Feasible Interior Methods Using Slacks for Nonlinear Optimization

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

February 26, 2002

Abstract

A slack-based feasible interior point method is described which can be derived as a modification of infeasible methods. The modification is minor for most line search methods, but trust region methods require special attention. It is shown how the Cauchy point, which is often computed in trust region methods, must be modified so that the feasible method is effective for problems containing both equality and inequality constraints. The relationship between slack-based methods and traditional feasible methods is discussed. Numerical results showing the relative performance of feasible versus infeasible interior point methods are presented.

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

1   Introduction

In many applications, it is desirable for all of the iterates generated by an optimization algorithm to be feasible with respect to some or all of the inequality constraints. For example, the objective function may be defined only when some of the constraints are satisfied, making this feature absolutely necessary. In other instances one may want to terminate an algorithm before optimality has been reached and be assured that the current approximate solution is feasible.

Various feasible active set methods (see, e.g. [15]) have been developed by including deflections in the search directions which ensure that the total step points towards the interior of the feasible region. They typically require the solution of two or more linear systems of equations per iteration, although some recent approaches [16] aim at decreasing the cost per iteration.

Interior point approaches provide a natural framework for deriving feasible methods for nonlinear programming. The methods proposed in [1, 7, 12, 13] either start with a feasible point or apply a phase-one procedure to compute one, and then generate strictly feasible iterates. Most other implementations of interior methods for nonlinear programming are based, however, on infeasible algorithms [5, 10, 20, 23] which may enter and leave the feasible region during the course of the minimization.

In this paper we describe a framework for transforming slack-based infeasible methods into feasible methods. In this framework, feasible and infeasible interior algorithms can be considered as variants of the same basic method. Feasibility is controlled by whether or not one resets the slack variables after a trial step has been taken, and how these variables are reset. Using this flexibility one can choose to enforce feasibility with respect to some, all, or none of the inequality constraints depending on what is needed or desired. In addition, this flexibility provides a convenient testing environment for analyzing the effects of staying feasible in interior point methods.

The slack reset strategies may experience difficulties on problems with both equality and inequality constraints. The difficulties will not arise in most line search methods, but can occur in trust region methods. We describe a procedure for generating search directions which ensures that the feasible methods proposed here behave efficiently for problems containing both equality and inequality constraints.

The paper is organized as follows. We first outline, in section 2, the general formulation of an infeasible interior point algorithm for nonlinear programming. In section 3 we describe a strategy which transforms infeasible interior methods into methods that satisfy some or all of the inequality constraints by resetting slack variables. The relationship between these slack-based feasible methods and the classical methods of Fiacco and McCormick [11] is discussed. In section 4 we describe potential difficulties with our strategy when equality constraints are present, and provide guidelines to deal with them. A concrete procedure for modifying the step-computation in the KNITRO algorithm is presented in section 5, and numerical results comparing feasible and infeasible methods are reported in section 6. We conclude the paper with final remarks in section 7.

2   Infeasible Methods

The problem under consideration will be formulated as

$$\min_{x} \quad f(x) \tag{2.1a}$$

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

$$g(x) \geq 0, \tag{2.1c}$$

where f, h and g are sufficiently smooth functions of the variable $x \in I\!\!R^n$. Here f is a scalar-valued function, and h and g are vector-valued functions. By a feasible method for (2.1) we mean one in which the starting point and all subsequent iterates satisfy the inequality constraints (2.1c).

Infeasible interior methods do not enforce satisfaction of the inequality constraints at each iteration. They typically make use of slack variables to transform (2.1) into the equivalent problem

$$\min_{x,s} \quad f(x) \tag{2.2a}$$

$$\text{s.t.} \quad h(x) = 0 \tag{2.2b}$$

$$g(x) - s = 0 \tag{2.2c}$$

$$s \geq 0. \tag{2.2d}$$

We will consider interior methods that, at each iteration, apply a form of Newton's method to solve, to some degree of accuracy, the barrier problem

$$\min_{x,s} \quad \psi(x, s; \mu) \equiv f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) \tag{2.3a}$$

$$\text{s.t.} \quad h(x) = 0 \tag{2.3b}$$

$$g(x) - s = 0 \tag{2.3c}$$

$$s > 0, \tag{2.3d}$$

where μ is a positive parameter and $\mathcal{I}$ is the set of indices corresponding to the inequality constraints. We will assume that the methods use a merit function of the form

$$\phi(x, s) = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) + \chi(c(x, s)), \tag{2.4}$$

where χ is some measure of infeasibility, and

$$c(x, s) = \left[\begin{array}{c} h(x) \\ g(x) - s \end{array} \right]. \tag{2.5}$$

The function χ can be chosen as a vector norm, or as some other function with the properties that $\chi(0) = 0$ and $\phi(x, s) \to \infty$ as $s \to 0$. We will also define ϕ to have the value ∞ for $s \leq 0$.

Let us consider the following very general type of iterative method for solving the barrier problem (2.3). This method will be applied until (2.3) is solved to some accuracy; then a new barrier parameter is chosen, and the method is applied again. Methods that change the barrier parameter at each iteration would then apply a single iteration of the following method to each barrier problem.

Algorithm 2.1 *Generic Algorithm (Infeasible method for problem* (2.3)*)*

An iterate x (possibly infeasible) and a slack vector $s > 0$ are given.
while *a stopping rule is not satisfied*
      Compute the step $d = (d_x, d_s)$.
      Define the trial point $x_T = x + d_x$; $s_T = s + d_s$.
      while $\phi(x_T, s_T)$ *is not sufficiently smaller than* $\phi(x, s)$
            Compute a shorter step d.
            Set $x_T = x + d_x$; $s_T = s + d_s$.
      end (while)
      Set $x_+ = x_T$; $s_+ = s_T$.
end (while)

In a trust region method, a shorter step would be obtained by decreasing the trust region radius and recomputing a step, whereas in a line search method, a backtracking line search would be employed. We assume that the step-generation procedure and the merit function ϕ are compatible in the sense that if $\|d\|$ is sufficiently small, the merit function will be decreased. No other assumptions will be made on d until we consider, in section 4, its effect when both equality and inequality constraints are present.

3   A Feasible Method with Slack Resetting

We now describe a way of transforming this infeasible Generic Algorithm into a feasible method while retaining the use of slack variables. The motivation for doing so is two-fold. First, by using slacks, the feasible and infeasible versions can be implemented with minimal changes to the algorithm and data structures. In particular, the linear system to be solved at each iteration of the interior method will be identical for the feasible and infeasible versions. Second, this framework makes it very simple to decide which constraints should be honored – and to change this choice if desired. By contrast, in a feasible method that does not use slacks, feasibility is imposed by applying a barrier function on the inqualities to be honored – something that results in substantial changes in the algorithm.

In what follows we will assume for simplicity that the feasible method must satisfy all inequality constraints at every iteration. It will become clear, however, that it is straightforward to extend the algorithm described below to the case when only some of the inequality constraints must be honored.

3.1 Feasible Algorithm Description

Assume the current iterate is feasible with respect to the inequality constraints. To ensure that the next iterate is also feasible, we introduce the following simple modification. After computing a step (d_x, d_s) we redefine the slacks as

$$s_{\mathrm{T}} \leftarrow g(x_{\mathrm{T}}), \tag{3.6}$$

and test whether the point $(x_{\mathrm{T}}, s_{\mathrm{T}})$ is acceptable for the merit function (2.4). If it is not, we reject the step and compute a new, shorter, trial step.

If the initial iterate x_0 does not satisfy all inequality constraints, we first apply the infeasible Generic Algorithm until all inequalities are greater than some positive threshold value. At that point the algorithm switches to the feasible mode, and stays feasible for the rest of the optimization calculation. This algorithm (Feas-Reset) is summarized below. We let e denote the vector of ones, of appropriate dimension. As before, we define the merit function ϕ to have the value ∞ for $s \leq 0$.

Algorithm 3.1 *Algorithm Feas-Reset*

An iterate x (possibly infeasible), a slack vector $s > 0$, and a positive threshold
value τ are given.
if $g(x) < \tau e$ **then**
   Run infeasible Generic Algorithm until $g(x) \geq \tau e$.
   Set $s = g(x)$.
end (if)
while *a stopping test is not satisfied*
   Compute the step $d = (d_x, d_s)$ as in the Generic Algorithm.
   Define the trial point $x_T = x + d_x$; $\boxed{s_T = g(x_T)}$.
   while $\phi(x_T, s_T)$ *is not sufficiently smaller than* $\phi(x, s)$
      Compute a shorter step d.
      Set $x_T = x + d_x$; $\boxed{s_T = g(x_T)}$.
   end (while)
   Set $x_+ = x_T$; $s_+ = s_T$.
end (while)

The test $g(x) \geq \tau e$, can be replaced by some other condition that does not treat each constraint equally and that takes into account the scale of the constraints.

