalities, linear, nonlinear), and the characteristics of the objective function. The problem set has been chosen for its variety: it contains problems with negative curvature (e.g. OPTMASS), problems with ill-conditioned matrices of constraint gradients (e.g. HAGER4), problems containing only simple bounds (OBSTCLAE, TORSION1), problems with highly nonlinear equality constraints, and problems with a large number of variables and constraints. On the other hand our test set is small enough to allow us to know each problem well and analyze each run in detail.

Problem	# of var	# of constr	variable types	constraint types	objective
CORKSCRW	456	350	free, bounded, fixed	linear eq, nonlin ineq	nonlinear
COSHFUN	61	20	free	nonlin ineq	linear
DIXCHLNV	100	50	bounded	nonlin eq	nonlinear
GAUSSELM	14	11	free, bounded, fixed	linear ineq, nonlin eq	linear
HAGER4	2001	1000	free, bounded, fixed	linear eq	nonlinear
HIMMELBK	24	14	bounded	linear eq, nonlin eq	linear
NGONE	100	1273	bounded, fixed	linear ineq, nonlin ineq	nonlinear
OBSTCLAE	1024	0	bounded, fixed		nonlinear
OPTCNTRL	32	20	free, bounded, fixed	linear eq, nonlin eq	nonlinear
OPTMASS	1210	1005	free, fixed	linear eq, nonlin ineq	nonlinear
ORTHREGF	1205	400	free, bounded	nonlin eq	nonlinear
READING1	202	100	bounded, fixed	nonlin eq	nonlinear
SVANBERG	500	500	bounded	nonlin ineq	nonlinear
TORSION1	484	0	bounded, fixed		nonlinear

Table 1: The main test problem set.

In Table 2 we present the results for the primal-dual version of our new algorithm, NITRO. For comparison we also solved the problems with LANCELOT [17] using second derivatives and all its default settings. The runs of NITRO were terminated when $E(x_k, s_k; 0) \leq 10^{-7}$, and LANCELOT was stopped when the projected gradient and constraint violations were less than 10^{-7}; the termination criteria for these two methods are therefore very similar. In all these problems the two codes approached the same solution point. Since both algorithms use the conjugate gradient method to compute the step, we also report in Table 2 the total number of CG iterations needed for convergence. All runs were performed on a Sparcstation 20 with 32 MB of main memory, using a FORTRAN 77 compiler and double precision; the CPU time reported is in seconds. An asterisk indicates that the stopping test was not satisfied after 10,000 iterations. The results of NITRO reported in Table 2 are highly encouraging, particularly the number of function evaluations.

	# of	# of	f evals		CG iters		Time	
Problem	var	constr	NITRO	LAN	NITRO	LAN	NITRO	LAN
CORKSCRW	456	350	61	171	430	114780	53.78	657.94
COSHFUN	61	20	40	149	1316	3421	4.22	5.83
DIXCHLNV	100	50	19	1445	83	1431	14.46	153.97
GAUSSELM	14	11	52	28	115	112	0.79	0.25
HAGER4	2001	1000	18	14	281	2291	37.34	99.65
HIMMELBK	24	14	33	154	89	1533	4.15	8.18
NGONE	100	1273	256	3997	1821	129963	1027.51	1446.09
OBSTCLAE	1024	0	26	5	6184	366	566.39	12.98
OPTCNTRL	32	20	47	25	165	65	1.44	0.3
OPTMASS	1210	1005	39	*	151	*	24.79	*
ORTHREGF	1205	400	30	192	78	315	57.09	48.18
READING1	202	100	40	720	130	13981	130.89	74.13
SVANBERG	500	500	35	82	5067	3908	2720.19	120.96
TORSION1	484	0	19	8	2174	66	58.39	1.11

Table 2: Number of function evaluations, number of CG iterations and CPU time for the new primal-dual interior point method (NITRO) and LANCELOT (LAN). A * indicates that the method did not meet the stopping test in 10,000 iterations.

In Table 3 we compare the primal version of NITRO using (3.1) and the primal-dual version using (3.11). The column under the header "%full steps" denotes the percentage of steps that did not encounter the trust region (3.18). We see that the primal-dual version (pd) outperforms the primal version (p), and its step tends to be constrained by the trust region less often.

