DYNAMIC CONTROL OF INFEASIBILITY IN EQUALITY 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 and local convergence of the algorithm are 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.

We consider the equality constrained optimization problem

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

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

Usual algorithms for solving problem (1.1) alternate normal (or "vertical") steps towards the feasible set $\mathcal{H}_0 = \{x : h(x) = 0\}$ with tangential (or "horizontal") steps towards the dual manifold $\nabla\mathcal{L} = \{x : \nabla L(x, \lambda) = 0\}$, where L is the Lagrangian function. Generally, these steps are obtained from some quadratic model for (1.1). This feature is shared, for example, by the trust region methods proposed by Biegler, Nocedal and Schmid [4], Byrd, Gilbert and Nocedal [9], Byrd, Hribar and Nocedal [10], Dennis and Vicente [16], El-Alem [18], Gomes, Maciel and Martínez [23], and Lalee, Nocedal and Plantenga [24].

In this paper, we propose an algorithm that uses normal and tangential trust region models in a more flexible way. Our bet is that, rather than taking one normal and one tangential step per iteration, we might do better if, at some iterations, $\nabla\mathcal{L}$ is pursued with some priority, so several successive horizontal steps are taken before one vertical step is computed. On the other hand, we believe that, in some cases, it is preferable to move closer and closer to $\mathcal{H}_0$, so we systematically force the vertical step. To allow this to occur, we introduce a single mechanism based on what we call *trust cylinders*.

Another feature that distinguishes our method from most of the nonlinear optimization algorithms recently proposed is that, using trust cylinders, we need not to rely on a filter (see, for example, [14, chap. 15]) or on a merit function (see [23]) to obtain global convergence. Instead, we accept the horizontal step if it sufficiently decreases the Lagrangian function, subject to the condition of staying (dynamically) close to feasibility, in a sense that will be explained in the sequel.

Algorithms that generate feasible iterates, without solving $h(x) = 0$ explicitly, go back to the early sixties, with methods usually classified either as Generalized Reduced

*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`).

Gradient (GRG) (see [45, 1, 2]), or as Projected Gradient (PG) [38, 39]. 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) [30, 31]. See also [33, 36, 37]. More recently, Martínez introduced a new class of algorithms called *inexact restoration methods* [25, 26, 27, 28, 29], that also controls infeasibility at each iteration.

Our approach has the flavor 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 large enough horizontal step, in a direction approximately tangent to the restrictions $h(x) = 0$, and keeping infeasibility under control. The main idea is to force each iterate x^k to remain in a trust cylinder defined by

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

where $\|.\|$ denotes the ℓ_2 norm.

The dynamic control of infeasibility is kept defining the "radii" ρ^k of the trust cylinders in such a way that

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

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 $\nabla h(x)$, the Jacobian of h. In our case of interest, $g_p(x)$ will be calculated at regular points of h, where $\nabla h(x)$ has full rank. In this situation, the least squares multiplier estimates, $\lambda_{LS}(x)$, are given by

$$\lambda_{LS}(x) = argmin\{\|\nabla h(x)^T \lambda + g(x)\|\} = -(\nabla h(x) \nabla h(x)^T)^{-1} \nabla h(x) g(x) \tag{1.3}$$

and the resulting projected gradient is

$$g_p(x) = g(x) + \nabla h(x)^T \lambda_{LS}(x). \tag{1.4}$$

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 $\rho = \rho^k$, such that

$$\|h(x_c)\| \leq \rho = O(\|g_p(x_c)\|) \tag{1.5}$$

and

$$\|x_c - x^{k-1}\| = O(\|h(x^{k-1})\|). \tag{1.6}$$

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

$$n_p(x_c) = \frac{\|g_p(x_c)\|}{\|g(x_c)\| + 1} \tag{1.7}$$

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 1.1 sketches the vertical and the horizontal steps of a typical iteration.

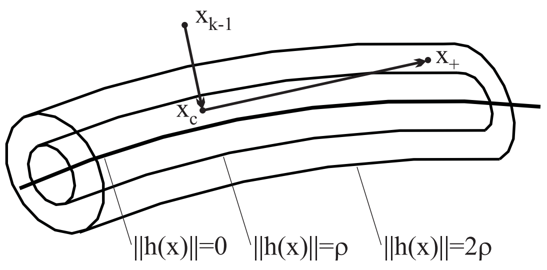

FIG. 1.1. *The step and the trust cylinders. x_c satisfies $\|h(x_c)\| < \rho$, while x_+ satisfies $\|h(x_+)\| < 2\rho$.*

An advantage of staying close to the feasible set is that 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 $\rho_{max} = \rho^k_{max}$ 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 function obtained in the horizontal step 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, followed by the local convergence theory, introduced in Section 4. Section 5 contains some preliminary numerical results. Finally, some conclusions and lines for future work are included in Section 6.