Note that the vector d_s is not needed in Algorithm Feas-Reset, but we still assume that a step in the slacks and variables is computed at every iteration, so that the feasible and infeasible modes require the same data structures and variables – and only differ in the two instructions enclosed in boxes. Our numerical experience indicates that the cost of solving the larger system (in (d_x, d_s)) using the HSL routine MA27 [14] is not significantly larger (if at all) than solving a reduced system (in d_x only).

Making the substitution (3.6) has the effect of replacing $\ln(s_i)$ with $\ln(g_i(x))$ in the merit function, which is the standard form of classical barrier functions [11]. If at a trial point we have that $g_i(x_{\mathrm{T}}) \leq 0$ for some inequality constraint, the value of the merit function is $+\infty$, and we reject the trial point. Note that this approach will also reject steps $x + d_x$ that are too close to the boundary of the feasible region because such steps increase the barrier term $-\mu \sum_{i \in \mathcal{I}} \ln(s_i)$ in the merit function (2.4).

In section 6 we show that a trust region implementation of this feasible method is efficient in practice for problems with inequality constraints only, but that in order to handle problems with both equalities and inequalities, a modification to the step computation must be made.

3.2   Equivalence of Slack-based and Classical Feasible Methods

We now ask if Algorithm Feas-Reset, in its feasible mode, is identical to a classical barrier method without slacks of the type described in [11]. By a classical barrier method we mean one in which Newton's method is applied to the problem:

$$\min_{x} \qquad f(x) - \mu \sum_{i \in \mathcal{I}} \ln(g_i(x)) \tag{3.7a}$$

$$\text{s.t.} \qquad h(x) = 0. \tag{3.7b}$$

(Throughout this section we assume that $g(x) > 0$.) At first it may appear that Algorithm Feas-Reset cannot be equivalent to this method because it uses slack variables in the step computation—even in feasible mode. It is easy to see, however, that for a class of interior methods, the reset (3.6) has the effect of eliminating the slacks in the step computation and working directly with problem (3.7).

To show this we first note that the KKT conditions of (3.7) are

$$\nabla f(x) - A_h \lambda_h - \mu A_g G(x)^{-1} e = 0$$
$$h(x) = 0,$$

where A_h^T and A_g^T denote the Jacobian matrices of h and g, respectively, λ_h is the vector of Lagrange multipliers for the equality constraints (3.7b), $G(x)$ is a diagonal matrix whose diagonal is given by the components of $g(x)$, and e is a vector of all ones. These conditions can be reformulated so as to be more benign for Newton's method: introducing the variable

$$\lambda_g = \mu G^{-1}(x) e,$$

we obtain

$$\nabla f(x) - A_h \lambda_h - A_g \lambda_g = 0$$
$$h(x) = 0 \tag{3.8}$$
$$-\mu e + G(x)\lambda_g = 0.$$

Applying Newton's method (in the variables, x, λ_h, λ_g) to this system gives a primal-dual interior method for (3.7); see e.g. [8].

To study the relationship between this method and Algorithm Feas-Reset, let us consider a slack-based, feasible method for solving (2.3) that computes steps by applying Newton's method in the variables $x, s, \lambda_h, \lambda_g$ to the system

$$
\begin{aligned}
\nabla f(x) - A_h \lambda_h - A_g \lambda_g &= 0 \\
\lambda_i s_i &= \mu, \quad i \in \mathcal{I} \\
h(x) &= 0 \\
g(x) - s &= 0,
\end{aligned}
$$

which is equivalent to the KKT conditions for (2.3). This system is the basis for primal-dual infeasible algorithms; see e.g. [5, 20]. Application of Newton's method gives rise to the linear system

$$
\begin{pmatrix}
\nabla_{xx}^2 L & 0 & A_h(x) & A_g(x) \\
0 & \Lambda & 0 & -S \\
A_h(x)^T & 0 & 0 & 0 \\
A_g(x)^T & -I & 0 & 0
\end{pmatrix}
\begin{pmatrix}
d_x \\
d_s \\
-d_{\lambda_h} \\
-d_{\lambda_g}
\end{pmatrix}
= -
\begin{pmatrix}
\nabla_x L(x, \lambda) \\
-\mu e + S \lambda_g \\
h(x) \\
g(x) - s
\end{pmatrix},
\tag{3.9}
$$

where S and Λ denote diagonal matrices with s and λ_g on their respective diagonals, and L stands for the Lagrangian of (2.1):

$$
L(x, \lambda_h, \lambda_g) = f(x) - \lambda_h^T h(x) - \lambda_g^T g(x).
\tag{3.10}
$$

Using the fact that $g(x) - s = 0$ due to the reset (3.6), we can eliminate d_s and s to obtain the equivalent linear system

$$
\begin{pmatrix}
\nabla_{xx}^2 L & A_h(x) & A_g(x) \\
A_h(x)^T & 0 & 0 \\
\Lambda A_g(x)^T & 0 & -G(x)
\end{pmatrix}
\begin{pmatrix}
d_x \\
-d_{\lambda_h} \\
-d_{\lambda_g}
\end{pmatrix}
= -
\begin{pmatrix}
\nabla_x L(x, \lambda) \\
h(x) \\
-\mu e + G(x) \lambda_g
\end{pmatrix}.
\tag{3.11}
$$

This system is just Newton's method in the variables x, λ_h, λ_g applied to (3.8). Therefore, a primal-dual step for (3.7) is equivalent to a primal-dual step for (2.3), when the reset (3.6) is applied.

This equivalence does not hold for interior methods [5, 9, 23] in which the step d is only an approximate solution of (3.9), or is computed as the solution of a related problem with a trust region. However, as the iterates of those methods approach a solution, their steps approximate (3.9) with increasing accuracy, and their feasible version resembles a classical barrier method.

4 Effects of Equality Constraints

An iteration of Algorithm Feas-Reset (Algorithm 3.1) is successful if it results in a decrease in the merit function (2.4). The computation of the step (d_x, d_s) is designed to cause such a decrease, but the slack reset step $s_\mathrm{T} \leftarrow g(x_\mathrm{T})$ can, however, offset it. In particular, if d_x leads toward the boundary of an inequality constraint, the reset $s_\mathrm{T} \leftarrow g(x_\mathrm{T})$ in Algorithm

Feas-Reset can cause the corresponding slack variable to take on a smaller value, increasing the term $-\mu \sum_{i \in \mathcal{I}} \ln(s_i)$ in the merit function. If this results in a total increase in the merit function, then a shorter step is computed. This behavior is not unexpected: it is the mechanism that prevents Algorithm Feas-Reset from generating steps that leave the feasible region or that get too close to its boundary. Indeed, for problems with inequality constraints only, this mechanism steers the iterates away from the boundary and results in an effective method.

It turns out, however, that when both equality and inequality constraints are present, it is harder to keep the iterates away from the boundary of the feasible region, and the attempt to stay feasible by means of slack resetting can actually cause the method to fail. To see what is the source of this problem, let us denote by δ_s the change in the slack due to the reset, i.e.,

$$s + d_s + \delta_s = s_{\mathrm{T}} = g(x + d_x). \tag{4.12}$$

As mentioned above, δ_s can cause an increase in the merit function even when the original step $d = (d_x, d_s)$ would have decreased it, and this is not necessarily undesirable. It is essential, however, that if a sequence of steps is rejected and the steps become increasingly small, any merit function increase due to the slack reset δ_s is eventually offset by the decrease in ϕ provided by the step d. This guarantees that, if the current iterate is not a stationary point, the algorithm will move away from this point. Unfortunately this may not be the case in methods that handle constraints by a trust region method. To illustrate this, we now present an example that occurs with a feasible version of the algorithm implemented in the KNITRO package [4, 5].

Example 1. Consider the problem GAUSSELM from the CUTE collection [2]. One version of this problem has 385 nonlinear equality constraints along with 750 linear inequality constraints and some bounds on the variables. Using an initial point which was feasible with respect to the inequality constraints, we attempted to solve this problem using Algorithm Feas-Reset with steps d generated by KNITRO, a trust region interior method. The merit function is given by (2.4) with $\chi(\cdot) = \nu \| \cdot \|_2$, and $\nu > 0$. Some of the first 20 and last 3 iterations of the run are shown in Table 1. All the iterations of the run occur for a fixed barrier parameter value μ.

In Table 1, `Iter` refers to the iteration number, `Step` indicates whether or not the trial point was accepted or rejected, `Barr Obj` is the barrier objective value ψ defined in (2.3a), $\|h(x)\|_2$ is the norm of the equality constraints, `Delta` is the trust region radius, `Merit Red` is the reduction in the merit function ϕ obtained by the step, `Trial` indicates whether the trial point is feasible or not and $\|\delta_s\|_2$ is the norm of the perturbation due to the slack reset (3.6). We note that the slack reset is not performed if the trial point is infeasible.