To observe whether the tangential subproblem becomes very difficult to solve as the barrier parameter approaches zero, we report in Table 4 the number of CG iterations required in the step computation during the *last* iteration of the interior point algorithm. At this stage the barrier parameter μ is of order 10^{-7}. Table 4 gives the number of CG iterations relative to the dimension $n - t$ of the linear system to be solved (recall that the code imposes a limit of 2 on this ratio). We also report if the step was inside the trust region (full), if it encountered the trust region (hit tr) or if the number of CG iterations reached the permissible limit of $2(n - t)$. These results, as well as an examination of the complete runs, indicate that the subproblems do not become particularly hard to solve as the problem approaches the solution. This is due to the preconditioning described before the statement of PCG Procedure.

To test the robustness of the new interior point method, we solved a set of problems from the Hock and Schittkowski collection [28], as programmed in CUTE. The results are given in Table 5, and include all the problems that we tested. Since these problems contain

Problem	# of var	# of constr	NITRO (pd) f evals	NITRO (pd) %full steps	NITRO (p) f evals	NITRO (p) %full steps
CORKSCRW	456	350	61	40	78	58
COSHFUN	61	20	40	83	472	6
DIXCHLNV	100	50	19	79	18	78
GAUSSELM	14	11	52	27	62	27
HAGER4	2001	1000	18	78	21	62
HIMMELBK	24	14	33	79	62	36
NGONE	100	1273	256	6	200	18
OBSTCLAE	1024	0	26	77	60	82
OPTCNTRL	32	20	47	92	51	73
OPTMASS	1210	1005	39	59	67	60
ORTHREGF	1205	400	30	30	31	30
READING1	202	100	40	78	33	33
SVANBERG	500	500	35	71	61	72
TORSION1	484	0	19	79	41	78

Table 3: Primal dual vs primal options of the new interior point method. The table gives the number of function evaluations and percentage of full steps. A * indicates that the stopping test was not satisfied in 10,000 iterations.

Problem	# of var	# of constr	NITRO (pd) CG iter	NITRO (pd) step type
CORKSCRW	456	350	0.03	full
COSHFUN	61	20	2.0	CG limit
DIXCHLNV	100	50	0.1	full
GAUSSELM	14	11	0.4	full
HAGER4	2001	1000	0.1	full
HIMMELBK	24	14	0.3	full
NGONE	100	1273	0.08	hit tr
OBSTCLAE	1024	0	2.0	CG limit
OPTCNTRL	32	20	0.3	full
OPTMASS	1210	1005	0.0	full
ORTHREGF	1205	400	0.006	full
READING1	202	100	0.03	full
SVANBERG	500	500	1.5	full
TORSION1	484	0	1.8	full

Table 4: Analysis of the last step computed by NITRO. Total number of CG iterations divided by the dimension of the linear system, $n - t$, and the type of step taken.

a very small number of variables, we do not report CPU time. NITRO failed for problems HS99 and HS109. In problem HS99, the code terminated very close to a solution because the trust region was too small. In problem HS109, the routine MA27 failed to factor the augmented systems in (3.56) and (3.57) because they were determined to be very close to singular. LANCELOT failed for four problems. In HS75, the code completed without reporting any errors. However, the point that was returned failed to satisfy the stopping test. In problems HS106, HS109, and HS116, LANCELOT was unable to compute a solution in 10,000 iterations.

It is reassuring to observe that NITRO failed on very few problems. Nevertheless its performance is not as good as that of LANCELOT on these small problems, and it appears that our strategy for decreasing the barrier parameter is overly conservative. We suspect that by decreasing it more rapidly, and in a carefully controlled manner [8], the number of function evaluations will be reduced significantly. We should also mention that we do not yet have a complete understanding of the behavior of NITRO on some of the problems on which it took a large number of iterations.

5. Final Remarks.

We have presented an interior point method for solving large nonlinear programming problems. Rather than trying to mimic primal-dual interior point methods for linear programming, we have taken the approach of developing a fairly standard SQP trust region method, and introduced in it some of the key features of primal-dual iterations. No attempt was made to obtain a rapidly convergent method: the barrier parameter was decreased at a linear rate, forcing the iterates of the algorithm to converge linearly. We have, however, given careful attention to the treatment of non-convexity, to the exploitation of sparsity though use of the conjugate gradient method and the sparse Cholesky code MA28, and have designed many features to make the algorithm robust on general problems. This approach appears to have paid off in that the algorithm has proved to be capable of solving a wide range of problems, even when ill-conditioning and non-convexity is present. Our tests seem to indicate that our code is competitive on large problems with a production code such as LANCELOT. We have also shown that the preconditioning of the tangential subproblem has, to a large extent, removed the effects of the ill-conditioning inherent in interior point methods, and that the CG iteration does not have particular difficulties in computing the tangential component of the step as the iterates approach the solution.