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 also based on the the variation of the Lagrangian at x^k_c, given by

$$\Delta L^k_c = L^k_c - L^{k-1}_c.$$

where $L^k_c = L(x^k_c, \lambda^k)$. Since our algorithm divides the step into two components, one vertical and one horizontal, this variation is also split according to

$$\Delta L^k_c = \Delta L^{k-1}_H + \Delta L^k_V, \tag{2.1}$$

where

$$\Delta L^{k-1}_H = L(x^{k-1}, \lambda^{k-1}) - L(x^{k-1}_c, \lambda^{k-1}), \tag{2.2}$$

$$\Delta L^k_V = L(x^k_c, \lambda^k) - L(x^{k-1}, \lambda^{k-1}). \tag{2.3}$$

In the vertical step of the algorithm, we seek a point x_c that satisfies (1.5) and (1.6). Under usual regularity assumptions for $\nabla h(x)$ along the iterations, we can achieve (1.5) and (1.6) with almost every algorithm available for the least squares problem

$$minimize \;\; \|h(x)\|^2. \tag{2.4}$$

Careful line searches in the Cauchy, Gauss-Newton or Newton directions, or in some combination of them, can be used, for example, to solve problem (2.4). As a matter of fact, algorithms for (2.4) that take steps in the form $d = -M\nabla h(x)^T h(x)$, where M represents a family of uniformly bounded and positive definite matrices, will produce a "sufficiently" fast convergence to a feasible point, and that is what we need to guarantee (1.6). In the implementation of the algorithm, we will give preference to a trust region method. Namely, our restoration step successively solves the linearized least squares problem

$$\begin{aligned} &\textit{minimize} \;\; \|h(x) + Ad\|^2 \\ &\textit{subject to} \;\; \|d\|_\infty \le \Delta_{VS}, \end{aligned} \tag{2.5}$$

where A is an approximation of $\nabla h(x)$ and $\Delta_{VS} > 0$ is a trust region radius adequately updated in the vertical subproblems.

In the horizontal step we solve the quadratic programming problem

$$\begin{aligned} &\textit{minimize} \;\; q(\delta) = g(x_c)^T\delta + \frac{1}{2}\delta^T B\delta \\ &\textit{subject to} \;\; \nabla h(x_c)\delta = 0, \\ &\qquad\qquad\;\; \|\delta\|_\infty \le \Delta, \end{aligned} \tag{2.6}$$

where B is a symmetric approximation of 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, λ^{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, ΔL_H^{k-1}, and the trust region radii, $\Delta_{VS} \ge \Delta_{min}$ and $\Delta \ge \Delta_{min}$.

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 $\rho$.
1.3.        REPEAT
1.3.1.        IF $\|h(x_c)\| > \rho$
1.3.1.1.        Find $x_c$, such that $\|h(x_c)\| \le \rho$.
1.3.2.        $g_p \leftarrow g_p(x_c)$;  $n_p \leftarrow \|g_p(x_c)\|/(\|g(x_c)\| + 1)$.
1.3.3.        Choose $\rho \in [10^{-4}n_p\rho_{max}, n_p\rho_{max}]$.
1.4.        UNTIL $\|h(x_c)\| \le \rho$.
1.5.        Compute $\lambda_+$.
2.        Convergence test:
2.1.        IF $(n_p = 0)$,
2.1.1.        QUIT ($x_c$ is a stationary point).
3.        $\rho_{max}$ update:
3.1.        $\Delta L_V^k \leftarrow L(x_c, \lambda_+) - L(x^{k-1}, \lambda^{k-1})$.
3.2.        IF $\Delta L_V^k \ge \frac{1}{2}[L_{ref} - L(x^{k-1}, \lambda^{k-1})]$,
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 $\delta_{CP}$, solution of
                $minimize \quad q(\mu g_p)$
                $subject\ to \quad \|\mu g_p\| \leq \Delta, \ \mu \in [0, \infty).$
4.1.2.        Compute a trial step $\delta_t$ such that
                $q(\delta_t) \leq q(\delta_{CP})$,
                $\|\delta_t\| \leq \Delta$, and
                $\nabla h(x_c)\, \delta_t = 0$.
4.1.3.        Optionally, compute a second order correction $\delta_{soc}$.
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 $r > \eta_2$,
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 1.3.1.1 will always succeed. Obviously, this may not occur, since problem (1.1) may be infeasible. Therefore, some termination criterion needs to be defined to prevent the algorithm from getting stuck on this step.

