KNITRO-Direct: A Hybrid Interior Algorithm for Nonlinear Optimization

R.A. Waltz* J.L. Morales† J. Nocedal* D. Orban*

August 8, 2003

Abstract

A hybrid interior-point method for nonlinear programming is presented. It enjoys the flexibility of switching between a line search based method which computes steps by factoring the primal-dual equations and an iterative method using a conjugate gradient algorithm and globalized by means of trust regions. Steps computed by a direct factorization are always tried first, but if they are deemed to be ineffective, a trust region iteration that guarantees progress toward stationarity is invoked. To demonstrate its effectiveness, the algorithm is implemented in the KNITRO [4, 28] software package and extensively tested on a selection of problems from the CUTEr [2, 19] test set.

1 Introduction

In this paper we describe an interior method for nonlinear programming and discuss its software implementation and numerical performance. A typical iteration computes a primary step by solving the primal-dual equations (using direct linear algebra) and performs a line search to ensure decrease in a merit function. In order to obtain global convergence in the presence of nonconvexity and Hessian or Jacobian singularities, the primary step is replaced, under certain circumstances, by a safeguarding trust region step. The algorithm can use second derivatives of the objective function and constraints, or can approximate them using quasi-Newton updating.

*Department of Electrical and Computer Engineering, Northwestern University. These authors were supported by National Science Foundation grants CCR-9987818, ATM-0086579 and CCR-0219438 and Department of Energy grant DE-FG02-87ER25047-A004.

†Departamento de Matemáticas, ITAM, Mexico City. This author was supported by Asociación Mexicana de Cultura, A.C.

The motivation for this paper is to widen the class of problems that can be efficiently solved by the KNITRO software package [28]. The algorithm implemented in the first release of KNITRO [4] is a trust region method, and uses a null-space decomposition and a projected conjugate gradient iteration to compute the step. This iterative approach has the advantage that the Hessian of the Lagrangian need not be formed or factored, which is effective for many large problems. The disadvantage is that the projected conjugate gradient iteration can be expensive when the Hessian of the Lagrangian is ill-conditioned, in which case it is preferable to use direct linear algebra techniques to compute the step.

By designing the new hybrid method so that it computes steps using direct linear algebra, whenever the quality of these steps can be guaranteed, and falling back on a trust region step otherwise, we can easily implement the new method within the KNITRO package. Moreover, since the hybrid algorithm can reduce to a pure line search or a pure trust region approach, we are able to implement two different interior methods in a unified algorithmic and software framework.

The problem under consideration has the form

$$\min_x \quad f(x) \tag{1.1a}$$

$$\text{s.t.} \quad h(x) = 0 \tag{1.1b}$$

$$g(x) \leq 0, \tag{1.1c}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^l$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions. A variety of line search interior algorithms have been proposed to date [4, 10, 14, 15, 25, 26, 30] and several of them have been implemented in high quality software packages [1, 25, 26]. The search direction is computed in these algorithms by factoring the primal-dual system. In order to achieve robustness, these line search approaches must successfully address two questions:

- How to define the search direction when the quadratic model used by the algorithm is not convex;
- How to handle rank deficiency (and near deficiency) of the Hessian of the Lagrangian and constraint Jacobian.

The first question is often addressed by adding a multiple of the identity matrix to the Hessian of the Lagrangian in order to convexify the model. This strategy was first shown to be effective in the context of nonlinear interior methods by the LOQO software package [25]. The second issue is handled

differently in each software implementation. The difficulties caused by singular and nearly singular constraint Jacobians are sometimes addressed at the linear algebra level by introducing perturbations during the factorization of the KKT matrix [1, 25]. Alternatives include the use of ℓ_1 or ℓ_2 penalizations of the constraints which provide regularization [14, 16, 24], or by invoking a feasibility restoration phase [13, 26].

In this paper we describe a mechanism for stabilizing the line search iteration that is different from those proposed in the literature. It consists of falling back, under certain conditions, on a trust region step that is guaranteed to make progress toward feasibility and optimality. A challenge is to design the algorithm so that there is a smooth transition between line search and trust region steps. We will argue that the hybrid method presented in this paper is not more expensive than alternative approaches, has favorable convergence properties and performs well on standard test problems.

Notation. Throughout the paper $\|\cdot\|$ denotes the Euclidean norm.

2 Outline of the Algorithm

We will consider an interior method that replaces the nonlinear program (1.1) by a sequence of barrier subproblems of the form

$$\min_z \quad \varphi_\mu(z) \equiv f(x) - \mu \sum_{i=1}^{m} \ln s_i \tag{2.1a}$$

$$\text{s.t.} \quad h(x) = 0 \tag{2.1b}$$

$$g(x) + s = 0. \tag{2.1c}$$

Here $s > 0$ is a vector of slack variables, $z = (x, s)$ and $\mu > 0$ is the barrier parameter. The Lagrangian function associated with (2.1) is defined by

$$\mathcal{L}(z, \lambda; \mu) = \varphi_\mu(z) + \lambda_h^T h(x) + \lambda_g^T (g(x) + s), \tag{2.2}$$

where $\lambda_h \in \mathbb{R}^l$ and $\lambda_g \in \mathbb{R}^m$ are Lagrange multipliers. The first-order optimality conditions for the barrier problem (2.1) can be written as

$$\begin{bmatrix} \nabla f(x) + A_h(x)^T \lambda_h + A_g(x)^T \lambda_g \\ S\Lambda_g e - \mu e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{2.3}$$

together with (2.1b) and (2.1c). Here S and Λ_g denote diagonal matrices whose diagonal entries are given by the vectors s and λ_g, respectively, and A_h and A_g are the Jacobian matrices of h and g.