Even though all iterates are feasible with respect to the inequality constraints, we can see from Table 1 that there exist violated equality constraints. From iteration 18 on, all the trial steps are feasible, but the merit function increases even as the trust region radius approaches zero. It is easy to see that this increase in the merit function is caused by the slack reset perturbation δ_s which is relatively large starting at iteration 5 and does not decrease quickly enough as the trust region approaches zero. Note that in the later

Iter	Step	Barr Obj	$\|h(x)\|_2$	Delta	Merit Red	Trial	$\|\delta_s\|_2$
1	OK	4.51e+01	1.294e-03	1.000e+00	6.98e-01	feas	3.486e-15
2	OK	4.14e+01	3.014e-03	7.000e+00	3.69e+00	feas	4.253e-15
3	OK	3.28e+01	7.173e-02	4.900e+01	8.50e+00	feas	5.989e-15
4	OK	3.07e+01	9.705e-01	9.800e+01	1.28e+00	feas	5.224e-15
5	OK	2.82e+01	8.083e-01	9.800e+01	9.89e-01	feas	2.345e-01
6	rej	2.66e+01	3.285e-01	9.800e+01	9.89e-01	inf	---------
7	rej	2.49e+01	4.642e-01	3.434e+00	9.89e-01	inf	---------
...							
14	OK	2.82e+01	7.980e-01	2.683e-02	8.09e-02	feas	4.347e-02
15	rej	2.81e+01	7.769e-01	5.366e-02	8.09e-02	inf	---------
16	rej	2.82e+01	7.874e-01	2.683e-02	8.09e-02	inf	---------
17	rej	2.82e+01	7.927e-01	1.341e-02	8.09e-02	inf	---------
18	rej	2.84e+01	7.954e-01	6.707e-03	-1.34e-01	feas	1.115e-02
19	rej	2.83e+01	7.967e-01	3.354e-03	-3.82e-02	feas	5.579e-03
20	rej	2.82e+01	7.974e-01	1.677e-03	-1.54e-02	feas	2.789e-03
...							
55	rej	2.82e+01	7.980e-01	4.880e-14	-2.86e-13	feas	8.136e-14
56	rej	2.82e+01	7.980e-01	2.440e-14	-5.42e-14	feas	4.088e-14
57	rej	2.82e+01	7.980e-01	1.220e-14	-1.55e-13	feas	2.007e-14

Table 1: Algorithm Feas-Reset on problem GAUSSELM

iterations the length of δ_s is comparable to the trust region radius and, as a consequence, to the length of the step d. The algorithm eventually fails because the trust region radius becomes smaller than a preset tolerance.

This example indicates that the step produced by a feasible trust region method, such as the one implemented in KNITRO, is not appropriate for a slack reset in the presence of equality and inequality constraints. We will show, however, that by modifying the step computation, slack resets can still be effective.

4.1   Step acceptance in Algorithm Feas-Reset

As we will show later on, it is notable that the difficulties in Algorithm Feas-Reset observed in Example 1 do not occur for problems with only inequality constraints. To propose remedies, we need to understand what is special about the handling of equality and inequality constraints in a trust region method.

Let us begin by considering, by way of contrast, line search interior methods. They typically demand [1, 7, 10, 13, 20] that the step satisfy the linearized constraints

$$c(x, s) + A(x)^T d = 0, \tag{4.13}$$

where $c(x, s)$ is defined by (2.5) and $A(x)^T$ denotes its Jacobian matrix, i.e.,

$$A(x) = \begin{bmatrix} A_h(x) & A_g(x) \\ 0 & -I \end{bmatrix}. \tag{4.14}$$

If x is feasible such that $g(x) - s = 0$, the second block of equations in (4.13) reads

$$A_g(x)^T d_x - d_s = 0. \tag{4.15}$$

We now show that, when all steps satisfy (4.15), the slack reset in Algorithm Feas-Reset will cause at most a small increase in the merit function ϕ which is offset by the decrease due to d—provided this step is of appropriate length. As a consequence, when (4.15) holds, the type of failure exhibited in Example 1 will not occur.

We first note that most constrained optimization methods generate a step d that is a direction of first order decrease of the merit function, or at the very least, is arbitrarily close to such a direction when $\|d\|$ is sufficiently small [8]. (This property holds at any iterate that is not a stationary point of the problem, and where regularity holds.) This implies that at such a point there is a positive constant γ such that when d is sufficiently small, the decrease in the merit function satisfies

$$\phi(x, s) - \phi(x + d_x, s + d_s) \geq \gamma \|d\|. \tag{4.16}$$

The following result gives conditions under which the perturbation δ_s defined in (4.12) will be much smaller than d, ensuring that the decrease in the merit function due to the step d will be maintained.

Theorem 4.1 *Suppose Algorithm Feas-Reset is applied to the barrier problem* (2.3), *and that the function χ in* (2.4) *and the Jacobian $A(x)^T$ are Lipschitz continuous. Let (x, s) be a feasible iterate with $s > 0$, and suppose that the step $d = (d_x, d_s)$ satisfies* (4.15). *Then the perturbation in the slacks due to the reset* (3.6) *satisfies*

$$\|\delta_s\| = O(\|d_x\|)^2.$$

As a consequence, if (4.16) *holds, then for d sufficiently small the total step $(d_x, d_s + \delta_s)$ will reduce the merit function and the step will be accepted.*

Proof. The equality $g(x) - s = 0$, and (4.15) imply that

$$\|g(x + d_x) - (s + d_s)\| = O(\|d_x\|^2). \tag{4.17}$$

Combining this with (4.12) we obtain

$$\|\delta_s\| = O(\|d_x\|^2). \tag{4.18}$$

The effect of this reset on the merit function ϕ is also $O(\|\delta_s\|)$ since ϕ is Lipschitz continuous so that

$$\phi(x + d_x, s + d_s) - \phi(x + d_x, s + d_s + \delta_s) = O(\|\delta_s\|) = O(\|d_x\|^2).$$

Together with (4.16) this implies

$$\phi(x,s) - \phi(x + d_x, s + d_s + \delta_s) \geq \gamma \|d\| + O(\|d_x\|^2), \qquad \gamma > 0.$$

Clearly for d sufficiently small the first term on the right hand side dominates the second term and we have that $\phi(x + d_x, s + d_s + \delta_s) < \phi(x,s)$ so that the step is accepted.

□

Theorem 4.1 guarantees that the failure observed in Example 1 cannot occur with line search methods since their search directions typically satisfy (4.15) and (4.16). On the other hand, many trust region methods, including the algorithm in KNITRO, do not always impose (4.15) and the guarantee of Theorem 4.1 is not available. (Of course for such trust region methods (4.16) still holds so the merit function increase in cases like Example 1 for small $\|d\|$ must be due to the reset δ_s.) We now discuss why (4.15) may not be satisfied by trust region methods, and how the difficulties illustrated in Example 1 can be remedied.

4.2 Behavior of feasible trust region methods

In a trust region method for solving the equality constrained barrier problem (2.3a)-(2.3c), the step $d = (d_x, d_s)$ is not computed by solving the Newton system (3.9). Instead it is defined as the (approximate or exact) solution of a subproblem of the form

$$\min_{d} \quad \tfrac{1}{2} d^T W d + \nabla \psi(x,s;\mu)^T d \tag{4.19a}$$

$$\text{s.t.} \quad h(x) + A_h^T d = r_h \tag{4.19b}$$

$$g(x) - s + A_g^T d_x - d_s = r_g \tag{4.19c}$$

$$\|d\| \leq \Delta. \tag{4.19d}$$

Here W is the Hessian of the Lagrangian of the barrier problem (2.3), or a related matrix, and ψ is the barrier function. The vectors r_h and r_g are chosen with the dual objective of ensuring that the constraints (4.19b)-(4.19d) are compatible, and that the step makes sufficient progress towards achieving feasibility. The methods of Vardi [21], Celis, Dennis and Tapia [6], Powell and Yuan [19], Byrd and Omojokun [3, 17], and Yamashita and Yabe [22] can be described in this framework.

If there is a step d satisfying (4.19b)-(4.19d) for $r_h = r_g = 0$, then these trust region methods will usually define r_h and r_g to be zero. In particular, these methods always set $r_g = 0$ when the problem contains *only inequality* constraints and the current iterate is feasible. In this case $g(x) = s$, and thus (4.19c) can be satisfied with $r_g = 0$ by arbitrarily small steps d that lie inside the trust region. Thus if Algorithm Feas-Reset is used with such a method, (4.15) will be always be satisfied, and by Theorem 4.1 a failure like Example 1 will not occur for problems containing only inequality constraints.

