DYNAMIC CONTROL OF INFEASIBILITY IN CONSTRAINED OPTIMIZATION

ROBERTO H. BIELSCHOWSKY * AND FRANCISCO A. M. GOMES †

Abstract. This paper describes a new algorithm for solving nonlinear programming problems with equality constraints. The method introduces the idea of using trust cylinders to keep the infeasibility under control. Each time the trust cylinder is violated, a restoration step is called and the infeasibility level is reduced. The radius of the trust cylinder has a nonincreasing update scheme, so eventually a feasible (and optimal) point is obtained. Global convergence of the algorithm is analyzed, as well as its numerical performance. The results suggest that the algorithm is promising.

Key words. nonlinear programming, constrained optimization, large-scale optimization.

AMS subject classifications. 65K05, 90C30

1. Introduction. In this article, we consider the equality constrained optimization problem

$$\begin{aligned} &\textit{minimize} \quad f(x) \\ &\textit{subject to} \quad h(x) = 0, \end{aligned} \tag{1.1}$$

where $f : I\!R^n \to I\!R$ and $h : I\!R^n \to I\!R^m$ are C^2 functions.

Probably, the very first strategy devised to solve this problem was the elimination method, a naive feasible point method that uses $h(x) = 0$ to eliminate some of the variables, in order to reduce (1.1) to a minimization problem without constraints.

Algorithms for (1.1) generating feasible iterates, without solving $h(x) = 0$ explicitly, go back to the early sixties, with methods usually classified either as Generalized Reduced Gradient (GRG) (see [32, 1, 2]), or as Projected Gradient (PG) [25, 26].

GRG algorithms have implementations which seem to be still competitive (see [7]). Variations of the PG method, including some strategy to relax feasibility in a controlled way, began to appear at the end of the sixties with the suggestive denomination of Sequential Gradient-Restoration Algorithm (SGRA) [18, 19]. See also [21, 23, 24]. More recently, Martínez introduced a new class of algorithm called *inexact restoration methods* [13, 14, 15, 16, 17], that also controls feasibility at each iteration.

Our approach to (1.1) has the flavour of a PG algorithm and could be characterized as a relaxed feasible point method, with a *dynamic control of infeasibility* (DCI). We look for a compromise between allowing a step big enough towards a solution of (1.1), in a direction approximately tangent to the restrictions $h(x) = 0$, and keeping infeasibility under control.

The main idea is to force the iterates $x^{(k)}$ to remain in *trust cylinders*

$$\mathcal{C}^{(k)} = \{x \in I\!R^n : \|h(x)\| \leq \rho^{(k)}\}.$$

where $\|.\|$ denotes the ℓ_2 norm. The dynamic control of infeasibility is kept defining the "radii" $\rho^{(k)}$ of the trust cylinders in such a way that

$$\rho^{(k)} = O(\|g_p(x^{(k)})\|) \tag{1.2}$$

*Departamento de Matemática, Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil. (rhbiel@ccet.ufrn.br)

†Departamento de Matemática Aplicada, IMECC, Universidade Estadual de Campinas, CP 6065, 13081-970, Campinas, SP, Brazil. (chico@ime.unicamp.br).

where $g_p(x)$ stands for the projected gradient, i.e., the orthogonal projection of the gradient $g(x) = \nabla f(x)$ onto the null space of the Jacobian of h, namely, $A = h'(x)$.

Given $x^{(k-1)}$, the k-th iteration begins with a *restoration step*, if necessary, in order to obtain a point $x_c = x_c^{(k)}$ and a radius ρ, such that

$$\|h(x_c)\| \leq \rho^{(k)} \tag{1.3}$$

and

$$\|x_c - x^{(k-1)}\| = O(\|h(x^{(k-1)}\|). \tag{1.4}$$

A radius $\rho = \rho^{(k)}$ satisfying (1.2) may be defined as $\rho = \nu\, n_p(x_c)\, \rho_{max}$, where

$$n_p(x_c) = \frac{\|g_p(x_c)\|}{\|g(x_c)\| + 1}$$

and $10^{-4} \leq \nu \leq 1$ and $\rho_{max} > 0$ are constants.

Given x_c in $\mathcal{C}^{(k)}$, the second part of the k^{th} iteration looks for a *horizontal step*, δ_t that provides a sufficient decrease for a quadratic approximation of f and guarantees that $x_+ = x_c + \delta_t$ remains in a bigger trust cylinder of radius 2ρ. An optional second order correction δ_{soc} may also be used to reduce the infeasibility, so $x^{(k)} = x_c + \delta_t + \delta_{soc}$.

Figure 1 sketches the vertical and the horizontal steps of a typical iteration.

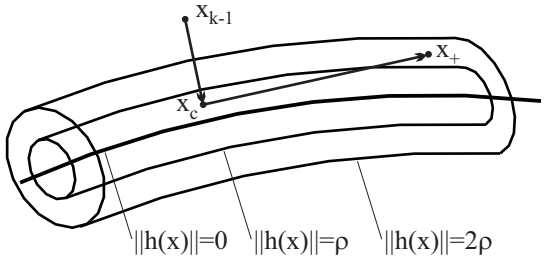

FIG. 1.1. *The step and the trust cylinders. x_c satisfies $\|h(x_c)\| < \rho$, while x_+ satisfies $\|h(x_+)\| < 2\rho$.*

One advantage of staying close to the feasible set is that one can expect the solutions of $x' = g_p(x)$, restricted to level sets close to the feasible one, to show a similar behaviour, in a generic sense (see [28]). In particular, a "good horizontal step" in a level set given by $h(x) = c$ is likely to be close to a "good horizontal step" in the feasible set given by $h(x) = 0$, if c is relatively small.

The parameter $\rho_{max} = \rho_{max}^{(k)}$ is non-increasing and is responsible for the trustability of the trust cylinders. It is decreased every time there is an evidence that the reduction of the Lagrangian obtained in the horizontal steps was menaced by a significant increase in the restoration step.

In the next section, we formalize the DCI algorithm. In section 3, a global convergence result for the algorithm is presented. Section 4 contains some preliminary numerical results. Finally, some conclusions and lines for future work are included in section 5.

2. The DCI Algorithm. In this section, we depict a typical iteration of our main algorithm. As usual, we use the Lagrangian function, defined as

$$L(x, \lambda) = f(x) + \lambda^T h(x),$$

to evaluate the algorithm behaviour. In fact, the control of the trust cylinder radius is based on the the variation of the Lagrangian at $x_c^{(k)}$, given by

$$\Delta L_c^{(k)} = L(x_c^{(k)}, \lambda^{(k)}) - L(x_c^{(k-1)}, \lambda^{(k-1)}).$$

Since our algorithm divides the step into two components, one "vertical" and one "horizontal", this variation is also split according to

$$\Delta L_c^{(k)} = \Delta L_H^{(k-1)} + \Delta L_V^{(k)}, \tag{2.1}$$

where

$$\begin{aligned} \Delta L_H^{(k)} &= L(x^{(k)}, \lambda^{(k)}) - L(x_c^{(k)}, \lambda^{(k)}), \\ \Delta L_V^{(k)} &= L(x_c^{(k)}, \lambda^{(k)}) - L(x^{(k-1)}, \lambda^{(k-1)}). \end{aligned}$$

In the vertical step of the algorithm, we seek for a point x_c that satisfies (1.3) and (1.4). One way to cope with this problem is to successively solve the least squares problem

$$\begin{aligned} &\textit{minimize} \quad \|h(x) + A(x)d\|^2 \\ &\textit{subject to} \quad \|d\| \leq \Delta_{VS}, \end{aligned} \tag{2.2}$$

until an acceptable point is found. Here, $A(x)$ is the Jacobian of the constraints at x and $\Delta_{VS} > 0$ is the trust region radius used solely to compute this step.