Applying Newton's method to the system (2.3), (2.1b), (2.1c), from the current iterate (z, λ), where $\lambda = (\lambda_h, \lambda_g)$, results in the primal-dual system

$$\begin{bmatrix} W(z,\lambda;\mu) & A(x)^T \\ A(x) & 0 \end{bmatrix} \begin{bmatrix} d_z \\ d_\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_z \mathcal{L}(z,\lambda;\mu) \\ c(z) \end{bmatrix}, \tag{2.4}$$

where we have defined

$$d_z = \begin{bmatrix} d_x \\ d_s \end{bmatrix}, \quad d_\lambda = \begin{bmatrix} d_h \\ d_g \end{bmatrix}, \quad c(z) = \begin{bmatrix} h(x) \\ g(x) + s \end{bmatrix} \tag{2.5}$$

and

$$A(x) = \begin{bmatrix} A_h(x) & 0 \\ A_g(x) & I \end{bmatrix}, \quad W(z,\lambda;\mu) = \begin{bmatrix} \nabla^2_{xx} \mathcal{L}(z,\lambda;\mu) & 0 \\ 0 & S^{-1}\Lambda_g \end{bmatrix}. \tag{2.6}$$

The new iterate is given by

$$z^+ = z + \alpha_z d_z, \quad \lambda^+ = \lambda + \alpha_\lambda d_\lambda. \tag{2.7}$$

The steplengths α_z and α_λ are computed in two stages. First we determine the largest positive scalars γ_z and γ_λ such that

$$s + \gamma_z d_s \geq (1-\tau)s \tag{2.8a}$$

$$\lambda_g + \gamma_\lambda d_g \geq (1-\tau)\lambda_g, \tag{2.8b}$$

with $\tau \approx 1$. (In our tests we use $\tau = .995$.) The algorithm then performs a backtracking line search that computes a steplength $\alpha_z \leq \min\{1, \gamma_z\}$ providing sufficient decrease in a merit function (to be defined below). The procedure for updating the barrier parameter μ will be described in section 3.5.

The iteration (2.4)-(2.7) provides the basis for most line search interior methods. This remarkably simple approach must, however, be modified to cope with non-convexity and to prevent convergence to non-stationary points [5, 27]. Instead of modifying the primal-dual matrix, as is commonly done, we use a safeguarding trust region step to stabilize the iteration, for two reasons. First, in the negative curvature case, modifying the Hessian $\nabla^2 \mathcal{L}(z,\lambda;\mu)$ may introduce undesirable distortions in the model and can also require several factorizations of the primal-dual system. We prefer to compute a descent direction using a null space approach in which the tangential component of the step is obtained by a projected Krylov iteration, as is done with conjugate gradients in the KNITRO package [28] and using a Lanczos method in the GALAHAD package [18].

Second, we would like to take advantage of the robustness of trust region steps in the presence of Hessian or Jacobian singularities. We have in mind trust region methods that provide Cauchy decrease for both feasibility and optimality at every iteration. Since it is known that, when line search iterations converge to non-stationary points, the steplengths α_z or α_λ in (2.7) converge to zero, we will monitor these steplengths. If they become smaller than a given threshold, we discard the line search iteration (2.4)-(2.7) and replace it with a trust region step. The resulting hybrid algorithm will possess global convergence properties similar to those of the algorithms implemented in FILTERSQP [13] and KNITRO [3].

We outline the method in Algorithm 2.1. Here $\phi_\nu(z)$ denotes a merit function using a penalty parameter ν, and $D\phi_\nu(z; d_z)$ denotes the directional derivative of ϕ_ν along a direction d_z.

In our tests we choose $\eta = 10^{-8}$, $\delta = 10^{-5}$, and in section 3.2 we describe how α_{min} is chosen, indirectly. The initial multipliers λ_0 are computed by least squares. When the line search step is discarded (the last If-Endif block) we compute one or more trust region steps until one of them provides sufficient reduction in the merit function.

One could consider an algorithm that, instead of switching between two methods, would follow a dog-leg type approach. Here Cauchy and Newton steps would be computed at every iteration and the algorithm would move along a direction in the span of these two steps. We have not followed such a method for two reasons. First, computing a Cauchy step that ensures progress toward feasibility and optimality requires the factorization of a system different from the primal-dual matrix [3, 30]. Hence computing Cauchy and Newton steps at every iteration would be too expensive, and this suggests using Cauchy steps only when needed. Second, a dog-leg approach is not well defined in the case of negative curvature, in which case a Newton-CG iteration [23] is more appropriate. These observations and the fact that the first release of KNITRO implements a Newton-CG iteration motivated us to follow the approach just outlined.

Many details of Algorithm 2.1 have not been specified and will be discussed next.

3 Complete Description of the Algorithm

In this section we will discuss in detail all aspects of the algorithm, with the exception of the safeguarding trust region steps which are described in [4] and are computed in practice with KNITRO. Therefore throughout most of

Algorithm 2.1 Outline of Hybrid Interior Algorithm
Choose $z_0 = (x_0, s_0)$, *and the parameters* $0 < \eta$, $0 < \delta < 1$ *and* $0 < \alpha_{min} < 1$. *Compute initial values for the multipliers* λ_0, *the trust-region radius* $\Delta_0 > 0$, *and the barrier parameter* $\mu_0 > 0$. *Set* $k = 0$.

Repeat *until a stopping test for the nonlinear program (1.1) is satisfied:*
  Repeat *until a termination test for the barrier problem (2.1) is satisfied:*
    Factor the primal-dual system (2.4) and record the number `neig` *of negative eigenvalues of its coefficient matrix.*
    Set `LineSearch = False`.
    If `neig` $\leq l + m$
      Solve (2.4) to obtain the search direction $d = (d_z, d_\lambda)$.
      Compute $\gamma_{\min} = \min\{\gamma_z, \gamma_\lambda\}$ *using* (2.8*a*),(2.8*b*).
      If $\gamma_{\min} > \delta$,
        Compute a steplength $\alpha_k \leq 1$ *such that*
        $\phi_\nu(z + \alpha_k \gamma_z d_z) \leq \phi_\nu(z) + \eta \alpha_k \gamma_z D\phi_\nu(z; d_z)$.
        If $\alpha_k > \alpha_{min}$,
          Set $\alpha_z = \alpha_k \gamma_z$, $\alpha_\lambda = \alpha_k \gamma_\lambda$.
          Set z_{k+1}, λ_{k+1} *using* (2.7).
          Compute Δ_{k+1} *and set* `LineSearch = True`.
        Endif
      Endif
    Endif
    If `LineSearch == False`,
      Compute (z_{k+1}, λ_{k+1}) *using a globally convergent safeguarding trust region method.*
      Compute Δ_{k+1}.
    Endif
    Set $k \leftarrow k + 1$.
  End
  Choose a new barrier parameter $\mu_{k+1} < \mu_k$.
End

this section we will focus mainly on the line search steps.