It is often the case, however, that for general problems the constraints (4.19b)-(4.19d) are inconsistent and nonzero values of r_h and r_g must be chosen. Moreover, in many standard trust region methods [3, 6, 17, 19] $r = (r_h, r_g)$ is chosen based on the constrained minimization of the squared Euclidean norm $\|r\|_2^2$ (see. e.g. (5.21)), making it almost

certain that if one of the vectors r_g, r_h is nonzero then *both* r_g and r_h will be nonzero. As a result the linear equation (4.15), which we have seen guarantees that the slack reset is not harmful, will not be satisfied in general. This is undesirable in the context of Algorithm Feas-Reset because the reset perturbation δ_s can be as large as the step d and cause a net increase in the merit function even as $\|d_x\| \to 0$. This can even occur when all constraints are linear.

Since the violation of (4.15) occurs no matter how small Δ is, it could cause Algorithm Feas-Reset to get stuck at a non-optimal point from which it could make no further progress, as seen from Example 1. Note that the problem cannot be overcome simply by backtracking along d since if $A_g(x)^T d_x - d_s \neq 0$ the same holds for any scalar multiple of d. We now propose a general strategy for overcoming this problem.

Feasible Step Conditions: The difficulties mentioned above can be resolved by demanding that, whenever $g(x) - s = 0$, the vector r_g be chosen to be zero in (4.19c). The vector r_h should still ensure that (4.19b) and (4.19d) are compatible and that the step makes sufficient progress toward satisfying the equality constraints $h(x)$.

Below we discuss how to apply this strategy in the algorithm implemented in the KNITRO package.

5 Feasible Steps in KNITRO

The algorithm implemented in KNITRO is a composite-step trust region method (following the terminology of [8]; Chapter 15). The step d is given as the sum of two components,

$$d = \eta + t, \tag{5.20}$$

where the "tangential component" t lies in the null space of $A(x)^T$ and attempts to move towards optimality, while the "normal component" η attempts to improve feasibility. The vectors r_h and r_g in (4.19) are obtained during the computation of the normal component η, and are not changed by the tangential component. Therefore, in order to enforce the satisfaction of (4.15) by the step d computed by KNITRO, we must make sure that the normal component of the step sets $r_g = 0$. We now discuss the computation of the normal component η in detail.

The normal component $\eta = (\eta_x, \eta_s)^T$ is defined as the solution of the auxiliary problem[1]

$$\min_{\eta} \quad \|A(x)^T \eta + c(x, s)\|_2^2 \tag{5.21a}$$

$$\text{s.t.} \quad \|\eta\|_{\text{TR}} \leq \theta \Delta, \tag{5.21b}$$

where A is given by (4.14) and the constant $\theta \in (0, 1]$. The trust region norm $\| \cdot \|_{\text{TR}}$ is scaled by $\|\eta\|_{\text{TR}} = \|(\eta_x, S^{-1}\eta_s)\|$ in order to steer the step away from the slack bounds; see

---

[1]In addition to (5.21b) a bound on slack steps is imposed, but this is done after the solution of (5.21).

[5]. Once the solution η is computed, we define

$$r_h = h(x) + A_h(x)^T \eta_x, \qquad r_g = g(x) - s + A_g(x)^T \eta_x - \eta_s. \tag{5.22}$$

To determine whether r_g is set to zero when $g(x) - s = 0$, we note that in KNITRO problem (5.21) is solved approximately using a dogleg method [18]: η is a linear combination of a Newton step η^N and a Cauchy direction. Specifically, it is the minimizer of (5.21a) along the piecewise linear path from 0 to $\alpha\eta^C$ to η^N subject to the trust region constraint, where $\alpha > 0$ is the minimizer of $\|A(x)^T \alpha\eta^C + c(x,s)\|_2^2$.

The Newton step is a solution of $A(x)^T \eta^N = -c(x,s)$ of minimum scaled norm $\|\cdot\|_{\mathrm{TR}}$. Hence by (4.14) when x is feasible the Newton step $\eta^N = (\eta_x^N, \eta_s^N)$ satisfies (4.15), i.e.,

$$A_g(x)^T \eta_x^N - \eta_s^N = 0. \tag{5.23}$$

If the Newton step lies inside the trust region (5.21b) then the dogleg method defines $\eta = \eta^N$, and r_g will be set to zero in (5.22). Therefore (4.15) will be satisfied when $g(x) - s = 0$ and Algorithm Feas-Reset will not encounter any difficulties in this case.

On the other hand, when the trust region is active, the dogleg method defines the step η to be a (nonzero) combination of the Cauchy direction η^C and the Newton step η^N. The Cauchy step does not satisfy (4.15) in general. To see this, note from (5.21) that η^C is the direction of steepest descent for the function

$$\|A(x)^T \eta + c(x,s)\|_2^2 \tag{5.24}$$

in the scaled norm. Differentiating (5.24) we see the when $g(x) - s = 0$, the steepest descent direction $\eta^C = (\eta_x^C, \eta_s^C)$ is given by

$$\eta_x^C = -A_h(x)h(x), \quad \text{and} \quad \eta_s^C = 0, \tag{5.25}$$

and therefore $A_g(x)^T \eta_x^C - \eta_s^C \neq 0$, in general. Thus, since the Cauchy direction η^C given by (5.25) will not, in general, satisfy (4.15) when the trust region is active, the same is true for the total step η which is a combination of η^C and η^N. Because of this, the difficulties illustrated in Example 1 may occur for Algorithm Feas-Reset with steps given by KNITRO when the trust region constraint is active in the normal step computation.

We can resolve these difficulties by redefining the Cauchy direction η^C. We will require that the new Cauchy direction satisfy the following three properties when $g(x) - s = 0$:

(i) it should satisfy (4.15),

$$A_g(x)^T \eta_x^C - \eta_s^C = 0; \tag{5.26}$$

(ii) it should make progress on the linearized equality constraints similar to that of the standard Cauchy step computed via (5.25). From (5.25) we can see that the first-order reduction on the equality constraints achieved by the standard Cauchy step is given by

$$\begin{aligned}
h(x) - (h(x) + \alpha A_h(x)^T \eta_x^C) &= -\alpha A_h(x)^T \eta_x^C \tag{5.27} \\
&= \alpha A_h(x)^T A_h(x)h(x). \tag{5.28}
\end{aligned}$$

In order to achieve similar reduction we require that the new Cauchy direction satisfy

$$A_h(x)^T \eta_x^C = -A_h(x)^T A_h(x) h(x). \tag{5.29}$$

(iii) Finally, to control the quality of the normal step and to take into account the shape of the trust region (5.21b), we require, as in the standard version of KNITRO, that the normal step should be in the range of the scaled Jacobian, i.e.,

$$\eta^C = \begin{bmatrix} A_h(x) & A_g(x) \\ 0 & -S^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$

for some vectors p_1, p_2; see (2.11) in [4].

Together these three desired properties of the modified Cauchy direction amount to requiring that $\eta^C = (\eta_x^C, \eta_s^C)$ solve the equations

$$\begin{bmatrix} I & 0 & A_h(x) & A_g(x) \\ 0 & I & 0 & -S^2 \\ A_h(x)^T & 0 & 0 & 0 \\ A_g(x)^T & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \eta_x^C \\ \eta_s^C \\ -p_1 \\ -p_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -A_h(x)^T A_h(x) h(x) \\ 0 \end{bmatrix}. \tag{5.30}$$

In summary, the change needed in the step computation of KNITRO occurs only in the normal component η of the step (5.20). This normal component is formed by a Newton step η^N, which needs no modification, and a Cauchy direction η^C, which is now computed via (5.30).

Note that if $A_h(x)$ is rank deficient, the Cauchy direction η^C is still determined by (5.30). If we delete the columns and rows of the matrix in (5.30) that correspond to the dependent columns of $A_h(x)$, the resulting matrix is nonsingular. If we then solve the resulting system, deleting the corresponding components of the right hand side, it is clear that the resulting solution will automatically satisfy the equations in (5.30) corresponding to the deleted columns due to the linear dependence.

We now apply Theorem 4.1 to show that the new step does not suffer from the difficulties illustrated in Example 1.

Corollary 5.1 *Suppose that Algorithm Feas-Reset is applied to the barrier problem (2.3), and that the step d is given by (5.20), where t is in the null space of $A(x)^T$, and η is a dogleg step whose Cauchy component is computed via (5.30). Assume the function χ in (2.4) is given by $\nu \|c(x, s)\|$, that the scalar ν is sufficiently large, and that $A(x)$ and $\nabla f(x)$ are Lipschitz continuous. Let (x, s) be a feasible iterate (with respect to the inequality constraints, and with $g(x) = s > 0$) that is not a stationary point of $\|h(x)\|^2$. If the trust region radius at (x, s) is sufficiently small, the step d will reduce the merit function and be accepted.*

Proof. The normal step η in (5.20) is a dogleg step which is a linear combination of a Newton step η^N and a Cauchy direction η^C. By (5.23) and (5.30) both components satisfy (4.15) when x is feasible, so that $A_g(x)^T \eta_x - \eta_s = 0$. It follows from (5.22) that $r_g = 0$,

so by (4.19c) and the fact that t is in the null space of $A(x)$, we have that the full step d satisfies (4.15) whenever $g(x) - s = 0$.