In the horizontal step we solve the quadratic programming problem

$$\begin{aligned} &\textit{minimize} \quad q(\delta) = g(x_c)^T\delta + \frac{1}{2}\delta^T B\delta \\ &\textit{subject to} \quad A(x_c)\delta = 0, \\ &\qquad\qquad \|\delta\| \leq \Delta, \end{aligned} \tag{2.3}$$

where B is a symmetric approximation for the Hessian of the Lagrangian and $\Delta > 0$ is the trust region radius.

We suppose that, at the beginning of the k^{th} iteration, the previous approximate solution, $x^{(k-1)}$, and the Lagrange multipliers estimate, $\lambda^{(k-1)}$, are available. Besides, we also suppose known the upper limit for the trust cylinder radius, ρ_{max}, the Lagrangian function at some previous iteration j, $L_{ref} = L(x_c^{(j)}, \lambda^{(j)})$, the horizontal variation of the Lagrangian, $\Delta L_H^{(k-1)}$, and the trust region radii, $\Delta_{VS} \geq \Delta_{min} > 0$ and $\Delta \geq \Delta_{min} > 0$.

ALGORITHM 2.1. *The k^{th} iteration of the DCI method.*

1. *Vertical step:*
1.1. $x_c = x^{(k-1)}$;
1.2. Choose an approximate value for ρ.
1.3. REPEAT
1.3.1. Find x_c such that $\|h(x_c)\| \leq \rho$.
1.3.2. $A \leftarrow h'(x_c)$; $g_p \leftarrow g_p(x_c)$; $n_p \leftarrow \|g_p(x_c)\|/(\|g(x_c)\| + 1)$.
1.3.3. Choose $\rho \in [10^{-4} n_p \rho_{max}, n_p \rho_{max}]$.
1.4. UNTIL $\|h(x_c)\| \leq \rho$,
1.5. Compute λ_+.

2. *Convergence test:*
2.1. IF ($\rho = 0$) OR ($n_p = 0$ AND $\|h(x_c)\| = 0$),
2.1.1. QUIT (x_c is a stationary point).
3. ρ_{max} *update:*
3.1. $\Delta L_V^{(k)} \leftarrow L(x_c, \lambda_+) - L(x^{(k-1)}, \lambda^{(k-1)})$.
3.2. IF $\Delta L_V^{(k)} \geq \frac{1}{2}[L_{ref} - L(x^{(k-1)}, \lambda^{(k-1)})]$,
3.2.1. $\rho_{max} \leftarrow \rho_{max}/2$.
3.3. IF $\Delta L_V^{(k)} > -\frac{1}{2}\Delta L_H^{(k-1)}$,
3.3.1. $L_{ref} \leftarrow L(x_c, \lambda_+)$.
4. *Horizontal step:*
4.1. REPEAT
4.1.1. Compute the Cauchy step δ_{CP}, solution of
$$\begin{array}{ll} minimize & q(\mu g_p) \\ subject\ to & \|\mu g_p\| \leq \Delta,\ \mu \in [0, \infty). \end{array}$$
4.1.2. Compute a trial step δ_t such that
$q(\delta_t) \leq q(\delta_{CP})$,
$\|\delta_t\| \leq \Delta$, and
$A\delta_t = 0$.
4.1.3. Optionally, compute a second order correction δ_{soc}.
4.1.4. $\delta_+ \leftarrow \delta_t + \delta_{soc}$; $x_+ \leftarrow x_c + \delta_+$.
4.1.5. $\Delta L_H^{(k)} \leftarrow L(x_+, \lambda_+) - L(x_c, \lambda_+)$; $r \leftarrow \Delta L_H^{(k)}/q(\delta_t)$.
4.1.6. IF ($\|h(x_+)\| > 2\rho$) OR ($r < \eta_1$),
4.1.6.1. $\Delta \leftarrow \alpha_R \Delta$.
4.1.7. ELSE IF $\Delta L_H^{(k)} > \eta_2 q(\delta_t)$,
4.1.7.1. $\Delta \leftarrow \alpha_I \Delta$.
4.2. UNTIL ($\|h(x_+)\| \leq 2\rho$) AND ($r \geq \eta_1$).
5. *Approximate solution update:*
5.1. $x^{(k)} \leftarrow x_+$; $\lambda^{(k)} \leftarrow \lambda_+$; $k \leftarrow k + 1$.
5.2. Choose $\Delta \geq \Delta_{min}$.

In Algorithm 2.1, we suppose that the restoration step 1.3.1 will always succeed. Obviously, this may not occur, since problem (1.1) may be infeasible. Therefore, some termination criterion need to be defined to prevent the algorithm to get stuck on this step.

Most of the constants used in algorithm 2.1 are explicitly shown above, so the reader does not need to guess the meaning of several obscure greek letters. We do prefer to write $\|h(x_+)\| > 2\rho$ instead of $\|h(x_+)\| > \zeta\rho$, for example, to make clear that, in steps 4.1.6 and 4.2, we are considering a larger trust cylinder. Naturally, the algorithm will also work if we use $\zeta = 3$, although this modification will slightly affect the proofs of some lemmas presented in the next section. Only four constants that control the behaviour of the trust region method used to compute the horizontal step were not specified: $0 < \eta_1 \leq 1/2$, $\eta_2 \geq \eta_1$, $0 < \alpha_R < 1$ and $\alpha_I \geq 1$. Possible values for these parameters are $\eta_1 = 10^{-3}$, $\eta_2 = 0.7$, $\alpha_R = 0.25$ and $\alpha_I = 2.5$.

The global convergence of DCI will be guaranteed, under reasonable assumptions, by a typical *sufficient decrease* argument for the Lagrangian function evaluated at $x_c^{(k)}$. The variation of the Lagrangian between two successive iterations is given by (2.1). To prevent the decrease of the Lagrangian obtained at the horizontal step to be destroyed by the restoration, ${\rho_{max}}^{(k)}$ is decreased in step 3 of DCI every time

$\Delta L_V^{(k)}$ is larger than a fraction of the difference between the Lagrangian at the current iteration and a reference value L_{ref}, fixed in some previous iteration j. If the increase in $\Delta L_V^{(k)}$ menaces significantly the decrease in the Lagrangian obtained since iteration j, ρ_{max} is divided by two and L_{ref} is updated. L_{ref} is also updated every time $\Delta L_V^{(k)} > -\frac{1}{2}\Delta L_H^{(k-1)}$. The main argument to guarantee global convergence establishes, under suitable assumptions, the existence of *enough normal space (ENS)*, dynamically calibrated for horizontal steps of reasonable size, in the sense that ${\rho_{max}}^{(k)}$ remains bounded away from zero, unless $\liminf(\|g_p(x_c^{(k)})\|) = 0$.