3.1 Merit Function

The merit function is defined by

$$\phi_\nu(z) = \varphi_\mu(z) + \nu\|c(z)\|, \tag{3.9}$$

where φ_μ is the barrier function defined in (2.1a), the constraints $c(z)$ are given by (2.5) and $\nu > 0$ is the penalty parameter which is updated at each iteration so that the search direction d given by (2.4) is a descent direction for ϕ_ν.

Our update rule for ν, proposed in [11], is inspired by trust region methods. Instead of requiring only that the directional derivative of ϕ_ν be negative, as is commonly done, we will choose ν based on a quadratic/linear model of the merit function. A step d is acceptable in trust region methods if the ratio of actual to predicted reduction of the merit function is greater than a given constant $\eta > 0$, i.e.,

$$\frac{ared(d)}{pred(d)} > \eta.$$

Here we define

$$ared(d) = \phi_\nu(z) - \phi_\nu(z + d_z)$$

and

$$pred(d) = -\nabla\varphi_\mu(z)^T d_z - \frac{\sigma}{2} d_z^T W d_z + \nu\left(\|c(z)\| - \|c(z) + A(x)d_z\|\right), \tag{3.10}$$

with

$$\sigma = \begin{cases} 1 & \text{if } d_z^T W d_z > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Here W stands for $W(z, \lambda; \mu)$. If $\sigma = 1$, $pred$ is a standard quadratic/linear model of the merit function used in a variety of trust region methods. We allow σ to have the value zero because, as we argue below, including the term $d_z^T W d_z$ in (3.10) when it is negative could cause the algorithm to fail.

Following [4, 11] we choose the penalty parameter ν so that

$$pred(d) \geq \rho\nu\|c(z)\|, \tag{3.12}$$

for some parameter $\rho \in (0, 1)$. (In our tests we use the value $\rho = 0.1$.) From (3.10), (3.12) and the fact that the step d_z computed from (2.4) satisfies $c(z) + A(z)d_z = 0$, we get that

$$\nu \geq \frac{\nabla\varphi_\mu(z)^T d_z + \frac{\sigma}{2} d_z^T W d_z}{(1-\rho)\|c(z)\|} \equiv \nu_{TRIAL}. \tag{3.13}$$

The update rule for the penalty parameter ν is:

$$\nu^+ = \begin{cases} \nu & \text{if } \nu \geq \nu_{TRIAL} \\ \nu_{TRIAL} + 1 & \text{otherwise.} \end{cases} \tag{3.14}$$

To see that this choice of ν guarantees that d_z is a descent direction for ϕ_ν, we note that (3.13) and the definition of σ imply that

$$\nabla\varphi_\mu^T d_z - \nu\|c(z)\| \leq -\rho\nu\|c(z)\|.$$

Since the directional derivative of ϕ_ν along d_z is given by

$$D\phi_\nu(z; d_z) = \nabla\varphi_\mu^T d_z - \nu\|c(z)\| \tag{3.15}$$

(see for example [21, p.545]) we have that

$$D\phi_\nu(z; d_z) \leq -\rho\nu\|c(z)\|. \tag{3.16}$$

Note, however, that this argument would not hold if σ were defined as 1 when $d_z^T W d_z < 0$, for then d_z might not be a descent direction for the merit function. (When $c(z) = 0$, we can set $\nu^+ = \nu$ because it is easy to show that in this case $D\phi_\nu(z; d) < 0$.)

We have experimented with other update procedures for ν which directly impose (3.16). One of them is given by (3.13) but with σ always equal to 0 (see for example [7]). We have observed that the rule described in this section, which may include curvature information about the Lagrangian, gives consistently better results. Note that including the term $d_z^T W d_z$ in (3.13) leads to larger estimates of the penalty parameter than when this term is not included.

3.2 Line Search and Second Order Correction

After computing the primal-dual direction (d_z, d_λ) defined by (2.4) we determine the steplengths γ_z and γ_λ satisfying the inequalities (2.8a)-(2.8b). If both are greater than the threshold δ, then we perform a backtracking line search that generates a series of steplengths $\alpha^0 > \alpha^1 > \alpha^2 > ...$ until one of the following two conditions is satisfied:

1. One of the steplengths $\alpha^i < \alpha_{min}$, in which case the line search is aborted, the primal-dual direction (d_z, d_λ) is discarded, and the algorithm invokes the safeguarding trust region method;

2. A steplength α^j satisfies the Armijo condition

$$\phi_\nu(z + \alpha^j \gamma_z d_z) \leq \phi_\nu(z) + \eta \alpha^j \gamma_z D\phi_\nu(z; d_z). \tag{3.17}$$

In this case, the line search terminates successfully, and we define $\alpha_k = \alpha^j$ and set $\alpha_z = \alpha_k \gamma_z$, $\alpha_\lambda = \alpha_k \gamma_\lambda$.

The steplengths α^j are computed as follows. First, $\alpha^0 = 1$. If the previous iteration was a line search iteration, we set $\alpha^1 = 1/2$, otherwise, if it was a trust region iteration, we set

$$\alpha^1 = \min\left(\frac{1}{2}, \frac{\Delta}{\|d_z\|}\right),$$

where Δ is the current trust region radius. For $j \geq 1$,

$$\alpha^{j+1} = \frac{1}{2}\alpha^j.$$

In our practical implementation, instead of terminating the line search when $\alpha^j \leq \alpha_{min}$, we terminate if $j = 3$.

The merit function (3.9), may reject steps that make good progress towards the solution, a phenomenon known as the Maratos effect. This deficiency can be overcome by applying a second-order correction step (SOC) which is a Newton-like step that aims to improve feasibility [12].