Now to use the second part of Theorem 4.1 we need only establish that d satisfies (4.16). As argued above, the matrix in (5.30) may be assumed nonsingular without loss of generality, so that η^C is well defined. Note that the dogleg step has the property that if the trust region radius $\theta\Delta$ is sufficiently small, the dogleg step is a scalar multiple of the Cauchy direction η^C. Thus, for sufficiently small Δ, we have from (5.25)

$$\eta = \theta\Delta\eta^C/\|\eta^C\|, \quad \text{and} \quad A_h(x)^T\eta_x = -\theta\Delta A_h(x)^T A_h(x)h(x)/\|\eta^C\|.$$

Since f and c are smooth and $h(x) \neq 0$, the merit function is continuously differentiable in a neighborhood of the iterate (x, s). Using this, (2.4), and the fact that $g(x) - s = 0$ we get

$$
\begin{aligned}
\phi(x,s) - \phi(x + d_x, s + d_s) &= -\nabla\phi(x,s)^T d + o(\|d\|) \\
&= -\nabla f(x,s)^T d_x + \mu e^T S^{-1} d_s - \nu\frac{(A_h(x)h(x))^T d_x}{\|h(x)\|} + o(\|d\|) \\
&= -\nabla f(x,s)^T d_x + \mu e^T S^{-1} d_s \\
&\quad + \nu\frac{\theta\Delta\|A_h(x)h(x)\|^2}{\|h(x)\|\|\eta^C\|} + o(\|d\|).
\end{aligned}
\tag{5.31}
$$

Note that $\|A_h h(x)\| \neq 0$ since (x, s) is not a stationary point of $\|h(x)\|^2$. Clearly, we can choose ν sufficiently large such that the sum of the first three terms of (5.31) is greater than half the (positive) third term, i.e.,

$$
\begin{aligned}
\phi(x,s) - \phi(x + d_x, s + d_s) &\geq \tfrac{1}{2}\nu\frac{\theta\Delta\|A_h(x)h(x)\|^2}{\|h(x)\|\|\eta^C\|} + o(\|d\|) \tag{5.32} \\
&\geq \gamma\|d\| + o(\|d\|), \tag{5.33}
\end{aligned}
$$

for some constant γ, where the last step follows from the fact that $\|d\| \leq \Delta$. It follows that (4.16) must hold for Δ sufficiently small, concluding the proof. (Note that the assumption that ν is sufficiently large is not restrictive since the type of adjustment described above is commonly done in practice; see e.g. [5]).

$\square$

The step defined by (5.30) is not the only modification that would be effective in the context of Algorithm Feas-Reset. For example, computing the steepest descent direction of $\|A_h(x)^T\eta_x + h(x)\|_2^2$ in the subspace of vectors satisfying (5.26) would yield an effective step, although it might be computationally expensive. A major advantage of using the solution to (5.30) is that (5.30) can be rewritten in such a way so that the coefficient matrix used to compute the new Cauchy direction is one which is already factored in KNITRO to compute the Newton direction η^N, Lagrange multiplier estimates, and each step of the projected conjugate gradient iteration used to solve (4.19). Thus the extra cost of computing the proposed Cauchy step is quite moderate, amounting to the cost of one backsolve using the factors of the coefficient matrix in (5.30), where several such backsolves usually are already performed for each iteraion.

Example 2. We consider again the behavior of Algorithm Feas-Reset on problem GAUS-SELM, this time using the modified normal step for KNITRO described above. The optimal solution is reached in 102 iterations without any difficulties. For the sake of space only the first 15 iterations (all using the same barrier parameter value) are shown in Table 2. We note that the perturbation caused by the slack reset, δ_s, is always of the order of unit roundoff, and therefore does not prevent a decrease in the merit function.

We also tested the modified normal step by comparing the performance of Algorithm Feas-Reset implemented in the KNITRO package with and without the modified step. Our test set had 153 problems, all containing equality constraints, and of those the two versions of KNITRO performed differently on 45 problems. Of these 45 problems the new step resulted in failure on 3 problems, as compared to 17 failures for the unmodified step. On the 25 problems solved by both, KNITRO with the new Cauchy step required significantly fewer iterations.

Iter	Step	Barr Obj	$\|h(x)\|_2$	Delta	Merit Red	Trial	$\|\delta_s\|_2$
1	OK	4.51e+01	1.294e-03	1.000e+00	6.98e-01	feas	3.486e-15
2	OK	4.14e+01	3.014e-03	7.000e+00	3.69e+00	feas	4.253e-15
3	OK	3.28e+01	7.173e-02	4.900e+01	8.50e+00	feas	5.989e-15
4	OK	3.04e+01	7.269e-01	9.800e+01	1.31e+00	feas	1.164e-14
5	OK	2.58e+01	5.199e-01	9.800e+01	4.86e+00	feas	4.063e-15
6	OK	1.20e+01	4.832e-01	9.800e+01	1.34e+01	feas	4.783e-15
7	OK	-2.77e-01	1.216e-01	9.800e+01	1.33e+01	feas	5.038e-15
8	OK	-1.03e+01	3.787e-01	9.800e+01	9.65e+00	feas	5.968e-15
9	OK	-1.62e+01	1.416e-01	9.800e+01	6.17e+00	feas	5.933e-15
10	OK	-1.82e+01	3.236e-02	9.800e+01	2.37e+00	feas	6.203e-15
11	OK	-1.98e+01	2.976e-01	9.800e+01	9.05e-01	feas	6.670e-15
12	OK	-2.52e+01	4.853e-02	9.800e+01	6.09e+00	feas	7.339e-15
13	OK	-2.98e+01	3.315e-02	9.800e+01	4.54e+00	feas	7.767e-15
14	OK	-3.44e+01	4.580e-02	9.800e+01	4.68e+00	feas	7.536e-15
15	OK	-3.82e+01	5.248e-02	9.800e+01	3.82e+00	feas	7.449e-15

Table 2: Algorithm Feas-Reset using the modified normal step on problem GAUSSELM

We conclude this section by briefly discussing another, even simpler modification of the Generic Algorithm 2.1 that would result in a feasible method. Instead of resetting the slack variables, as in Algorithm Feas-Reset, we could simply reject any trial iterate x_T that is infeasible. In a trust region method we would reduce the trust region and compute a new step; in a line search method we would simply backtrack. This algorithm would therefore be identical to Feas-Reset method except for the fact that the slack variables are never redefined.

For different reasons, the presence of equality constraints can also cause this algorithm to be very inefficient or even fail in some cases. Suppose for example that the current iterate

lies at the boundary of the feasible region defined by the inequality constraints and that the problem contains also equality constraints. Then it is possible for the direction generated by an interior point method to point towards the outside of the feasible region. In this case a line search method would fail when the trial steplength approached zero. If the current iterate lies near the boundary of the feasible region, the algorithm would generate very small steps.

A trust region version of this approach can also fail. (We observed such failures in our numerical experiments.) In analogy with Algorithm Feas-Reset, one could also try to improve upon this method by modifying the step computation. However, since this method is inherently different, a new type of step modification is in order. We have not explored this because Algorithm Feas-Reset, with its step modification, has proven to be a robust and efficient feasible method, but it is plausible that a careful modification of the approach just described could be just as effective.

6   Numerical Tests

To our knowledge there have been no studies comparing the performance of feasible versus infeasible interior point methods for nonlinear optimization. The algorithmic framework described in this paper is convenient for doing such tests as it allows us to easily switch between a feasible method and an infeasible method while keeping all other algorithmic features the same. One might expect that the flexibility provided by an infeasible method would be advantageous in certain circumstances allowing the iterates to take a more direct path to the solution through infeasible intermediate steps. However, by the same token, it is also conceivable that this additional freedom may allow for poor steps not permitted by feasible methods; this may make an infeasible code less efficient at times. To our surprise, the tests we report below do not indicate a clear superiority of either method and the potential advantages of each approach do not appear to be a major factor.

To compare the relative merits of these approaches we tested both feasible and infeasible versions of KNITRO on a set of 216 test problems from the CUTE collection. The problems, and some of their properties, are listed in the Appendix. The infeasible version follows the Generic Algorithm (Algorithm 2.1) with steps generated by KNITRO. The feasible version is based on Algorithm Feas-Reset (Algorithm 3.1) with steps determined by KNITRO modified to incorporate the new Cauchy step given by (5.30); the threshold value τ is chosen as $\tau = 10^{-4}$.

All the results were performed on a Sun Ultra 5/10 workstation with 384 MB of memory, in double precision Fortran. The termination test for the runs was the default KNITRO stopping test with a tolerance of 10^{-6}; this test is based on a normalized KKT condition.