The algorithm presented here is not as rapidly convergent as it can be. We are currently developing [8] various mechanisms to accelerate the iteration; these include the use of higher-order corrections and rules for decreasing the barrier parameter at a superlinear rate. We should also note that the technique for refining the solution of linear systems referred to at the end of §3.4 is very conservative (in that it demands very tight accuracy) and leads to high execution times on some problems. More efficient techniques for refining the solution of linear systems are the subject of current investigation [26].

Acknowledgments. We would like to thank Guanghui Liu for help in the preparation of this article.

Problem	NITRO	LAN	Problem	NITRO	LAN
HS2	18	7	HS75	107	141**
HS3	12	5	HS77	17	22
HS4	11	2	HS78	5	12
HS7	8	18	HS79	6	10
HS10	17	19	HS80	13	15
HS11	14	19	HS81	13	17
HS13	40	81	HS83	36	26
HS14	14	13	HS84	20	60
HS16	15	16	HS85	1658	17
HS17	27	20	HS86	16	18
HS19	47	36	HS93	14	6
HS20	18	23	HS95	156	8
HS22	15	11	HS96	196	8
HS24	19	8	HS97	45	19
HS26	16	39	HS98	53	18
HS28	3	4	HS99	*	70
HS31	13	13	HS100	20	46
HS32	19	9	HS104	34	62
HS33	28	12	HS105	34	15
HS39	19	21	HS106	221	*
HS46	16	29	HS107	15	40
HS51	3	3	HS108	49	24
HS52	3	8	HS109	*	*
HS53	8	8	HS111	15	47
HS63	13	14	HS112	14	44
HS64	43	53	HS113	17	81
HS65	20	28	HS114	33	763
HS70	35	29	HS116	71	*
HS71	16	16	HS117	40	50
HS72	44	94	HS118	28	17
HS73	29	18	HS119	31	29
HS74	15	28			

Table 5: The number of function evaluations for NITRO and LANCELOT to solve a subset of the Hock and Schittkowski test collection. An asterisk (*) indicates a failure to obtain a solution within 10,000 iterations. A double asterisk indicates that LANCELOT computed a point that was not a local minimum.

6. *

References

[1] K.M. Anstreicher and J.P. Vial. On the convergence of an infeasible primal-dual interior point method for convex programming. *Optimization Methods and Software*, 3, pp. 273–283, 1994.

[2] M. Argaez. Exact and inexact Newton linesearch interior-point algorithms for nonlinear programming problems, TR97-13, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1997.

[3] P.T. Boggs and J.W. Tolle. Sequential Quadratic Programming, Acta Numerica 1995, Cambridge University Press, 1-51, 1995.

[4] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint. CUTE: Constrained and Unconstrained Testing Environment, *ACM Transactions on Mathematical Software*, 21, pp. 123-160, 1995.

[5] J.F. Bonnans and C. Pola. A trust region interior point algorithm for linearly constrained optimization, Rapport de recherche 1948, INRIA, France 1993.

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

[7] R.H. Byrd, J.C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. Report OTC 96/02, Optimization Technology Center, Northwestern University, 1996.

[8] R.H. Byrd, G. Liu and J. Nocedal (1997). "On the Local Behavior of an Interior Point Method for Nonlinear Programming", in Numerical Analysis 1997, D.F. Griffiths and D.J. Higham, eds, pp.37-56, Addison Wesley Longman.

[9] R.H. Byrd, R.B. Schnabel, and G.A. Shultz (1987). "A trust region algorithm for nonlinearly constrained optimization". *SIAM Journal on Numerical Analysis*, Vol. 24, pp. 1152–1170.

[10] M.R. Celis, J.E. Dennis and R.A. Tapia. A trust region strategy for nonlinear equality constrained optimization, in *Numerical Optimization 1984*, P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds., pp. 71–82, SIAM, 1985.

[11] R. M. Chamberlain, C. Lemaréchal, H. C. Pedersen and M. J. D. Powell. The watchdog technique for forcing convergence in algorithms for constrained optimization, *Mathematical Programming Studies*, 16, pp. 1-17, 1982.

[12] T. F. Coleman and A.R. Conn. On the local convergence of a quasi-Newton method for the nonlinear programming problem, *Mathematical Programming*, 21, pp. 755–769, 1984.

[13] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds, *SIAM Journal on Optimization*, 6 pp. 418-445, 1996.

[14] T. F. Coleman and Y. Li. On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds, *Mathematical Programming*, 67, pp. 189-224, 1994.

[15] A.R. Conn, N.I.M. Gould and Ph.L. Toint. A primal-dual algorithm for minimizing a non-convex function subject to bound and linear equality constraints, Report RC 20639, IBM T.J. Watson Research Center, Yorktown Heights, New York, 1997.