We apply the second order correction when the first trial steplength $\alpha^0 = 1$ is rejected, and if the reason for the rejection can be attributed solely to an increase in the norm of the constraints, i.e., the second term in (3.9). More specifically, if $\alpha^0 \gamma_z$ does not satisfy the Armijo condition (3.17) and the barrier objective function $\varphi_\mu(z)$ did not increase, then before computing a shorter steplength we attempt a second-order correction step $d^{\text{soc}} = (d_z^{\text{soc}}, d_\lambda^{\text{soc}})$ which is the solution of

$$\begin{bmatrix} W & A(x)^T \\ A(x) & 0 \end{bmatrix} \begin{bmatrix} d_z^{\text{soc}} \\ d_\lambda^{\text{soc}} \end{bmatrix} = \begin{bmatrix} -\nabla\varphi_\mu(x) - \gamma_z W d_z - A(x)^T[\lambda + \gamma_\lambda d_\lambda] \\ -c(z + \gamma_z d_z) \end{bmatrix}, \tag{3.18}$$

where $W = W(z, \lambda; \mu)$. Using it we define

$$\bar{d}_z = \gamma_z d_z + d_z^{\text{soc}}, \quad \bar{d}_\lambda = \gamma_\lambda d_\lambda + d_\lambda^{\text{soc}}, \tag{3.19}$$

where $\bar{d}_z = (\bar{d}_x, \bar{d}_s)$ and $\bar{d}_\lambda = (\bar{d}_h, \bar{d}_g)$. We determine the largest positive scalars γ_z^{soc} and $\gamma_\lambda^{\text{soc}}$ such that

$$s + \gamma_z^{\text{soc}} \bar{d}_s \geq (1 - \tau)s \tag{3.20a}$$

$$\lambda_g + \gamma_\lambda^{\text{soc}} \bar{d}_g \geq (1 - \tau)\lambda_g, \tag{3.20b}$$

and define

$$z^{\text{soc}} = z + \gamma_z^{\text{soc}} \bar{d}_z, \quad \lambda^{\text{soc}} = \lambda + \gamma_\lambda^{\text{soc}} \bar{d}_\lambda.$$

If $\phi_\nu(z^{\text{soc}}) < \phi_\nu(z)$ then we accept the second-order correction step. Otherwise, we discard d^{soc} and continue the backtracking line search along the primal-dual step (d_z, d_λ) by selecting a new steplength α^1.

Note that in the left-hand side of (3.18), all quantities are unchanged when compared to the system from which d itself was computed—they are not evaluated at $z + \gamma_z d_z$. The right-hand side does not require any additional constraint evaluation since $c(z+\gamma_z d_z)$ has been evaluated to measure the desirability of γd. Hence, the computational cost of the SOC step defined by (3.18) is a linear system solve with a different right-hand side. Since the factorization of the coefficient matrix has been performed during the computation of d, the total cost of the SOC step is one forward solve, and one backsolve. It should be noted however that the total step z^{soc} will have to be tested for acceptance, resulting in an additional function evaluation.

3.3 Slack resets.

Given the form (3.9) of the merit function ϕ_ν and the fact that for fixed $\mu > 0$, the function $-\mu \ln s_i$ is a decreasing function of s_i, Algorithm 2.1 resets certain slack components after each trial step in order to increase the odds for the step to be accepted by the merit function, while at the same time improving the satisfaction of the constraints. More precisely, before the trial step $z^+ = z + \alpha_z d_z$ is tested for acceptance by the merit function, the slack components $s_i^+ = s_i + \alpha_z [d_s]_i$ for which the inequality $-g_i(x^+) > s_i^+$ is satisfied are reset to the value

$$s_i^+ = -g_i(x^+). \tag{3.21}$$

Thus (3.21) increases the slacks, thereby decreasing further the barrier term in (3.9) and improving the satisfaction of the constraints (2.1c), thus decreasing the penalty term $\|c(z)\|$. Hence, this reset may only cause the merit function to decrease further and thus promote step acceptance.

3.4 NLP Stopping Test

Ideally, the stopping test should be invariant under the scaling of the variables, the objective and constraints. It is difficult, however, to achieve complete scale invariance, and the following tests attempt to achieve a balance between practicality and scale invariance.

The stopping tolerances ϵ^{opt} and ϵ^{feas} are given. (In our tests they are both set to 10^{-6}.) The algorithm terminates if an iterate (x, s, λ) satisfies

$$\begin{aligned} \|\nabla f(x) + A_h(x)^T\lambda_h + A_g(x)^T\lambda_g\|_\infty &\leq \max\{1, \|\nabla f(x)\|_\infty\}\epsilon^{opt} && (3.22)\\ \|S\lambda_g\|_\infty &\leq \max\{1, \|\nabla f(x)\|_\infty\}\epsilon^{opt} && (3.23)\\ \|(h(x), g(x)^+)\|_\infty &\leq \max\{1, \|(h(x_0), g(x_0)^+)\|_\infty\}\epsilon^{feas}, && (3.24) \end{aligned}$$

where $g(x)^+ = \max\{0, g(x)\}$ and x_0 is the starting point. The scaling factor in (3.22) makes this test invariant to scalings in f, c and to linear changes of the variable x. The test for complementarity (3.23) is based on the fact that the scale of s is dependent on the scale of c, and the magnitude of λ_g is proportional to $\|\nabla f\|/\|A\|$. Thus (3.23) is invariant to scaling of f and c. (A more complex scaling factor employing $\|A\|$ would make it invariant to linear changes in the variables, but we use the same scaling factors in (3.22)-(3.23) for simplicity.

The scale factor for feasibility is difficult to choose. If the constraints were linear, then $\|(h(0), g(0)^+)\|_\infty$ is an appropriate normalization factor since it measures the magnitude of the right hand side vectors of all constraints. But since $h(0)$ or $g(0)$ may not be defined in some problems, we opted for using the initial point, x_0.

3.5 Update of μ and Barrier Stopping Test

The sequence of barrier parameters $\{\mu_k\}$ must converge to zero, and should do so quickly if possible. Superlinear rules for decreasing μ have been studied in [9, 17, 22, 29], but they employ various parameters which can be difficult to select in practice. We use, instead the following simple strategy for updating μ that has performed as well in our tests as more complicated superlinear rules.

If the most recent barrier problem was solved in less than three iterations, we set

$$\mu_{k+1} = \mu_k/100;$$

otherwise

$$\mu_{k+1} = \mu_k/5.$$

Let us now discuss the termination test for the barrier problem. It is our experience that the choice of this stopping test has significant impact in the efficiency and robustness of interior methods. For the current value of

μ, we choose tolerances ϵ_μ^{opt} and ϵ_μ^{feas} (to be defined below) and impose the barrier stopping tests:

$$\begin{aligned} \|\nabla f(x) + A_h(x)^T\lambda_h + A_g(x)^T\lambda_g\|_\infty &\leq \max\{1, \|\nabla f(x)\|_\infty\}\epsilon_\mu^{opt} \quad (3.25) \\ \|S\lambda_g - \mu e\|_\infty &\leq \max\{1, \|\nabla f(x)\|_\infty\}\epsilon_\mu^{opt} \quad (3.26) \\ \|(h(x), g(x) + s)\|_\infty &\leq \max\{1, \|(h(x_0), g(x_0)^+\|_\infty\}\epsilon_\mu^{feas}. \quad (3.27) \end{aligned}$$

Note that the scaling factors are identical to those used for the NLP stopping test (3.22)-(3.24), and that the left hand sides of these two tests differ only in the use of the additional term $-\mu e$ in (3.26) and the use of $g(x)+s$ rather than $g(x)^+$ in (3.27) .