One possible choice for λ_+ could be the least squares multipliers evaluated at $x_c^{(k)}$, defined as

$$\lambda_{LS}(x_c) = argmin\{\|A(x_c)^T\lambda + g(x_c)\|\} = -(A(x_c)A(x_c)^T)^{-1}A(x_c)g(x_c) \tag{2.4}$$

We say that x is a regular point of h if the Jacobian $A(x) = h'(x)$ is of maximal rank, in which case λ_{LS} can be computed. Notice that

$$g_p(x) = g(x) + A(x)^T\lambda_{LS}(x), \tag{2.5}$$

at every regular point of h.

3. Global convergence. The global convergence analysis of the DCI algorithm are based on the following hypothesis:

H1 (Differentiability): f and h are C^2.

H2 (Compacity): The generated sequences $\{{x_c}^{(k)}\}$ and $\{x^{(k)}\}$, the Hessian approximations $B^{(k)}$ and the multipliers $\{\lambda^{(k)}\}$ remain uniformly bounded.

H3 (Regularity and restoration): The restoration never fails and $Z = \{{x_c}^{(k)}\}$ remains far from the singular set of h, in the sense that h is regular in the closure of Z. Equivalently, $\{\|h'({x_c}^{(k)})^T h'({x_c}^{(k)})^{-1}\|\}$ remains uniformly bounded. Also, if the generated sequence $\{{x_c}^{(k)}\}$ is infinite, it satisfies

$$\|{x_c}^{(k+1)} - x^{(k)}\| = O(\|h(x^{(k)})\|). \tag{3.1}$$

H4 (Second order correction): $\|{\delta_{soc}}^{(k)}\| = O(\|{\delta_t}^{(k)}\|^2)$.

Supposing that H1 holds, we can assure that the remaining hypothesis will hold if, for example, the feasible set $\mathcal{H}_0$ is compact, regular (i.e. $h'(x)$ is of maximal rank on it) and $x^{(0)}$ is feasible. In this case, for some initial $\rho_{max}^{(0)}$ sufficiently small, we can still keep $h'(x)$ with maximal rank and standard algorithms for restoration, like the Gauss-Newton method, will guarantee (3.1). It would also be the case if we replace the compacity property of $\mathcal{H}_0$ by adequate properties on f, like asking f to satisfy $\lim_{x\to\infty} f(x) = \infty$. In such situations, H2-H4 can be guaranteed by construction.

From now on we assume that the sequences $\{x_c^{(k)}\}$ and $\{x^{(k)}\}$, generated by DCI, satisfy H1-H4. Besides, when we say that a number is a constant, we mean that it can be used for all k, in one specific sequence generated by DCI.

The main result of this section, presented in Theorem 3.4, is based on three lemmas. The first one establishes that, under H1-H4, each iteration succeeds, so the Lagrangian is sufficiently decreased.

LEMMA 3.1. *If $x_c^{(k)}$ is not a stationary point for (1.1), then x_+ is eventually accepted in step 4 of DCI. Moreover, we can define positive constants ξ_1, ξ_2 and ξ_3 such that*

$$\begin{aligned} -\Delta L_H^{(k)} &= L(x_c^{(k)}, \lambda^{(k)}) - L(x^{(k)}, \lambda^{(k)}) \\ &\geq \xi_1\|g_p(x_c^{(k)})\| \min\{\xi_2\|g_p(x_c^{(k)})\|, \xi_3(\rho^{(k)})^{1/2}, \Delta_{min}\} \end{aligned} \tag{3.2}$$

Proof. To simplify the notation we will omit here the superscript (k). Suppose x_c is not stationary for (1.1). Let $x_+ = x_c + \delta_+ = x_c + \delta_t + \delta_{soc}$ be a candidate obtained in step 4 of the k^{th} iteration of the DCI algorithm, and let λ_+ be the corresponding multiplier. From H2, we can define a positive constant ξ_B such that $\|B\| \leq \xi_B$. Since $A\delta_t = 0$, we have $q(\delta_t) = g_p(x_c)^T\delta_t + \frac{1}{2}\delta_t^T B\delta_t$. Combining this with H1, H2 and H4, and taking into account that $q(\delta) \leq 0$, we can define positive constants $\overline{\xi}_1$, $\overline{\xi}_2$ and $\overline{\xi}_3$, such that

$$\|h(x_+) - h(x_c)\| \leq \overline{\xi}_1\|\delta_t\|^2 \tag{3.3}$$

and

$$\Delta L_H^+ = L(x_+, \lambda_+) - L(x_c, \lambda_+) \leq g_p(x_c)^T\delta_t + \overline{\xi}_2\|\delta_t\|^2 \leq q(\delta_t) + \overline{\xi}_3\|\delta_t\|^2. \tag{3.4}$$

Because δ_{CP}, defined in step 4.1.1 of DCI, is a Cauchy step tangent to the restrictions, we have

$$\|\delta_{CP}\| \geq \min\left\{\frac{\|g_p(x_c)\|}{\xi_B}, \Delta\right\} \tag{3.5}$$

and

$$q(\delta_{CP}) \leq \frac{1}{2}g(x_c)^T\delta_{CP} \leq -\frac{1}{2}\|g_p(x_c)\|\min\left\{\frac{\|g_p(x_c)\|}{\xi_B}, \Delta\right\} \tag{3.6}$$

Let us define

$$\overline{\Delta} = \min\{\overline{\xi}_4\|g_p(x_c)\|, \overline{\xi}_5\sqrt{\rho}, \Delta_{min}\}. \tag{3.7}$$

where $\overline{\xi}_4 = \min\left\{\frac{1}{2\xi_B}, \frac{1}{8\overline{\xi}_3}\right\}$ and $\overline{\xi}_5 = \frac{1}{2\sqrt{\overline{\xi}_1}}$. It is easy to deduce from (3.5) that if

$$\Delta \leq 2\overline{\Delta} \tag{3.8}$$

then $\|\delta_{CP}\| = \Delta$. Now, using (3.8), (3.7), the step 4.1.2 of the DCI algorithm and (3.6), we obtain

$$\overline{\xi}_3\|\delta_t\|^2 \leq \overline{\xi}_3\Delta^2 \leq 2\overline{\xi}_3\overline{\xi}_4\|g_p(x_c)\|\Delta \leq \frac{1}{4}\|g_p(x_c)\|\Delta \leq \frac{1}{2}|q(\delta_{CP})| \leq \frac{1}{2}|q(\delta_t)|. \tag{3.9}$$

From (3.9) and (3.4), we get $\Delta L_H^+ \leq \frac{1}{2}q(\delta_t)$, which implies that $r = \Delta L_H^+/q(\delta_t) \geq \eta_1$. Now, from (3.3), (3.8), (3.7), the definition of $\overline{\xi}_5$ and the fact that $\|h(x_c)\| \leq \rho$, we have

$$\|h(x_+)\| \leq \rho + \overline{\xi}_1\|\delta_t\|^2 \leq \rho + 4\overline{\xi}_1\overline{\Delta}^2 \leq \rho(1 + 4\overline{\xi}_1\overline{\xi}_5^2) = 2\rho.$$

Therefore, whenever $\Delta \leq 2\overline{\Delta}$, both conditions stated at step 4.2 of the algorithm are satisfied, so $\delta_+ = \delta_t + \delta_{soc}$ is accepted by DCI.