[16] A.R. Conn, N.I.M. Gould and Ph.L. Toint. A note on using alternative second-order models for the subproblems arising in barrier function methods for minimization, *Numer. Math.* 68, pp 17–33, 1994.

[17] A.R. Conn, N.I.M. Gould and Ph.L. Toint. *LANCELOT : A Fortran Package for Large-Scale Nonlinear Optimization* (Release A), Springer Series in Computational Mathematics, 17, Springer-Verlag, Berlin, 1992.

[18] J.E. Dennis, M. Heinkenschloss and L.N. Vicente. Trust-region interior-point algorithms for a class of nonlinear programming problems, *SIAM J. Control and Optimization*, 36, pp. 1750–1794, 1998.

[19] A.S. El-Bakry, R.A. Tapia, T. Tsuchiya, and Y. Zhang. On the formulation and theory of the Newton interior-point method for nonlinear programming. *J. Optim. Theory. Appl.*, 89, pp. 507–5451, 1996.

[20] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Wiley & Sons, 1968.

[21] R. Fletcher. *Practical Methods of Optimization, Second Edition*, John Wiley & Sons, New York, 1990

[22] A. Forsgren and P.E. Gill. Primal-dual interior methods for nonconvex nonlinear programming, *SIAM Journal on Optimization* 8, pp. 1132-1152, 1998.

[23] D.M. Gay, M.L. Overton and M.H. Wright. A primal-dual interior point method for non-convex nonlinearly constrained optimization, in *Advances in Nonlinear Programming*, Y. Yuan, ed., pp. 31-56, Kluwer Academic Publishers, Dordrecht, 1998.

[24] P.E. Gill, W. Murray, and M.A. Saunders. An SQP algorithm for large-scale optimization. Report NA 97-2, Department of Mathematics, University of California, San Diego, 1997.

[25] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*, Academic Press, Inc., 1981.

[26] N.I.M. Gould, M.E. Hribar and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization, to appear.

[27] Harwell Subroutine Library, A catalogue of subroutines (release 12), AEA Technology, Harwell, Oxfordshire, England, 1995.

[28] W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, 187, Springer-Verlag, 1981.

[29] M.E. Hribar. Large-scale constrained optimization, Ph.D. Dissertation, EECS, Department, Northwestern University, 1996.

[30] F. Jarre and S.J. Wright. On the role of the objective function in barrier methods. Technical Report MCS-P485-1294, MCS Division, Argonne National Laboratory, 1994.

[31] M. Lalee, J. Nocedal, and T. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM J. Optimization*, 8,3, pp. 682-706, 1998.

[32] N. Maratos. Exact penalty function algorithms for finite dimensional and control optimization problems, Ph.D. Dissertation, University of London, 1978.

[33] R.D.C. Monteiro and Y. Wang. Trust region affine scaling algorithms for linearly constrained convex and concave programs, (1998), *Mathematical Programming*, 80, pp. 283-313.

[34] E. Omojokun. Trust region algorithms for optimization with nonlinear equality and inequality constraints, Ph.D. Dissertation, Dept. of Computer Science, University of Colorado, 1989.

[35] Z. Parada. A Modified Augmented Lagrangian Merit Function and Q-Superlinear Characterization Results for Primal-Dual Quasi-Newton Interior-Point Methods for Nonlinear Programming, TR97-12, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1997.

[36] T. Plantenga. Large-scale nonlinear constrained optimization using trust regions. PhD thesis, EECS Department, Northwestern University, 1994.

[37] M.J.D. Powell. A hybrid method for nonlinear equations, in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Gordon and Breach, London, 1970.

[38] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal*, 20, pp. 626-637, 1983.

[39] R.J. Vanderbei. *Linear Programming*, Kluwer, 1996.

[40] A. Vardi. A trust region algorithm for equality constrained minimization: convergence properties and implementation. *SIAM Journal on Numerical Analysis*, 22, pp. 575–591, 1985.

[41] M. Wright. Why a pure primal Newton barrier step may be infeasible. *SIAM J. Optimization*, 5, pp. 1–12, 1995.

[42] S.J. Wright. *Primal-Dual Interior Point Methods*, SIAM, 1997.

[43] H. Yamashita. A globally convergent primal-dual interior point method for constrained optimization. Technical Report, Mathematical Systems Institute Inc., Tokyo, Japan, 1994.

[44] H. Yamashita and H. Yabe. Superlinear and quadratic convergence of some primal-dual interior point methods for constrained optimization, *Mathematical Programming*, 75, pp. 377–397, 1996.