As we shall see, the tolerances ϵ_μ^{opt} and ϵ_μ^{feas} will be chosen to be proportional to μ most of the time, and thus become tighter as μ decreases. We want to avoid letting μ, ϵ_μ^{opt} and ϵ_μ^{feas} take unnecessarily small values because this can have detrimental effects on the algorithm. Therefore we determine the values of μ, ϵ_μ^{opt} and ϵ_μ^{feas} for which satisfaction of the barrier stopping test automatically implies satisfaction of the NLP stopping test.

Since the scaling factors and left hand sides for (3.22) and (3.25) are identical we have that any point which satisfies (3.25) automatically satisfies (3.22) for all values $\epsilon_\mu^{opt} \leq \epsilon^{opt}$. Also, from (3.26) we have that for a given complementary pair $s_i[\lambda_g]_i$ which satisfies the barrier stopping test,

$$s_i[\lambda_g]_i \leq \mu + \max\{1, \|\nabla f(x)\|_\infty\}\epsilon_\mu^{opt}$$

and in order to also satisfy the NLP stopping test (3.23) it must satisfy

$$s_i[\lambda_g]_i \leq \max\{1, \|\nabla f(x)\|_\infty\}\epsilon^{opt}.$$

Therefore an iterate (x, s) which satisfies (3.26) also satisfies (3.23) as long as

$$\epsilon_\mu^{opt} \leq \epsilon^{opt} - \mu/(\max\{1, \|\nabla f(x)\|_\infty\}). \quad (3.28)$$

For simplicity we remove the scaling factor, and it follows that a point which satisfies the barrier KKT stopping conditions (3.25),(3.26) also satisfies the NLP KKT stopping conditions (3.22),(3.23) for all values of ϵ_μ^{opt} which satisfy

$$\epsilon_\mu^{opt} \leq \epsilon^{opt} - \mu. \quad (3.29)$$

For feasibility, from the slack reset scheme (3.21) we have that $g(x) + s \geq 0$, and therefore since $s > 0$,

$$\|g(x)^+\|_\infty \leq \|g(x) + s\|_\infty.$$

Therefore from (3.24) and (3.27) it follows that for all

$$\epsilon_\mu^{feas} \leq \epsilon^{feas} \tag{3.30}$$

the NLP feasibility test (3.24) will be satisfied, if the barrier feasibility test (3.27) is satisfied.

Subject to the minimum values given by (3.29)-(3.30) we set ϵ_μ^{opt} and ϵ_μ^{feas} equal to $\theta\mu$, where θ is a fixed algorithm parameter (currently $\theta = 1$). Based on this then, the barrier stopping tolerances are determined by the following formulas.

$$\epsilon_\mu^{opt} = \max\{\theta\mu, \epsilon^{opt} - \mu\} \tag{3.31}$$

$$\epsilon_\mu^{feas} = \max\{\theta\mu, \epsilon^{feas}\}. \tag{3.32}$$

This ensures that we do not oversolve the barrier subproblem but it does not place any lower bound on the barrier parameter μ. Although an overly small value of μ will not cause us to oversolve the barrier subproblem because of the limits on the barrier tolerances established above, we still want to prevent μ from becoming unnecessarily small since an overly small value of μ could have a negative effect on the step computation.

From (3.31), (3.32) we have that satisfaction of the barrier stopping test will imply satisfaction of the NLP test if $\mu \leq \min\{\epsilon^{opt}/(1+\theta), \epsilon^{feas}/\theta\}$. Currently we have $\theta = 1$, but to provide a margin of safety we enforce a minimum value of μ based on the NLP tolerance

$$\mu_{min} = \frac{\min\{\epsilon^{opt}, \epsilon^{feas}\}}{100}. \tag{3.33}$$

3.6 Transition to and from Safeguarding Steps

If the line search step is not acceptable, the safeguarding trust region algorithm is invoked. It is efficient to provide this algorithm with a trust region radius that reflects current problem information. In particular, we are concerned with the case when two safeguarding trust region steps are separated by a long sequence of line search steps, and with preserving the global convergence properties of the trust region iteration.

Therefore, if the most recent step was an acceptable line search step, d_k, we set

$$\Delta_{k+1} = 2\|d_k\| \tag{3.34}$$

in order to keep the trust region information up-to-date. Otherwise, if the most recent step was either a trust region step or a rejected line search step,

the trust region radius is updated according to standard trust region update rules. In our implementation we follow the update strategy given in [4].

We note that, in addition to using the trust region for the safeguarding steps, the trust region radius may also play a role in the backtracking line search as explained in section 3.2. Also, in order to try to avoid the repeated computation of unsuccessful line search steps, if a trust region iteration is rejected, subsequent iterations will be computed with the trust region method until a successful step is obtained.

3.7 Solution of Primal-Dual Equations

The primal-dual system (2.4) is solved using the symmetric indefinite factorization implemented in the HSL library routine `MA27` [20]. This routine provides the inertia of the primal-dual system, and in particular the number of negative eigenvalues, `neig`. An important practical consideration is the choice of the pivot tolerance. We set it initially to 10^{-8}. If `MA27` returns a message of singularity we increase it by a factor of 10; this process is repeated if necessary until the maximum pivot tolerance of 0.1 is reached. The use of a very small pivot tolerance can yield significant savings in computing time.

We also make use of iterative refinement on occasion. If the norm of the relative residual of the primal-dual system provided by the computed solution is greater than some threshold, we may apply a maximum of five iterative refinement steps to try to decrease this residual and improve the solution.

3.8 Quasi-Newton Approximations

The algorithm includes several quasi-Newton options for problems in which second derivatives cannot be computed, or when it is too expensive to do so. A quasi-Newton version of the primal-dual step is obtained by replacing $\nabla^2_{xx}\mathcal{L}$ in (2.6) by a quasi-Newton approximation B. Our algorithm implements dense BFGS and SR1 methods as well as a limited memory BFGS method.