Since δ_t is accepted, we also have

$$\Delta L_H^+ \leq \eta_1 q(\delta_t) \leq \eta_1 q(\delta_{CP}). \tag{3.10}$$

Combining (3.10), (3.6), (3.8) and (3.7), we obtain (3.2). □

Our second lemma establishes that, between successive iterations without changes in ρ_{max}, the Lagrangian decreases proportionally to the descent in the corresponding horizontal steps.

LEMMA 3.2. *If $L_{ref} = L_c^{(k)}$ and $\rho_{max}^{(k+1)} = \rho_{max}^{(k+2)} = \ldots = \rho_{max}^{(k+j)}$, for $j \geq 1$, then*

$$L_c^{(k+j)} - L_c^{(k)} = \sum_{i=k+1}^{k+j} \Delta L_c^{(i)} \leq \frac{1}{4} \sum_{i=k}^{k+j-1} \Delta L_H^{(i)} \tag{3.11}$$

Proof. Let suppose that L_{ref} doesn't change between iterations $k+1$ and $k+j_1-1$, where $0 < j_1 \leq j+1$. In this case, by the criterion defined in step 3.3 of the algorithm, we have

$$L_c^{(k+j_1-1)} - L_c^{(k)} = \sum_{i=k+1}^{k+j_1-1} (\Delta L_V^{(i)} + \Delta L_H^{(i-1)}) \leq \frac{1}{2} \sum_{i=k}^{k+j_1-2} \Delta L_H^{(i)}. \tag{3.12}$$

If L_{ref} is changed at iteration $(k+j_1)$, then, by the condition in step 3.3, the supposition that ρ_{max} stays unchanged at this iteration (so the inequality at step 3.2 is not satisfied) and the fact that $\Delta L_H \leq 0$, we have

$$\begin{aligned} L_c^{(k+j_1)} - L_c^{(k)} &= \Delta L_V^{(k+j_1)} + L(x^{(k+j_1-1)}, \lambda^{(k+j_1-1)}) - L_c^k \\ &\leq \frac{1}{2}(L(x^{(k+j_1-1)}, \lambda^{(k+j_1-1)}) - L_c^k) \\ &= \frac{1}{2}(\Delta L_H^{(k+j_1-1)} + L_c^{(k+j_1-1)} - L_c^{(k)}) \leq \frac{1}{4} \sum_{i=k}^{k+j_1-1} \Delta L_H^{(i)}. \end{aligned} \tag{3.13}$$

If $j_1 = j$, then (3.12) and (3.13) imply (3.11). On the other hand, if L_{ref} is also updated at iterations $k+j_2, \ldots, k+j_s$, where $j_s \leq j$, then applying the same procedure described above several times and defining $j_0 = 0$ we obtain

$$L_c^{(k+j)} - L_c^{(k)} = \sum_{i=1}^{s} [L_c^{(k+j_i)} - L_c^{(k+j_{i-1})}] + L_c^{(k+j)} - L_c^{(k+j_s)} \leq \frac{1}{4} \sum_{i=k}^{k+j-1} \Delta L_H^{(i)}.$$

□

Our third lemma establishes the existence of enough normal space in the trust cylinders $\mathcal{C}^{(k)}$ to guarantee that the Lagrangian can be sufficiently decreased.

LEMMA 3.3. *If DCI generates an infinite sequence $\{x^{(k)}\}$, then*

i) There are positive constants ξ_5 and ξ_ρ such that, whenever

$$\rho_{max}^{(k)} \leq \min\{\xi_\rho \|g_p(x_c)^k\|, \xi_5\}, \tag{3.14}$$

$\rho_{max}^{(k)}$ don't change in the k^{th} iteration. Furthermore, if $\liminf \|g_p(x_c^{(k)})\| > 0$ then there exists $k_0 > 0$ such that, for every k,

$$\rho_{max}^{(k)} \geq \rho_{max}^{(k_0)}. \tag{3.15}$$

ii) If the horizontal step and the vector of Lagrange multipliers satisfy

$$\|x^{(k)} - x_c^{(k)}\| = O(\|g_p(x_c^{(k)})\|) \tag{3.16}$$

$$\|\lambda^{(k)} - \lambda_{LS}(x_c^{(k)})\| = O(\|g_p(x_c^{(k)}))\|) \tag{3.17}$$

then (3.15) is satisfied, no matter what is the value of $\liminf \|g_p(x_c^{(k)})\|$. *In other words,* $\rho_{max}^{(k)}$ *remain bounded away from zero.*

Proof. In the proof of the first part of this lemma, we will show that, for $\rho_{max}^{(k)}$ sufficiently small, if $\liminf \|g_p(x_c^{(k)})\| > 0$, then $|\Delta L_H^{(k)}|$ is bigger than a fraction of $\sqrt{\rho^{(k)}}$, while, in a restoration, the Lagrangian can't grow asymptotically that fast because $\Delta L_V^{(k+1)} = O(\rho^{(k)})$.

From Lemma 3.1 and the choice of $\rho^{(k)}$ in step 1.3.3 of the algorithm, there exist positive constants $\overline{\xi_1}$, $\overline{\xi_2}$, $\overline{\xi_3}$, such that

$$\begin{aligned} -\Delta L_H^{(k)} &= |L(x^{(k)}, \lambda^{(k)}) - L(x_c^{(k)}, \lambda^{(k)})| \\ &\geq \|g_p(x_c^{(k)})\| \min\{\overline{\xi_1}\|g_p(x_c^{(k)})\|, \overline{\xi_2}{\rho_{max}^{(k)}}^{1/2}\|g_p(x_c^{(k)})\|^{1/2}, \overline{\xi_3}\Delta_{min}\} \end{aligned} \tag{3.18}$$

Assumptions H1 and H2 together ensure that L is Lipschitz on the iterates. In particular, we have

$$|L(x_c^{(k+1)}, \lambda^{(k+1)}) - L(x^{(k)}, \lambda^{(k+1)})| = O(x_c^{(k+1)} - x^{(k)}). \tag{3.19}$$

Combining (3.19) with H3, we obtain

$$\begin{aligned} |\Delta L_V^{(k+1)}| &= |L(x_c^{(k+1)}, \lambda^{(k+1)}) - L(x^{(k)}, \lambda^{(k)})| \\ &\leq |L(x_c^{(k+1)}, \lambda^{(k+1)}) - L(x^{(k)}, \lambda^{(k+1)})| + |(\lambda^{(k+1)} - \lambda^{(k)})^T h(x^{(k)})| \\ &= O(\|h(x^{(k)})\|) = O(\rho^{(k)}). \end{aligned}$$

Now, supposing that the vertical step succeeds, so $x_c^{(k+1)}$ satisfies

$$\|h(x_c^{(x+1)})\| \leq \rho^{(k+1)} \leq \|g_p(x_c^{(k)})\|\rho_{max}^{(k)}, \tag{3.20}$$

then there exists $\xi_4 > 0$ such that

$$|\Delta L_V^{(k+1)}| \leq \xi_4 \rho_{max}^{(k)} \|g_p(x_c^{(k)})\|. \tag{3.21}$$

Let us define $\xi_\rho = \frac{1}{2}\min\{\overline{\xi_1}/\xi_4, (\overline{\xi_2}/\xi_4)^2\}$ and $\xi_5 = \Delta_{min}\overline{\xi_3}/(2\xi_4)$. If $\rho_{max}^{(k)}$ satisfies (3.14) for these values of ξ_ρ and ξ_5, then

$$|\Delta L_V^{(k+1)}| < -\frac{1}{2}\Delta L_H^{(k)}. \tag{3.22}$$

Therefore, ρ_{max} is not updated in step 3.2 of DCI, i.e. $\rho_{max}^{(k+1)} = \rho_{max}^{(k)}$. This proves the first part of the lemma.