The results are summarized in Figure 1, and are reported in detail in the Appendix. Figure 1 plots the ratio

$$\log_2 \frac{(\text{fevals Infeasible Alg})}{(\text{fevals Feas-Reset Alg})}, \tag{6.34}$$

where fevals denotes the number of function evaluations required to meet the stopping test. In the figure, the problems are arranged along the horizontal axis in decreasing order of the

absolute value of (6.34). We report results only for those problems for which both methods converged to the same solution. Also we only include problems where the feasible algorithm entered feasible mode (obtained an iterate where $g(x) \geq \tau e$).
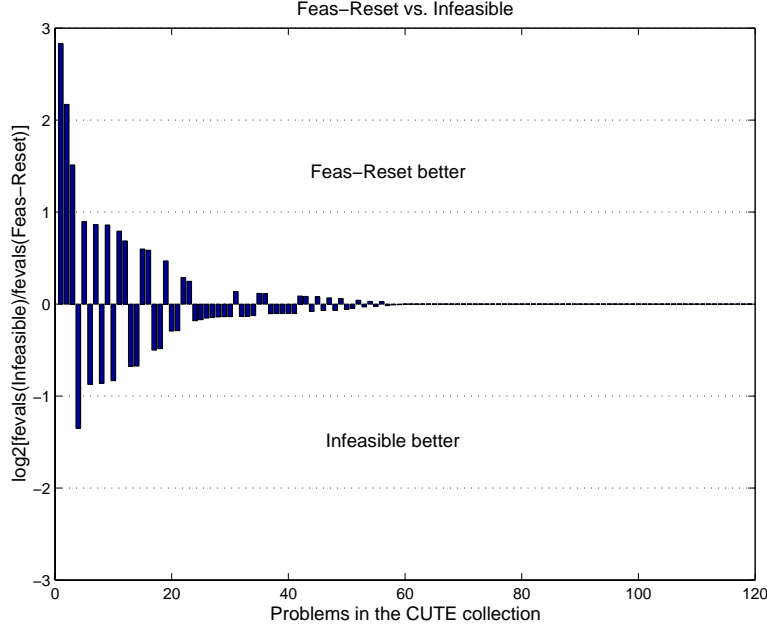
Figure 1: Comparison of infeasible and feasible versions of KNITRO on 119 problems in which both algorithms found the same solution and where the feasible version entered feasible mode. A bar on the upper half means that the feasible algorithm required fewer function evaluations. The logarithmic scale (6.34) was used.

The feasible algorithm solved 181 problems and the infeasible algorithm 179; they mainly failed on the same problems. ¿From this fact and Figure 1 we remark that for most of these problems, there is little difference between the two approaches. We observed that on 58 of the 216 problems Algorithm Feas-Reset never entered feasible mode. On these problems, therefore, the feasible and infeasible methods are identical. Moreover on a number of other problems there is little or no difference either because Algorithm Feas-Reset only entered feasible mode near the end of the run or the problems were solved too quickly or easily for there to be much difference. On three problems, the feasible algorithm was dramatically better than the infeasible method, but an examination of the runs does not suggest that this is due to the properties of the feasible iteration.

We interpret these results as being very positive because they indicate that staying feasible has no overall detrimental effect on the performance of the problems tested. They also suggest that the modified Cauchy step successfully overcomes the difficulties discussed in section 4. In particular, these results suggest that the feasible version of KNITRO represents a robust approach for handling problems in which feasibility with respect to

some or all of the inequality constraints must be retained.

We should point out that there is a feature of the KNITRO step computation that contributes to the lack of disparity between the feasible and infeasible methods compared in Figure 1. Although the standard version of KNITRO [5] is an infeasible algorithm, it makes use of a slack adjustment to accelerate convergence. KNITRO adjusts the slack variables at the end of each iteration (regardless of whether the iterate is feasible or not) according to the rule

$$s_{\mathrm{T}} \leftarrow \max(g(x_{\mathrm{T}}), s + d_s), \tag{6.35}$$

where the max function is applied component-wise; see step 5 of Algorithm I in [4]).

If one of the components of g is non-positive, this rule will not modify its associated slack variable. But suppose that one of the components of g, say g_j, satisfies $g_j(x_{\mathrm{T}}) > 0$ and that $(s + d_s)_j < g_j(x_{\mathrm{T}})$. Then the rule (6.35) increases the corresponding slack so that $(g(x_{\mathrm{T}}) - s_{\mathrm{T}})_j = 0$, while at the same time achieving a smaller barrier term value $-\mu \sum_{i \in \mathcal{I}} \ln(s_i)$ in the merit function. Note also that if one of the inequality constraints satisfies $g_j(x_{\mathrm{T}}) > 0$ and $(s + d_s)_j > g_j(x_{\mathrm{T}})$, then the value of the slack will not be changed by (6.35), even though by decreasing it we would obtain $(g(x_{\mathrm{T}}) - s_{\mathrm{T}})_j = 0$. We choose not to adjust the slacks in this case because this could cause an increase in the merit function.

This slack adjustment cannot cause the difficulties discussed in section 4.1 because it never gives rise to an increase in the merit function, and there is no need for modifying the step when equalities are present. The rule (6.35) has been analyzed in the context of the KNITRO algorithm in [4], and in practice has proven to be superior to using no slack adjustment scheme. In the numerical tests described above, the slack adjustment rule (6.35) is applied at every iteration of the infeasible method. In Algorithm Feas-Reset the slack adjustment is applied only before this method enters feasible mode (at which point it switches to the feasible slack reset scheme). Hence, the two methods benefit from the slack adjustment (6.35) where possible. We should stress that using this adjustment dampens the disparity between the two methods, since the slack adjustment is identical to the feasible reset when $g(x_{\mathrm{T}}) > s + d_s$. However, we use it in our comparison taking the view that all methods should be tested using their best implementation within the KNITRO framework.

7 Final Remarks

We have shown how infeasible slack-based interior methods can be transformed into feasible methods by using slack resets. Even though the modification is straightforward in most line search methods, care must be taken in trust region methods because, for problems containing both equality and inequality constraints, the slack reset can be harmful. We have given guidelines for the step computation so as to maintain feasibility with respect to inequality constraints, while improving the equality constraints. A specific normal step computation for the algorithm in KNITRO was developed and shown to be effective in practice.

The formulation (2.1) of a nonlinear program made it simple to transform infeasible interior methods into feasible methods since it identifies the inequality constraints and the slack variables. It is important not to erase this information during the problem formulation.

In particular, if all inequalities are transformed to equalities by introducing slacks, and the problem is given in the form

$$\min_x f(x) \quad \text{s.t.} \quad c(x) = 0, \quad x \geq 0,$$

without any distinction between the components of x that are slacks and those that are not, then the notion of feasibility with respect to the original inequality is lost, and it would be very inconvenient to apply the techniques discussed in this paper.

We should also point out that in all cases of slack resetting, the reset should take place before the evaluation of the merit function. If it is done after evaluating the merit function performance can be significantly degraded since the slack reset effectively changes the iterate, but the merit function does not have the opportunity to measure its quality.

The ideas presented in this paper are applicable to the case when only some of the inequality constraints must be honored. The slack reset in Algorithm Feas-Rest would only be applied to those inequality constraints that must be satisfied. Similarly, to obtain the improved normal component discussed in section 4, we must maintain linear feasibility for the inequality constraints that must be honored.

We conclude by pointing out that the merit function considered in this paper has the form (2.4), but this choice is not of particular importance. Many other choices of merit functions would be allowed in Algorithm Feas-Reset.

8   Acknowledgments

We would like to thank Guanghui Liu who performed some initial experiments with the feasible methods described in this paper. We also thank Marcelo Marazzi for valuable suggestions during this research, and Jose Luis Morales for suggesting the method for displaying the results in the figures.