In all these methods we initialize B to be the identity matrix, and the correction pairs $(y, \Delta x)$ required by the quasi-Newton updating formulae are obtained as follows. After computing a step from (z, λ) to (z^+, λ^+), we define

$$\begin{aligned} y &= \nabla_x \mathcal{L}(x^+, \lambda^+) - \nabla_x \mathcal{L}(x, \lambda^+) \\ \Delta x &= x^+ - x. \end{aligned}$$

To ensure that the BFGS method generates a positive definite matrix, the update is skipped if $y^T \Delta x \leq 0$. (A damping procedure is also an option.) SR1 updating is safeguarded using standard rules; see e.g. [21, §8.2].

For large problems it is desirable to use limited memory updating to avoid the storage and manipulation of a dense $n \times n$ matrix. We have implemented a limited memory BFGS method using the compact representations described in [6]. Here B has the form

$$B = \xi I + W M W^T, \tag{3.35}$$

where $\xi > 0$ is a scaling factor, W is an $n \times 2p$ matrix, M is a $2p \times 2p$ symmetric and nonsingular matrix, and p denotes the number of correction pairs saved in the limited memory updating procedure. The matrices W and M are formed using the vectors $\{\Delta x^k\}$ and $\{y^k\}$ accumulated in the last p iterations.

Since the limited memory matrix B is positive definite and A has full rank (otherwise we revert to the trust region step), the primal-dual matrix is nonsingular, and we can compute the solution to (2.4) by formally inverting the coefficient matrix. Before doing so we decompose B, as the sum of the low-rank term WMW^T and the multiple of the identity matrix,

$$\begin{bmatrix} B & 0 & A_h^T & A_g^T \\ 0 & S^{-1}\Lambda_g & 0 & I \\ A_h & 0 & 0 & 0 \\ A_g & I & 0 & 0 \end{bmatrix}^{-1}$$

$$= \left(\begin{bmatrix} \xi I & 0 & A_h^T & A_g^T \\ 0 & S^{-1}\Lambda_g & 0 & I \\ A_h & 0 & 0 & 0 \\ A_g & I & 0 & 0 \end{bmatrix} + \begin{bmatrix} W \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} MW^T & 0 & 0 & 0 \end{bmatrix} \right)^{-1}$$

$$\equiv [C + UV^T]^{-1}.$$

Applying the Sherman-Morrison-Woodbury formula the right hand side can be written as

$$C^{-1} - C^{-1}U \left(I + V^T C^{-1} U\right)^{-1} V^T C^{-1}. \tag{3.36}$$

Computing the primal-dual step (2.4) therefore requires the solution of systems of the form $Cz = b$, which will be done by means of the HSL routine `MA27`.

4 Numerical Results

The hybrid algorithm described in the previous sections has been implemented in the KNITRO package. We will refer to the new algorithm as Knitro-Direct because in practice the majority of the iterations are obtained by factoring the primal-dual system. We will compare it with the trust region algorithm in KNITRO 3.0, which will be called henceforth Knitro-CG. We use the CUTEr collection [2, 19] as of June 5, 2003, from which 968 problems have been retained; the remaining problems have been discarded because they are too memory consuming.

All tests were performed on a 2.8GHz Pentium Xeon, with 3Gb of memory running Red Hat Linux. KNITRO 3.0 is written in C and Fortran 77, and was compiled using the gcc and g77 compilers with the "`-O`" compilation flag, and was run in double precision. Limits of 15 minutes of CPU time and 3000 outer iterations were imposed for each problem; if one of these limits was reached the algorithm was considered to have failed. The stopping tolerances in (3.22)-(3.24) were set as $\epsilon^{opt} = \epsilon^{feas} = 10^{-6}$. For details on the convergence criteria used in KNITRO 3.0 see [28].

First, we compare in Figure 1 the relative performance of Knitro-Direct and Knitro-CG, in terms of number of function/constraint evaluations, on the 968 problems from CUTEr. For the number of function/constraint evaluations we take the maximum of these two quantities. Here and in the figures that follow we use the logarithmic performance profiles proposed by Dolan and Moré [8]. Note that the two algorithms are remarkably similar in terms of robustness and efficiency, and a profile based on iterations is almost identical to that given in Figure 1. One has to note that even though these two algorithms are quite different in the way they generate steps, they share many other features including common NLP and barrier stop tests, merit function, second-order-correction, and barrier parameter update strategy.

Next we compare computing times. Since timing results are uninteresting, and can be unreliable on small problems which are solved very quickly, we will consider only those problems for which $n + \bar{m} \geq 1000$, where $\bar{m}$ is the number of equality and general inequalities (excluding bounds). Figure 2 gives the performance profile for all CUTEr problems with this size restriction.

One could expect that Knitro-Direct would require less computing time than Knitro-CG on problems in which the Hessian of the Lagrangian is ill-conditioned (causing a very large number of CG iterations) and where factoring the KKT matrix is not excessively expensive. A detailed examination of the results shows that this is the case, but we should add that

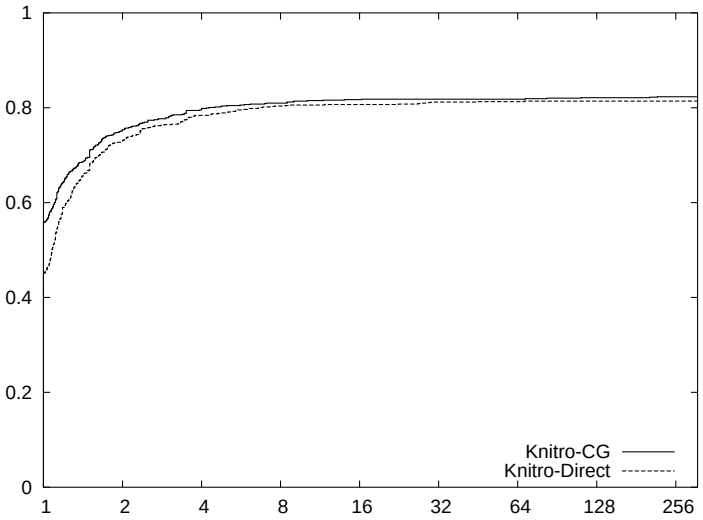

Figure 1: Number of function evaluations for Knitro-Direct and Knitro-CG on 968 CUTEr problems.