To prove the second part of the lemma we will begin observing that H1-H3 imply that $\lambda_{LS}(x)$ and $g_p(x)$ are well defined and of class C^1 in a compact neighbourhood of $\overline{\mathcal{Z}}$, the closure of $Z = \{{x_c}^k\}$. Therefore, $\lambda_{LS}(x)$ and $g_p(x)$ are Lipschitz in $\overline{\mathcal{Z}}$, so we have

$$\|\lambda_{LS}(x_c^{(k+1)}) - \lambda_{LS}(x_c^{(k)})\| = O(\|x_c^{(k+1)} - x_c^{(k)}\|) \tag{3.23}$$

and

$$\|g_p(x_c^{(k+1)}) - g_p(x_c^{(k)})\| = O(\|x_c^{(k+1)} - x_c^{(k)}\|). \tag{3.24}$$

From (3.20), (3.1), (3.16) and (3.24) we get

$$\|x_c^{(k+1)} - x_c^{(k)}\| = O(\|g_p(x_c^{(k)})\|) \tag{3.25}$$

and

$$\|g_p(x_c^{(k+1)})\| = O(\|g_p(x_c^{(k)})\|). \tag{3.26}$$

Let us now decompose $\Delta L_V^{(k+1)}$ into a sum of four terms:

$$\begin{aligned}\Delta L_V^{(k+1)} &= L(x_c^{(k+1)}, \lambda^{(k+1)}) - L(x^{(k)}, \lambda^{(k)}) \\ &= [L(x_c^{(k+1)}, \lambda_{LS}(x_c^{(k+1)})) - L(x^{(k)}, \lambda_{LS}(x_c^{(k+1)}))] + \\ &\quad [\lambda^{(k+1)} - \lambda_{LS}(x_c^{(k+1)})]^T h(x_c^{(k+1)}) + \\ &\quad [\lambda_{LS}(x_c^{(k+1)}) - \lambda_{LS}(x_c^{(k)})]^T h(x^{(k)}) + \\ &\quad [\lambda_{LS}(x_c^{(k)}) - \lambda^{(k)}]^T h(x^{(k)}).\end{aligned}$$

By doing a Taylor series expansion and using H2, (3.20), (3.1) and (3.26), we obtain

$$\begin{aligned}&L(x_c^{(k+1)}, \lambda_{LS}(x_c^{(k+1)})) - L(x^{(k)}, \lambda_{LS}(x_c^{(k+1)})) = \\ &g_p(x_c^{(k+1)})^T(x_c^{(k+1)} - x^{(k)}) + O(\|x_c^{(k+1)} - x^{(k)}\|^2) = \\ &O(\|g_p(x_c^{(k)})\|\rho^{(k)}).\end{aligned}$$

Thus, the first term of (3.27) is $O(\|g_p(x_c^{(k)})\|\rho^{(k)})$. From (3.1), (3.16), (3.17), (3.23) and (3.26), we can also prove that the remaining three terms are $O(\|g_p(x_c^{(k)})\|\rho^{(k)})$. Therefore, (3.20) implies that there exists a positive constant ξ_5 such that

$$\Delta L_V^{(k+1)} \leq \xi_5 \rho_{max}^{(k)} \|g_p(x_c^{(k)})\|^2.$$

Using this inequality together with (3.18), we can ensure that $\rho_{max}^{(k)}$ will not change anymore if

$$\rho_{max}^{(k)} < \min\{\overline{\xi_1}/(2\xi_5), \overline{\xi_2}^2/(2\xi_5^2 G), \overline{\xi_3}\Delta_{min}/(2\xi_5 G)\},$$

where $G = \sup\{\|\nabla f(x^{(k)})\|^2 : 1 \leq k \leq \infty\}$. This completes the proof of the Lemma. □

We say that a point x is stationary for (1.1), i.e. it satisfies the KKT conditions for the problem, if $h(x) = 0$ and $g_p(x) = 0$. The next theorem states that, under H1-H4, the sequence $\{x_c{}^{(k)}\}$ generated by the DCI algorithm has stationary points for (1.1) in its accumulation set. Some additional conditions are defined to ensure that every accumulation point results stationary for (1.1).

THEOREM 3.4. *Under H1-H4, either DCI stops in a stationary point for (1.1), in a finite number of iterations, or generates a sequence with stationary points in its accumulation set. Besides, if we impose the horizontal step and the Lagrange multipliers to satisfy (3.16) and (3.17), then every accumulation point of* $x_c{}^(k)$ *is stationary for (1.1).*

Proof. We begin claiming that if there is a k_0 such that $\rho_{max}{}^{(k)} \geq \rho_{max}{}^{(k_0)}$ for every $k \geq k_0$, then

$$\lim_{k\to\infty} g_p(x_c^{(k)}) = 0. \tag{3.27}$$

To prove this result, let's suppose, by contradiction, that $\rho_{max}^{(k)} \geq \rho_{max}^{(k_0)}$ but $\|g_p(x_c)^{(k_\ell)})\| \geq b > 0$ for some infinite subsequence $\{k_\ell\}$. In this case, from (3.11) and (3.18), we have, for $k > k_0$,

$$L(x_c^{(k)}, \lambda^{(k)}) - L(x_c^{(k_0)}, \lambda^{(k_0)}) = \sum_{i=k_0+1}^{k} \Delta L_c^{(i)} \leq \frac{1}{4} \sum_{i=k_0}^{k-1} \Delta L_H^{(i)} \leq -n_k \bar{\xi}_1,$$

where

$$\bar{\xi}_1 = \frac{1}{4} \xi_1 b \min\{\xi_2 b, \xi_3 (\rho_{max}^{(k_0)})^{1/2}, \Delta_{min}\} > 0$$

and n_k is the number of iterations, between k_0 and $k-1$, for which $\|g_p(x_c^{(k_\ell)})\| \geq b > 0$. Because we are supposing that there is an infinite number of such indices, we obtain

$$\lim_{k \to \infty} L(x_c^{(k)}, \lambda^{(k)}) = -\infty,$$

which contradicts H1-H2, validating our claim.

Let us suppose now, for the purpose of obtaining another contradiction, that $\liminf(\|g_p(x_c^{(k)})\|) \neq 0$. In this case, Lemma 3.3 together with our claim above imply (3.27). This proves $\liminf(\|g_p^{(k)}\|) = 0$.

For the second part of the theorem, let us assume that (3.16) and (3.17) apply. In this case, Lemma 3.3 ensures that ${\rho_{max}}^{(k)}$ remains bounded away from zero. As we saw above, this implies (3.27), completing the proof. □

4. Numerical experience. The success of an algorithm is based not only on its theoretical convergence results, but also on its practical behaviour. In this section, we present one possible implementation for the DCI algorithm, along with the numerical results obtained applying it to some problems from the CUTEr collection [10].