9   Appendix

Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
AGG*	163	488	**(1)**	**(1)**	**(1)**	**(1)**
AIRPORT*	84	42	12	12	4.80e+04	4.80e+04
AUG2DCQP	850	400	27	27	3.24e+03	3.24e+03
AUG2DQP*	850	400	25	25	2.20e+03	2.20e+03
AUG3DCQP	156	27	20	20	3.93e+01	3.93e+01
AUG3DQP	156	27	20	20	4.18e+00	4.18e+00
BATCH*	48	73	385	385	2.61e+05	2.61e+05
BLOCKQP1	205	101	14	15	2.49e+00	2.49e+00
BLOCKQP2	205	101	12	12	-9.61e+01	-9.61e+01
BLOCKQP3	205	101	17	17	2.48e+00	2.48e+00
BLOCKQP4	205	101	22	22	-4.81e+01	-4.81e+01
						continued on next page

continued from previous page						
Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
BLOCKQP5	205	101	15	15	2.48e+00	2.48e+00
BLOWEYA	202	102	11	11	-4.56e-01	-4.56e-01
BLOWEYB	202	102	11	11	-3.05e-01	-3.05e-01
BLOWEYC	202	102	11	12	-3.05e-01	-3.05e-01
BRAINPC0	6907	6900	(2)	(2)	(2)	(2)
BRAINPC1	6907	6900	97	34	4.00e-04	4.27e-04
BRAINPC2	13807	13800	(2)	(2)	(2)	(2)
BRAINPC3	6907	6900	(2)	116	(2)	3.65e-01
BRAINPC4	6907	6900	(2)	(2)	(2)	(2)
BRAINPC5	6907	6900	(2)	(2)	(2)	(2)
BRAINPC6	6907	6900	200	366	3.46e-01	3.47e-01
BRAINPC7	6907	6900	(2)	(2)	(2)	(2)
BRAINPC8	6907	6900	(2)	(2)	(2)	(2)
BRAINPC9	6907	6900	(2)	314	(2)	3.51e-01
C-RELOAD	342	284	60	54	-1.03e+00	-1.03e+00
CLNLBEAM	303	200	21	25	3.50e+02	4.13e+02
CORKSCRW*	906	700	146	146	4.44e+01	4.44e+01
COSHFUN	61	20	(1)	957	(1)	-7.99e-01
CRESC100	6	200	(1)	671	(1)	5.69e-01
CRESC132*	6	2654	(4)	(4)	(4)	(4)
CRESC50*	6	100	(3)	(3)	(3)	(3)
CVXQP1	1000	500	11	11	1.09e+06	1.09e+06
CVXQP2	1000	250	13	13	8.20e+05	8.20e+05
CVXQP3	1000	750	13	12	1.36e+06	1.36e+06
DALLASL*	906	667	100	100	-2.00e+05	-2.00e+05
DALLASM*	196	151	49	49	-4.53e+04	-4.53e+04
DECONVC	61	1	70	99	1.11e-05	9.67e-07
DISCS*	36	66	(1)	(1)	(1)	(1)
DITTERT*	601	521	(3)	(3)	(3)	(3)
DIXCHLNV	100	50	19	19	1.44e-19	1.52e-19
DNIEPER	61	24	12	12	1.87e+04	1.87e+04
DRUGDIS	604	400	48	54	4.27e+00	4.26e+00
DRUGDISE	603	500	(1)	(4)	(1)	(4)
DUAL1	85	1	16	16	3.50e-02	3.50e-02
DUAL2	96	1	14	15	3.37e-02	3.37e-02
DUAL3	111	1	16	16	1.36e-01	1.36e-01
DUAL4	75	1	13	13	7.46e-01	7.46e-01
DUALC1*	9	215	66	66	6.16e+03	6.16e+03
DUALC2*	7	229	42	42	3.58e+03	3.58e+03
DUALC5*	8	278	21	21	4.27e+02	4.27e+02
DUALC8*	8	503	32	32	1.83e+04	1.83e+04
EG3	101	200	31	38	1.28e-01	1.28e-01
EIGMAXB	101	101	36	52	-7.79e-02	-4.72e-02
EIGMINA*	101	101	25	25	1.00e+00	1.00e+00
EIGMINB	101	101	36	52	7.79e-02	4.72e-02

continued from previous page						
Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
ELATTAR	7	102	59	65	1.43e-01	1.43e-01
EXPFITB	5	102	43	43	5.02e-03	5.02e-03
EXPFITC	5	502	188	178	2.33e-02	2.33e-02
FEEDLOC*	90	259	126	126	1.93e-08	1.93e-08
GAUSSELM	819	1926	257	244	-1.30e+01	-1.35e+01
GILBERT	1000	1	38	42	4.82e+02	4.82e+02
GMNCASE1	175	300	9	9	2.67e-01	2.67e-01
GMNCASE2	175	1050	10	10	-9.94e-01	-9.94e-01
GMNCASE3	175	1050	9	9	1.53e+00	1.53e+00
GMNCASE4*	175	350	27	27	5.95e+03	5.95e+03
GOFFIN	51	50	46	47	1.28e-05	1.28e-05
GOULDQP2	699	349	2	2	2.53e-04	2.53e-04
GOULDQP3	699	349	15	15	2.03e+00	2.03e+00
GPP	250	498	15	21	1.44e+04	1.44e+04
GRIDNETA	924	484	22	21	1.35e+02	1.35e+02
GRIDNETC	924	484	19	19	7.64e+01	7.64e+01
GRIDNETD	924	484	20	21	1.48e+02	1.48e+02
GRIDNETF	924	484	20	22	8.79e+01	8.79e+01
GRIDNETG	924	484	21	22	1.48e+02	1.48e+02
GRIDNETI	924	484	28	28	8.79e+01	8.79e+01
GROUPING*	100	125	(2)	(2)	(2)	(2)
HADAMARD*	401	1010	(3)	(3)	(3)	(3)
HAGER4	201	100	17	17	2.80e+00	2.80e+00
HAIFAL*	343	8958	(2)	(2)	(2)	(2)
HAIFAM	99	150	262	141	-4.50e+01	-4.50e+01
HANGING	300	180	41	45	-6.20e+02	-6.20e+02
HELSBY	1408	1399	2	2	5.77e+01	5.77e+01
HET-Z	2	1002	25	26	1.00e+00	1.00e+00
HIMMELBI	100	12	54	55	-1.74e+03	-1.74e+03
HUES-MOD	1000	2	(1)	408	(1)	3.48e+07
HUESTIS	1000	2	(1)	403	(1)	3.48e+10
HVYCRASH	404	300	38	32	-2.19e-01	-2.18e-01
HYDROELL	1009	1008	(1)	(1)	(1)	(1)
HYDROELM	505	504	(1)	(1)	(1)	(1)
HYDROELS	169	168	493	504	-3.56e+06	-3.57e+06
KISSING	211	903	190	105	8.43e-01	8.46e-01
KSIP	20	1001	34	34	5.76e-01	5.76e-01
LAKES	90	78	(3)	38	(3)	3.51e+05
LEAKNET	156	153	2	2	1.53e+01	1.53e+01
LHAIFAM	99	150	(3)	(3)	(3)	(3)
LINSPANH*	97	33	10	10	-7.70e+01	-7.70e+01
LISWET1*	403	400	8	8	1.04e+00	1.04e+00
LISWET10*	403	400	8	8	1.06e+00	1.06e+00
LISWET11*	403	400	7	7	1.04e+00	1.04e+00
LISWET12*	403	400	9	9	3.06e+01	3.06e+01

continued from previous page						
Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
LISWET2*	403	400	8	8	1.05e+00	1.05e+00
LISWET3*	403	400	8	8	1.06e+00	1.06e+00
LISWET4*	403	400	8	8	1.05e+00	1.05e+00
LISWET5*	403	400	8	8	1.05e+00	1.05e+00
LISWET6*	403	400	8	8	1.05e+00	1.05e+00
LISWET7*	403	400	8	8	1.05e+00	1.05e+00
LISWET8*	403	400	8	8	1.22e+00	1.22e+00
LISWET9*	403	400	9	9	2.83e+01	2.83e+01
LUBRIF*	751	500	(1)	(1)	(1)	(1)
MADSSCHJ	201	398	55	(1)	-4.99e+03	(1)
MANNE	300	200	64	9	-9.74e-01	-9.74e-01
MINC44	583	519	39	32	9.95e-04	1.00e-03
MINPERM	583	520	90	91	1.19e-03	9.37e-04
MODEL*	1542	38	(1)	(1)	(1)	(1)
MOSARQP1	900	30	11	11	-3.80e+02	-3.80e+02
MOSARQP2	900	30	10	10	-5.10e+02	-5.10e+02
MRIBASIS	36	55	50	55	1.82e+01	1.82e+01
NCVXQP1	1000	500	55	54	-7.16e+07	-7.16e+07
NCVXQP2	1000	500	50	49	-5.78e+07	-5.78e+07
NCVXQP3	1000	500	67	72	-3.08e+07	-3.08e+07
NCVXQP4	1000	250	50	48	-9.40e+07	-9.40e+07
NCVXQP5	1000	250	54	51	-6.63e+07	-6.63e+07
NCVXQP6	1000	250	90	87	-3.46e+07	-3.47e+07
NCVXQP7	1000	750	40	37	-4.34e+07	-4.34e+07
NCVXQP8	1000	750	55	59	-3.05e+07	-3.05e+07
NCVXQP9	1000	750	68	(3)	-2.15e+07	(3)
NGONE	100	1273	314	(1)	-6.43e-01	(1)
OET1	3	1002	36	24	5.38e-01	5.38e-01
OET2	3	1002	252	56	8.72e-02	8.72e-02
OET3	4	1002	23	23	4.51e-03	4.51e-03
OET4	4	1002	51	49	4.30e-03	8.56e-01
OET5	5	1002	122	311	2.66e-03	2.66e-03
OET6	5	1002	82	146	8.72e-02	8.72e-02
OET7	7	1002	100	(1)	2.09e-03	(1)
OPTCDEG2	302	200	30	34	2.37e+02	2.37e+02
OPTCDEG3	302	200	22	40	4.76e+01	4.76e+01
OPTMASS	610	505	26	15	-1.26e-01	-1.26e-01
ORTHREGE	999	662	168	268	1.32e+02	1.32e+02
ORTHREGF	680	225	37	23	9.19e+00	9.19e+00
POWELL20*	1000	1000	596	596	5.21e+07	5.21e+07
PRIMAL1	325	85	22	22	-3.50e-02	-3.50e-02
PRIMAL2	649	96	21	21	-3.37e-02	-3.37e-02
PRIMAL3	745	111	21	21	-1.36e-01	-1.36e-01
PRIMAL4	1489	75	22	22	-7.46e-01	-7.46e-01
PRIMALC1	230	9	17	17	-6.15e+03	-6.15e+03
						continued on next page