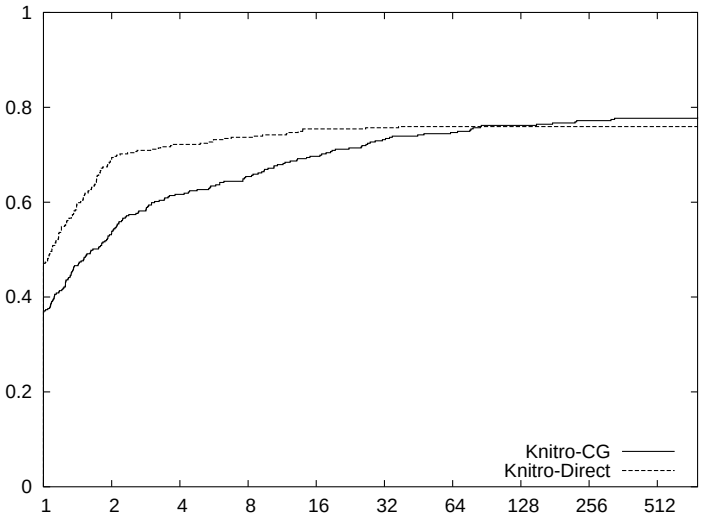

Figure 2: CPU time for Knitro-Direct and Knitro-CG on 399 CUTEr problems for which $n + \bar{m} \geq 1000$.

Prob type	# probs	TR invoked	neg curv	backtrack	cut-back
All	788	22.9%	15.9%	6.0%	1.0%
Inequality	481	22.3%	13.8%	7.1%	1.4%

Table 1: Frequency and reason for invoking trust region step.

the problems in which Knitro-Direct has a clear advantage in speed are also characterized by the fact that negative curvature was rarely encountered.

For example, when considering the 29 problems in which Knitro-Direct was at least 10 times faster than Knitro-CG, we observe that typically 90% of Knitro-CG's time was taken by the CG iteration (excluding the factorization), and only in one of those problems did Knitro-Direct report negative curvature. On the other hand, in the runs in which Knitro-Direct required much more computing time than Knitro-CG we observe that the factorization of the primal-dual matrix is very expensive (requiring typically 85% of the total time or more) and/or negative curvature was encountered frequently.

4.1 Transition between algorithms and negative curvature.

Table 1 provides statistics on the transition to trust region steps. It is based on the 788 CUTEr problems in which Knitro-Direct reported finding an optimal solution. The table gives the percentage of iterations in which the trust region step was invoked, and then gives a breakdown of this number in terms of the three factors that can cause this: (i) negative curvature was encountered, (ii) the line search did not succeed in reducing the merit function after 3 backtracking steps, or (iii) the length of the primal-dual step had to be cut back by more than the factor δ. The first row of this table looks at the whole set of 788 problems while the second row looks only at the 481 problems from this set which have at least one inequality constraint or bound (such that the cut back procedure for the primal-dual step is relevant). It can be seen from Table 1, that the great majority of steps taken in the Knitro-Direct algorithm are direct steps. Moreover, most of the conjugate gradient iterations result from negative curvature as opposed to short steplengths resulting from perhaps singularities.

We have observed a deficiency in our procedure for handling negative curvature. In problem BLOCKQP1, every iteration of Knitro-Direct reports negative curvature and reverts to the trust region method. However, the trust region iteration terminates after only 1 or 2 CG steps without detecting

negative curvature. Therefore, the algorithm converges to a stationary point that is not a minimizer, and this is unsatisfactory in this case since we had a clear indication from the factorization of the primal-dual matrix that negative curvature existed.

It would seem that a way of overcoming the problem would be to replace the Steihaug-CG iteration implemented in Knitro-CG by a Krylov routine like GLTR that is able to continue after negative curvature is detected. Interestingly, this would not have helped in this case because the CG iteration terminated by meeting the stop tolerance and before reaching the boundary of the trust region. Although this deficiency has been observed only in a handful of problems, it calls for a more robust procedure for exploiting negative curvature in these situations.

4.2 Quasi-Newton Options

We first analyzed the performance of the three quasi-Newton options in Knitro-Direct on all small and medium size problems in CUTEr, which were defined as problems with $n + \bar{m} < 1000$. This size restriction is necessary because Knitro-Direct-BFGS and Knitro-Direct-SR1 implement dense quasi-Newton approximations to the Hessian of the Lagrangian. Knitro-Direct-LM refers to the limited memory BFGS option storing 20 correction pairs (i.e. $p = 20$ in the notation of §3.8). Figure 3 compares these three options in terms of function/constraint evaluations.

Note that in these tests the SR1 method performed better than BFGS. More significantly, the limited memory BFGS method is very similar in performance to its dense counterpart.

Next we compare in Figure 4 the performance, in terms of function evaluations, of the following 4 methods: (i) Knitro-Direct with exact Hessian (ii) Knitro-Direct with limited memory BFGS updating (iii) Knitro-CG using limited memory BFGS and (iv) Knitro-CG with an option in which Hessian-vector products are computed by finite differences of gradients [28]. As before, in the limited memory methods we stored 20 correction pairs. Here we compare on the complete test set of 968 problems of all sizes.

As expected, the quasi-Newton versions are less robust and efficient than the second derivative options, but we find their performance acceptable. Out of 968 problems, Knitro-Direct with second derivatives solved 788 problems, whereas Knitro-Direct with limited memory BFGS solved 693. We note, in passing, that the finite-difference option is only slightly less efficient than the methods that use exact second derivatives. The observation that the limited memory option is comparable on small problems to its dense counterparts,

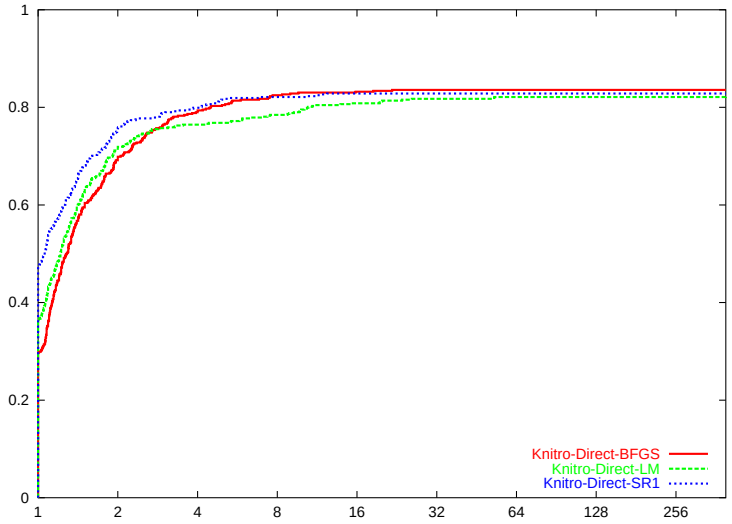

Figure 3: Number of function evaluations for 3 quasi-Newton versions of Knitro-Direct on 548 CUTEr problems for which $n + \bar{m} < 1000$.