We do not claim we have implemented the ultimate version of the algorithm. On the contrary, our implementation is quite simple and should be improved in order to compete with modern commercial codes. Our only purpose is to show that the algorithm can successfully solve medium-sized equality constrained problems. Some hints on how to improve the code are given in the next section.

4.1. A practical implementation of the algorithm. We begin the detailed description of algorithm explaining how the vertical and the horizontal steps can be implemented. After that, we discuss how to solve the linear systems that appear when computing these steps. Finally, we present a second order correction used to reduce the infeasibility after applying the horizontal step.

4.1.1. Vertical step. Whenever $\|h(x_c)\| > \rho$ at the beginning of an iteration, we need to reduce the infeasibility. This is done applying Powell's *dogleg* method [22] to the box constrained linear least squares problem (2.2), replacing x by x_c.

To find an approximate solution for this trust region problem, the dogleg method uses a path consisting of two line segments. The first connects the origin to the Cauchy point, defined as

$$s_{CS} = -\gamma A^T(x_c) h(x_c),$$

where

$$\gamma = \min\left\{ \frac{\Delta_{VS}}{\|A^T(x_c)h(x_c)\|}, \frac{\|A^T(x_c)h(x_c)\|^2}{\|A(x_c)A^T(x_c)h(x_c)\|^2} \right\}.$$

The second line runs from the Cauchy point to the Newton point

$$s_{NS} = -A^T(x_c)(A(x_c)A^T(x_c))^{-1}h(x_c). \tag{4.1}$$

If $\|s_{NS}\| \leq \Delta_{VS}$, then the Newton point is the solution of the problem. Otherwise, the point of intersection of the dogleg path and the trust region boundary is chosen.

The trust region radius Δ_{VS} used to compute the vertical step is updated using rules similar to those defined for the horizontal step.

Let P_{red} denote the predicted reduction and A_{red} the actual reduction of the infeasibility. The step is rejected if $A_{red}/P_{red} < 10^{-3}$. In this case, Δ_{VS} is divided by four. On the other hand, if $A_{red}/P_{red} \geq 0.5$, we double Δ_{VS}.

Sometimes, it is necessary to apply the dogleg method several times in order to obtain the desired level of infeasibility. To avoid recomputing A frequently, we try to take a new step using the same matrix whenever the dogleg method is able to reduce $\|h(x_c)\|$ by at least 10%. This expedient is used up to four times in a row, after what A is recalculated.

Another way to avoid frequent recalculations of A is to choose the trust cylinder radius ρ carefully. Unfortunately, ρ depends on $n_p(x_c)$, and this term, on its turn, depends on matrix $A(x_c)$. Naturally, it would not be wise to compute A just before calling the restoration, as we will need to update this matrix after this step. For this reason, in step 1.2 of Algorithm 2.1, we define an approximate value for ρ, replacing n_p by

$$n_p^a = \frac{|\Delta L_H|}{|f(x^{(k-1)}) - f(x_c^{(k-1)})| + \|\delta_t^{(k-1)}\|}.$$

After the restoration, $A(x_c)$ is available and we need to choose ρ satisfying the conditions stated at step 1.3.3 of Algorithm 2.1. These conditions are quite loose, so a good scheme for defining the trust cylinder radius can be devised, taking into account some problem characteristics and the values of ρ_{max} and n_p. In our implementation, however, a naive rule was used. If the approximate ρ computed at step 1.2 satisfies $10^{-4} n_p \rho_{max} \leq \rho\, n_p \leq \rho_{max}$, we keep this value. Otherwise, we simply define

$$\rho = \min\{\rho_{max} n_g,\, \max\{0.75\rho_{max},\, 10^{-4}\rho_{max} n_g\}\}. \tag{4.2}$$

The reduction obtained by the dogleg method may be small depending on the curvature of h. When this happens, we abandon the box constrained linear least squares problem and try to apply the Moré and Thuente line search algorithm [20] to the unconstrained nonlinear least squares problem

$$\textit{minimize} \quad \|h(x)\|^2, \tag{4.3}$$

using a BFGS approximation for the Hessian of the objective function.

Since this last approach is more time consuming than the dogleg method, it is applied only if $\|h(x_c)\|/\|h(x_{k-1})\| < 0.95$ for 3 successive dogleg steps. Fortunately, this is unlikely to occur, as the dogleg method usually works well.

4.1.2. Horizontal step. The horizontal step of the method consists in solving the quadratic programming problem (2.3). If Z is a matrix that spans the null space of $A(x_c)$, then it is possible to rewrite (2.3) as the box constrained nonlinear programming problem

$$\begin{aligned} &\textit{minimize} \quad g(x_c)^T Zv + \frac{1}{2} v^T Z^T B Z v \\ &\textit{subject to} \quad \|Zv\| \leq \Delta, \end{aligned} \tag{4.4}$$

where δ was replaced by Zv.

One should notice that B need not to be positive definite, so we cannot use the dogleg method to solve (4.4), as we did in the vertical step. Instead of that, we use the Steihaug-Toint method [29, 30], that is an extension of the conjugate gradient (CG) method for nonconvex problems.

Since computing the product of Z times a vector several times would be too costly, we write the Steihaug-Toint algorithm using δ directly, as described by Lalee, Nocedal and Plantenga in [11].

The method starts by computing the Cauchy step defined in step 4.1.1 of Algorithm 2.1. If this point falls inside the trust region, it is improved by applying successive CG iterations until $q(\delta) \leq 0.01q(\delta_{CP})$, or a direction of negative curvature is found, or the trust region boundary is violated. In the last two cases, a point over the boundary of the trust region is chosen.

4.1.3. Linear systems. In the core of both the vertical and the horizontal step, we have linear systems involving AA^T. Such systems need to be solved when we compute

- the Newton step (4.1), in the dogleg method;
- the Lagrange multipliers (2.4) and, consequently, the projected gradient (2.5);
- the second order correction (see (4.5) in the next subsection);
- the projection of the residual vector onto $N(A)$, in the Steihaug-Toint method.

Two routines are provided for solving these systems. One is based on the sparse Cholesky decomposition of AA^T. The second uses the conjugate gradient method to generate an approximate solution.

If we choose to work with the Cholesky decomposition, the approximate minimum degree algorithm of Amestoy, Davis and Duff [3] is used to reorder the rows and columns of AA^T, so the fill-in created during the factorization is minimized. For the CG method, a band preconditioner has been implemented to accelerate the method.

4.1.4. Second order correction. In DCI, a second order correction (SOC) can be used to reduce the infeasibility after the horizontal step, as the acceptance of this compound step is more probable to happen. Clearly, $\delta_{soc} = 0$ would be a possibility for the SOC term. In fact, any $\delta_{soc} = O(\|\delta_t\|^2)$ is acceptable for global convergence purposes. The non-zero natural candidate corresponds to

$$\begin{aligned} \delta_{soc} &= argmin\{\|A_k\delta + h(x_c + \delta_t)\|\} \\ &= -A_k^T(A_kA_k^T)^{-1}h(x_c + \delta_t). \end{aligned} \tag{4.5}$$

If we find g_p with a Cholesky factorization of AA^T, δ_{soc} results computationally cheap. On the other hand, if we use iterative methods to compute $g_p = g + A^T\lambda_{LS} = argmin\{\|A^T\lambda + g\|\}$, it looks reasonable to relax the convergence to g_p so we can save some time for computing the second order correction.