continued from previous page						
Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
PRIMALC2	231	7	14	14	-3.52e+03	-3.52e+03
PRIMALC5	287	8	19	19	-4.27e+02	-4.27e+02
PRIMALC8	520	8	27	29	-1.83e+04	-1.83e+04
PRODPL0*	60	29	38	38	5.88e+01	5.88e+01
PRODPL1*	60	29	31	31	3.57e+01	3.57e+01
PT	2	501	26	26	1.78e-01	1.78e-01
QPCBLEND*	83	74	9	9	-7.52e-03	-7.52e-03
QPCBOEI1*	384	351	(3)	(3)	(3)	(3)
QPCBOEI2*	143	166	233	233	8.17e+06	8.17e+06
QPCSTAIR*	467	356	269	269	6.20e+06	6.20e+06
QPNBLEND*	83	74	12	12	-9.12e-03	-9.12e-03
QPNBOEI1*	384	351	155	155	6.75e+06	6.75e+06
QPNBOEI2*	143	166	(3)	(3)	(3)	(3)
QPNSTAIR*	467	356	328	328	5.15e+06	5.15e+06
READING1	202	100	52	55	-1.60e-01	-1.60e-01
READING2	303	200	12	12	-1.25e-02	-1.25e-02
READING3	202	101	35	35	-1.53e-01	-1.53e-01
READING4	501	500	140	77	-2.89e-01	-2.89e-01
READING5	501	500	6	6	-2.30e-10	-2.40e-10
READING6	102	50	20	20	-1.45e+02	-1.45e+02
READING7	1002	500	41	41	-1.24e+03	-1.24e+03
READING9	402	200	15	15	-4.38e-02	-4.38e-02
ROTDISC	905	1081	(1)	(1)	(1)	(1)
SAWPATH	583	774	15	24	7.50e+02	7.50e+02
SINROSNB	1000	999	(1)	90	(1)	1.41e+00
SIPOW1	2	2000	17	16	-1.00e+00	-1.00e+00
SIPOW1M	2	2000	17	17	-1.00e+00	-1.00e+00
SIPOW2	2	2000	29	29	-1.00e+00	-1.00e+00
SIPOW2M	2	2000	34	34	-1.00e+00	-1.00e+00
SIPOW3	4	2000	25	25	5.35e-01	5.35e-01
SIPOW4	4	2000	29	29	2.72e-01	2.72e-01
SMBANK	117	64	23	(3)	-7.13e+06	(3)
SMMPSF*	720	263	365	365	1.03e+06	1.03e+06
SOSQP1	200	101	21	23	9.06e-05	7.96e-05
SOSQP2	200	101	21	21	-4.87e+01	-4.87e+01
SPANHYD*	97	33	18	18	2.40e+02	2.40e+02
SREADIN3	202	101	30	30	-1.53e-01	-1.53e-01
SSEBLIN	194	72	212	213	1.62e+07	1.62e+07
SSEBNLN*	194	96	74	74	1.89e+07	1.89e+07
SSNLBEAM*	303	200	23	23	3.43e+02	3.43e+02
STATIC3*	434	96	(1)	(1)	(1)	(1)
STCQP1	257	128	11	11	4.04e+03	4.04e+03
STCQP2	257	128	11	11	1.43e+03	1.43e+03
STEENBRA	432	108	147	148	1.70e+04	1.70e+04
STEENBRB	468	108	154	252	9.19e+03	9.08e+03
					continued on next page	

continued from previous page						
Problem	n	m	# of function evaluations		final function value	
			Infeasible	Feasible	Infeasible	Feasible
STEENBRC	540	126	527	489	2.76e+04	2.75e+04
STEENBRD*	468	108	**(3)**	**(3)**	**(3)**	**(3)**
STEENBRE*	540	126	**(3)**	**(3)**	**(3)**	**(3)**
STEENBRF	468	108	171	296	9.13e+03	8.99e+03
STEENBRG	540	126	390	616	2.75e+04	2.74e+04
STNQP1	257	128	10	10	-4.47e+03	-4.47e+03
STNQP2	257	128	11	11	-7.23e+03	-7.23e+03
SVANBERG	1000	1000	18	22	1.67e+03	1.67e+03
SWOPF	83	92	18	13	6.79e-02	6.79e-02
TFI1	3	101	45	50	5.33e+00	5.33e+00
TFI2*	3	101	12	12	1.14e+00	1.14e+00
TFI3	3	101	25	25	4.30e+00	4.30e+00
TRAINF	808	402	355	345	3.08e+00	3.08e+00
TRAINH	808	402	485	441	1.23e+01	1.23e+01
TRIMLOSS	142	75	78	43	9.06e+00	9.06e+00
UBH1	909	600	852	**(1)**	1.12e+00	**(1)**
UBH5	110	70	**(1)**	**(1)**	**(1)**	**(1)**
YAO*	202	200	9	9	2.02e+01	2.02e+01
YORKNET*	312	256	**(1)**	**(1)**	**(1)**	**(1)**
ZAMB2	1326	480	56	37	-4.14e+00	-4.14e+00
ZAMB2-10	270	96	40	40	-1.58e+00	-1.58e+00
ZAMB2-11	270	96	30	30	-1.12e+00	-1.12e+00
ZAMB2-8	138	48	28	28	-1.53e-01	-1.53e-01
ZAMB2-9	138	48	32	33	-3.55e-01	-3.55e-01
ZIGZAG*	604	500	**(1)**	**(1)**	**(1)**	**(1)**

Table 3: Comparison of Infeasible and Feasible versions of KNITRO in terms of function evaluations: **(1)** Maximum number of iterations (1000) reached; **(2)** Maximum allowable CPU time (60 minutes) reached; **(3)** Terminate because trust region radius, $\Delta \leq 10 \times \epsilon_{mach}$, where ϵ_{mach} is unit roundoff error; **(4)** Other abnormal termination. A "*" next to the problem name indicates that the Feasible version never entered feasible mode.

References

[1] P. Armand, J.-Ch. Gilbert, and S. Jan-Jégou. A feasible BFGS interior point algorithm for solving strongly convex minimization problems. *SIAM Journal on Optimization*, 11:199–222, 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. Robust trust region methods for constrained optimization. Third SIAM Conference on Optimization, Houston, Texas, May 1987.

[4] 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.

[5] 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, 2000.

[6] M. R. Celis, J. E. Dennis, and R. A. Tapia. A trust region strategy for nonlinear equality constrained optimization. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 71–82, Philadelphia, USA, 1985. SIAM.

[7] A. R. Conn, N. I. M. Gould, D. Orban, and Ph. L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical Programming*, 87(2):215–249, 2000.

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

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A primal-dual algorithm for minimizing a nonconvex function subject to bound and linear equality constraints. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Related Topics*, pages 15–50, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.

[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] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. J. Wiley and Sons, Chichester, England, 1968. Reprinted as *Classics in Applied Mathematics 4*, SIAM, Philadelphia, USA, 1990.

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

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

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

[15] C. T. Lawrence and A. L. Tits. Nonlinear equality constraints in feasible sequential quadratic programming. *Optimization Methods and Software*, 6(4):265–282, 1996.

[16] C. T. Lawrence and A. L. Tits. A computationally efficient feasible sequential quadratic programming algorithm. Technical report, Institute for Systems Research Technical Report TR 98-46, University of Maryland, College Park, 1998.

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

[18] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–114, London, 1970. Gordon and Breach.

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

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

[21] A. Vardi. A trust region algorithm for equality constrained minimization: convergence properties and implementation. *SIAM Journal on Numerical Analysis*, 22(3):575–591, 1985.

[22] H. Yamashita and H. Yabe. Superlinear and quadratic convergence of some primal-dual interior point methods for constrained optimization. *Mathematical Programming, Series A*, 75(3):377–397, 1996.

[23] H. Yamashita, H. Yabe, and T. Tanabe. A globally and superlinearly convergent primal-dual point trust region method for large scale constrained optimization. Technical report, Mathematical Systems, Inc., Sinjuku-ku, Tokyo, Japan, 1997.