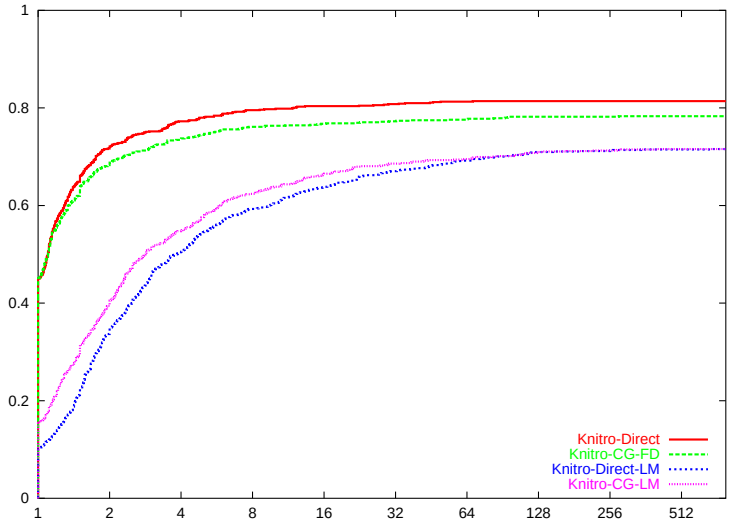

Figure 4: Number of function evaluations for options using exact Hessian, limited memory quasi-Newton and finite differences on 968 CUTEr problems.

and the fact that it is applicable to large problems, suggests that it can be used as the default quasi-Newton option in Knitro-Direct.

5 Final Remarks

In this paper we have discussed an algorithm in which line search steps based on the direct factorization of the primal-dual equations are always tried first and which employs trust region iterations as a safeguard. The flexibility provided by the new software allows other algorithmic options. For example, one could always attempt trust region steps first, but if the cost of the CG iteration is deemed to be excessive, revert to Knitro-Direct. Or one could choose the type of iteration dynamically during the problem solution. Such options, as well as improved procedures for dealing with negative curvature will be the subject of future investigations.

Acknowledgments. We would like to thank Guanghui Liu and Marcelo Marazzi for their work on an early version of this software. We also thank Richard Byrd for many useful conversations on the design of this algorithm.

References

[1] J. Betts, S. K. Eldersveld, P. D. Frank, and J. G. Lewis. An interior-point nonlinear programming algorithm for large scale optimization. Technical report MCT TECH-003, Mathematics and Computing Technology, The Boeing Company, P.O. Box 3707, Seattle WA 98124-2207, 2000.

[2] 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(1):123–160, 1995.

[3] R. H. Byrd, J. Ch. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[4] 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.

[5] R. H. Byrd, M. Marazzi, and J. Nocedal. On the convergence of Newton iterations to non-stationary points. Technical Report OTC 2003/3,

Optimization Technology Center, Northwestern University, Evanston, Illinois, USA, March 2003. To appear in Mathematical Programming.

[6] R. H. Byrd, J. Nocedal, and R. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 49(3):285–323, 1991.

[7] A. R. Conn, N. I. M. Gould, and Ph. Toint. *Trust-region methods*. MPS-SIAM Series on Optimization. SIAM publications, Philadelphia, Pennsylvania, USA, 2000.

[8] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.

[9] J. P. Dussault. Numerical stability and efficiency of penalty algorithms. *SIAM Journal on Numerical Analysis*, 32(1):296–317, 1995.

[10] 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. *Journal of Optimization Theory and Applications*, 89(3):507–541, June 1996.

[11] 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.

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

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

[14] A. Forsgren and P. E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4):1132–1152, 1998.

[15] D. M. Gay, M. L. Overton, and M. H. Wright. A primal-dual interior method for nonconvex nonlinear programming. In Y. Yuan, editor, *Advances in Nonlinear Programming (Beijing, 1996)*, pages 31–56, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.

[16] E. M. Gertz and P. E. Gill. A primal-dual trust region algortihm for nonlinear programming. Numerical Analysis Report NA 02-1, University of California, San Diego, 2002.

[17] N. I. M. Gould, D. Orban, A. Sartenaer, and Ph. L. Toint. Superlinear convergence of primal-dual interior-point algorithms for nonlinear programming. *SIAM Journal on Optimization*, 11(4), 2001.

[18] N. I. M. Gould, D. Orban, and Ph. Toint. GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. Technical Report RAL-TR-2002-014, Rutherford Appleton Laboratory, Chilton, England, 2002.

[19] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. Technical Report TR/PA/01/04, CERFACS, Toulouse, France, 2003. To appear in Transactions on Mathematical Software.

[20] Harwell Subroutine Library. *A catalogue of subroutines (HSL 2000).* AEA Technology, Harwell, Oxfordshire, England, 2002.

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

[22] F. Potra. Q-superlinear convergence of the iterates in primal-dual interior-point methods. *Mathematical Programming B*, 91(1), 2001.

[23] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[24] A. L. Tits, A. Wächter, S. Bakhtiari, T. J. Urban, and C. T. Lawrence. A primal-dual method for nonlinear programming with strong global and local convergence properties. Technical Report 2002-29, Institute for Systems Research, University of Maryland, College Park, MD, 2002.

[25] R. J. Vanderbei and D. F. Shanno. An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[26] 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.

[27] A. Wächter and L. T. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Mathematical Programming*, 88(3):565–574, 2000.

[28] R. A. Waltz and J. Nocedal. KNITRO user's manual. Technical Report OTC 2003/05, Optimization Technology Center, Northwestern University, Evanston, IL, USA, April 2003.

[29] H. Yabe and H. Yamashita. Q-superlinear convergence of primal-dual interior point quasi-Newton methods for constrained optimization. *Journal of the Operations Research Society of Japan*, 40(3):415–436, 1997.

[30] H. Yamashita. A globally convergent primal-dual interior-point method for constrained optimization. Technical report, Mathematical System Institute, Inc., Tokyo, Japan, May 1992. Revised March 1994.