The second order correction is called if, after computing the horizontal step, we have

$$\|h(x_c + \delta_t)\| > \min\{2\rho,\ 2\|h(x_c)\| + 0.5\rho\}$$

or

$$\|h(x_c)\| \leq 10^{-5} \ \text{ and } \ \|h(x_c + \delta_t)\| > \max\{10^{-5},\ 2\|h(x_c)\|\},$$

If the second order correction is refused, it is not calculated again at the same global iteration of Algorithm 2.1.

4.2. Algorithm performance. To analyze the behavior of the algorithm just described, we used a set of 53 medium-size equality constrained problems extracted from the CUTEr collection [10]. The selected problems are presented in Table 4.1. The number of variables of the problem is given by n, while m is the number of constraints.

TABLE 4.1
Selected medium-size problems from the CUTEr collection.

Problem	n	m	Problem	n	m
AUG2D	20200	10000	HAGER1	10001	5000
AUG2DC	20200	10000	HAGER2	10001	5000
AUG3D	27543	8000	HAGER3	10001	5000
AUG3DC	27543	8000	LCH	3000	1
CATENA	3003	1000	LUKVLE1	10000	9998
CATENARY	501	166	LUKVLE10	10000	9998
CHAIN	802	401	LUKVLE11	9998	6664
DTOC1L	14995	9990	LUKVLE13	9998	6664
DTOC1NA	7495	4990	LUKVLE14	998	664
DTOC1NB	7495	4990	LUKVLE15	997	747
DTOC1NC	7495	4990	LUKVLE16	9997	7497
DTOC1ND	7495	4990	LUKVLE3	10000	2
DTOC2	5998	3996	LUKVLE4	10000	4999
DTOC3	14999	9998	LUKVLE5	10002	9996
DTOC4	14999	9998	LUKVLE6	9999	4999
DTOC5	9999	4999	LUKVLE7	10000	4
DTOC6	10001	5000	LUKVLE8	10000	9998
EIGENA2	2550	1275	LUKVLE9	10000	6
EIGENACO	1640	820	OPTCTRL3	4502	3000
EIGENB2	2550	1275	ORTHRDM2	4003	2000
EIGENBCO	1640	820	ORTHRDS2	1003	500
EIGENC2	2652	1326	ORTHREGA	2053	1024
EIGENCCO	1722	861	ORTHREGC	1005	500
ELEC	600	200	ORTHREGD	1003	500
GRIDNETB	3444	1764	ORTHRGDM	2003	1000
GRIDNETE	7564	3844	ORTHRGDS	1003	500
GRIDNETH	7564	2844			

Originally, all of the equality constrained problems of the CUTEr library were selected to compose the test set. However, at this moment, the DCI algorithm is not prepared to handle singular Jacobian matrices, so some of the problems needed to be excluded from the list.

The DCI algorithm was implemented in FORTRAN 77 and the executable program was generated using the ifort 9.0 compiler, under the Fedora 4 Linux operating system. To evaluate the performance of the new method, it was compared to Lancelot (release B), the well known nonlinear programming package distributed along with the Galahad library [9]. Although Lancelot is outperformed by most nonlinear programming codes available nowadays, we decided to use it because it is freely available and includes a good interface for solving CUTEr problems.

The tests were performed on a Dell Optiplex GX280 computer, using an Intel Pentium 4 540 processor, with a clock speed of 3.2GHz, 1MB of cache memory, a 800MHz front side bus and the Intel 915G chipset. The Lancelot default parameters were adopted, except for the maximum number of iterations that was increased to 10000. Exact first and second derivatives were computed by both methods.

The DCI algorithm was designed to declare convergence when both $\|h(x)\| < \epsilon_h$ and $n_p < \epsilon_g$, as well as when $\rho_{max} < \epsilon_r$. However, since Lancelot uses the infinity

norm in its convergence criteria, we decided to change the first criterion, stopping the algorithm when $\|h(x)\|_\infty < \epsilon_h$ and one of $\|g_p\|_\infty < \epsilon_g$ or $n_p < \epsilon_p$ occurs. Besides, it also terminates if $\|\delta_t\| < \epsilon_d\|x\|$ for 10 successive iterations or if the restoration fails to obtain a feasible point. The constants $\epsilon_h = 10^{-5}$, $\epsilon_g = 10^{-5}$, $\epsilon_p = 10^{-7}$, $\epsilon_r = 10^{-7}$ and $\epsilon_d = 10^{-8}$ were adopted, so the stopping tolerances are compatible with those used in Lancelot.

Other parameters used in the algorithm are

$$\rho^0_{max} = \max\{10^{-5},\, 5.1\|h(x^{(0)})\|,\, 50\, n_p(x^{(0)})\}, \tag{4.6}$$

$$\Delta^0 = \Delta^0_{VS} = \max\{10\|x^{(0)}\|,\, 10^5\},$$

and $\Delta_{min} = 10^{-5}$.

The comparison of the methods were done using the performance profiles defined by Dolan and Moré [5]. To draw the performance profiles for a set S of solvers on a set P of problems, we need to compute, for each problem $p \in P$ and each solver $s \in S$, the performance ratio defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

where $t_{p,s}$ is the time spent by the solver s to solve problem p. The overall performance of solver s is represented by function

$$P(t) = \frac{1}{n_p} size\{p \in P : r_{p,s} \leq t\},$$

where n_p is the number of problems considered. In words, $P(t)$ is the fraction of the number of problems that are solved by s within a factor t of the time spent by the fastest solver (for each problem). Plotting $P(t)$, we get a performance profile for a particular solver.

For the 53 equality constrained problems selected, the performance profiles of DCI and Lancelot are shown in Figure 4.2

One can deduce from Figure 4.2 that the DCI algorithm took less time than Lancelot to obtain the solution of almost three quarters of the problems. Besides, DCI solved 90% of the problems within a factor 8.1 of the best solver, while it was necessary to increase the factor to more than 40 in order for Lancelot to solve the same 90% of the problems.

Both method obtained an optimal solution (i.e. an stationary point for (1.1)) for all of the problems. However, the performance of DCI was never worse than a factor 50 of the performance of Lancelot, while Lancelot took more than 200 times the time spent by DCI to solve the CHAIN problem.

The experiments with these CUTEr problems revealed also that the choice of an initial value for ρ_{max} is still an open problem. For several problems, a particular value of ρ^0_{max} has led to a much better performance of the algorithm, if compared to (4.6). One possible way to circumvent this problem is to use a few iterations of the algorithm only to calibrate this parameter, prior to use the rules for updating it. This modification will be investigated in the near future.

The choice of the linear systems solver also affects the performance of the algorithm. For all of the problems presented here, we used the Cholesky decomposition to compute the solution of $(AA^T)s = b$, although, for many of them, it would be preferable to use the preconditioned conjugate gradient method.

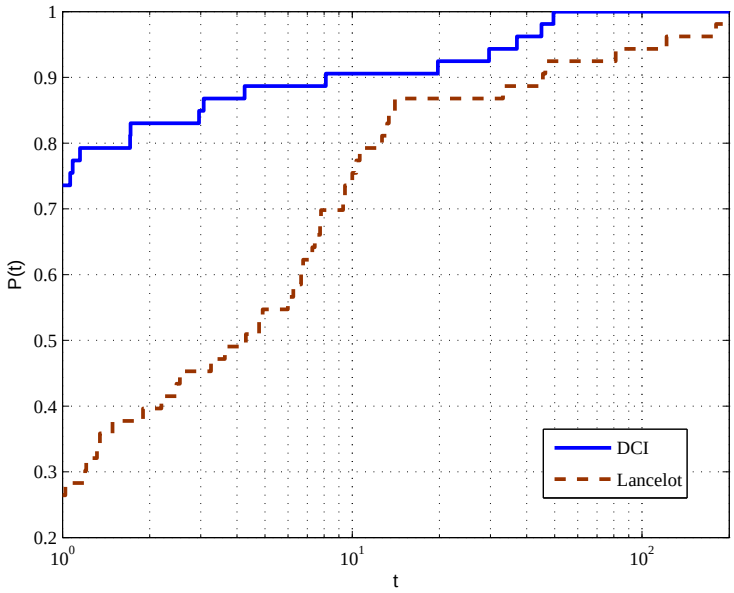

FIG. 4.1. *Performance profiles for 53 CUTEr problems.*

5. Conclusions. In this paper, we have presented a new algorithm for solving nonlinear programming problems with equality constraints. The method uses the idea of a trust cylinder to keep the infeasibility under control. The radius of this cylinder is reduced as the algorithm approaches the optimal point. The algorithm is globally convergent in the sense that its accumulation set has stationary points for (1.1).

Our current implementation of the algorithm works well when applied to medium-sized problems, so we believe that it is worth investigating its performance for larger problems. Some of the improvements that are to be made to the code after solving large-scale problems include:

- reformulating the algorithm so inexact solutions for the linear subroutines are admitted;
- using BFGS approximations to the Hessian of the Lagrangian when computing the horizontal step;
- devising a rule to define the initial value of ρ_{max};
- allowing the code to deal with rank deficient Jacobians.

Besides, we also have plans to extend the algorithm to solve inequality constrained problems.

REFERENCES

[1] J. ABADIE AND J. CARPENTIER, *Some numerical experiments with the GRG method for nonlinear programming*, Paper HR7422, Électricité de France, Paris, France, 1967.

[2] J. ABADIE AND J. CARPENTIER, *Generalization of the Wolfe Reduced Gradient Method to the case of nonlinear constraints*, in Optimization, R. Fletcher, ed., Academic Press, London, England, 1969, pp. 37–47.

[3] P. AMESTOY, T. A. DAVIS AND I. S. DUFF, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. and Applics., 17 (1996), pp. 886–905.

[4] L. T. Biegler, J. Nocedal, C. Schmid and D. Ternet, *Numerical experience with a reduced Hessian method for large scale constrained optimization*, Comput. Optim. and Applics., 15 (2000), pp. 45–67.

[5] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Programming, 91 (2002), pp. 201–213.

[6] A. Drud, *CONOPT - A GRG code for large sparse dynamic nonlinear optimization problems* Math. Programming, 31 (1985), pp. 153–191.

[7] Y. Fan, S. Sarkar and L. Lasdon, *Experiments with successive quadratic programming algorithms*, J. Optim. Theory and Applics., 56 (1988), pp. 359–383.

[8] N. I. M. Gould, M. E. Hribar and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Computing, 23 (2001), pp. 1375–1394.

[9] N. I. M. Gould, D. Orban and P. L. Toint, *GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, 29 (2003), pp. 353-372.

[10] N. I. M. Gould, D. Orban and P. L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[11] M. Lalee; J. Nocedal and T. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*, SIAM J. Optim., 8 (1998), pp. 682–706.

[12] L. Lasdon, *Reduced gradient methods*, in Nonlinear Optimization 1981, M. J. D. Powell, ed., Academic Press, New York, NY, 1982, pp. 235–242.

[13] J. M. Martínez, *A trust-region SLCP model algorithm for non-linear programming*, in Foundations of Computational Mathematics, F. Cucker and M. Shub, eds., Springer, New York, NY, 1997, pp. 246–255.

[14] J. M. Martínez, *Two-phase model algorithm with global convergence for nonlinear programming*, J. Optim. Theory and Applics., 96 (1998), pp. 397–436.

[15] J. M. Martínez, *Inexact restoration method with Lagrangian tangent decrease and new merit function for nonlinear programming*, J. Optim. Theory and Applics., 111 (2001), pp. 39–58.

[16] J. M. Martínez and E. A. Pilotta, *Inexact restoration algorithm for constrained optimization*, J. Optim. Theory and Applics., 104 (2000), pp. 135–163.

[17] J. M. Martínez and E. A. Pilotta, *Inexact restoration methods for nonlinear programming: advances and perspectives*, To appear in Optimization and Control with applications, L. Qi, K. Teo and X. Yang, eds., Springer, New York, NY, 2005.

[18] A. Miele; H. Y. Huang; J. C. Heideman, *Sequential gradient-restoration algorithm for the minimization of constrained functions - ordinary and gradient versions*, J. Optim. Theory and Applics., 4 (1969), pp. 213–246.

[19] A. Miele; A. V. Levy; E. E. Cragg, *Modifications and extensions of the conjugate gradient-restoration algorithm for mathematical programming problems*, J. Optim. Theory and Applics., 7 (1971), pp. 450–472.

[20] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, 20 (1994), pp. 286–307.

[21] H. Mukai and E. Polak, *On the use of approximations in algorithms for optimization problems with equality and inequality constraints*, SIAM J. Num. Anal., 15 (1978), pp. 674–693.

[22] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical methods for nonlinear algebraic equations, P. Rabinowitz, ed., Gordon and Breach, London, UK, 1970, pp. 87–114.

[23] M. Rom and M. Avriel, *Properties of the Sequential Gradient-Restoration Algorithm (SGRA), Part1: Introduction and comparison with related methods*, J. Optim. Theory an Applics., 62 (1989), pp. 77–98.

[24] M. Rom and M. Avriel, *Properties of the Sequential Gradient-Restoration Algorithm (SGRA), Part2: Convergence Analysis*, J. Optim. Theory an Applics., 62 (1989), pp. 99–125.

[25] J. B. Rosen, *The gradient projection method for nonlinear programming, part I - Linear constraints*, SIAM J. Appl. Math., 8 (1960), pp. 181–217.

[26] J. B. Rosen, *The gradient projection method for nonlinear programming, part II - Nonlinear constraints*, SIAM J. Appl. Math., 9 (1961), pp. 514–532.

[27] J. B. Rosen, *Two-phase algorithm for nonlinear constraint problems*, in Nonlinear Programming 3, O. L. Mangasarian, R. R. Meyer and S. M. Robinson, eds., Academic Press, New York, NY, 1978, pp. 97–124.

[28] S. Smale, *On gradient dynamical systems*, Ann. of Math., 74 (1961), pp. 199–206.

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

[30] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, Lundon, UK, 1981, pp. 57-88.

[31] R. A. Waltz & J. Nocedal, *KNITRO user's manual*, Tech. Report OTC 2003/05, Optimization Technology Center, Northwestern University, Evanston, IL, 2003.

[32] P. Wolfe, *Methods of nonlinear programming*, in Recent Advances in mathematical Programming, R. L. Graves and P. Wolfe, eds., McGraw Hill, New York, NY, 1963, pp. 67–86.