In step 1.3.2 of the above algorithm and in the next section, we assume that n_p is computed according to (1.7). As a matter of fact, all we need for our convergence theory is that $n_p = O(\|g_p(x_c)\|)$, but we decided to use an explicit formula for n_p in order to keep the text more readable. Another choice we made was to define λ_+ as the vector of least squares multipliers (1.3) computed at x_c^k, although other update schemes would work as well.

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 Δ_{min}[*] and the four constants that control the behavior of the trust region method used to compute the horizontal step were not specified. These parameters must satisfy $0 < \eta_1 \leq 1/2$, $\eta_1 \leq \eta_2 < 1$, $0 < \alpha_R < 1$, $\alpha_I \geq 1$ and $\Delta_{min} > 0$. Possible values are $\eta_1 = 10^{-3}$, $\eta_2 = 0.7$, $\alpha_R = 0.25$, $\alpha_I = 2.5$ and $\Delta_{min} = 10^{-5}$.

The following relations, easily derived from steps 1.3 and 4.2 of Algorithm 2.1, will

[*]The parameter Δ_{min} plays no role in the global convergence theory of Algorithm 2.1. It is also unnecessary for the local convergence theory if we use the true Lagrangean second order polynomial as the quadratic model $q(\delta)$ or if B is a good approximation of $\nabla_{xx}^2 L(x, \lambda)$ in the plane that contains the Cauchy and the quasi-Newton directions, as it will become clear in section 4. However, our numerical tests indicate that keeping $\Delta \geq 10^{-5}$ may slightly improve the performance of the algorithm, so we decided to include step 5.2.

be used frequently in the next two sections and are presented here for convenience.

$$\rho^k \leq \rho^k_{max} \|g_p(x^k_c)\| \tag{2.7}$$

$$\rho^k_{max} \leq 10^4 \rho^k \frac{1}{n_p} = 10^4 \rho^k \frac{\|g(x^k_c)\| + 1}{\|g_p(x^k_c)\|} \tag{2.8}$$

$$\|h(x^k_c)\| \leq \|h(x^{k-1})\| \leq 2\rho^{k-1} \tag{2.9}$$

The global convergence of DCI will be guaranteed, under reasonable assumptions, by a typical *sufficient decrease* argument for the Lagrangian function evaluated at x^k_c. The variation of the Lagrangian between two successive iterations is given by (2.1). The idea is to prevent the decrease of the Lagrangian obtained at the horizontal step to be destroyed by the restoration. For that, ρ^k_{max} is decreased in step 3 of DCI every time ΔL^k_V 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 ΔL^k_V 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 $\Delta L^k_V > -\frac{1}{2}\Delta L^{k-1}_H$. 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 ρ^k_{max} remains bounded away from zero, unless $\liminf(\|g_p(x^k_c)\|) = 0$.

3. Global convergence.

The global convergence analysis of the DCI algorithm is based on the following hypotheses:

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, $\{\|\nabla h({x_c}^k)^T \nabla h({x_c}^k)^{-1}\|\}$ 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 hypotheses will hold if, for example, the feasible set $\mathcal{H}_0$ is compact, regular (i.e. $\nabla h(x)$ is of maximal rank on $\mathcal{H}_0$) and x^0 is feasible. In this case, if we choose an initial ρ^0_{max} sufficiently small, we can keep $\nabla h(x)$ with maximal rank and assure (3.1) using standard algorithms for restoration, like the Gauss-Newton method. We could also replace the compacity property of $\mathcal{H}_0$ by adequate properties on f, such as requiring 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 $\{x^k_c\}$ 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, and is associated to these specific sequences generated by DCI.

Denoting by δ^k_V the vertical step and by δ^k_H the horizontal step in the k^{th} iteration, we have

$$\delta^k_V = {x_c}^k - x^{k-1} \quad \text{and} \quad \delta^k_H = x^k - {x_c}^k = {\delta_t}^k + \delta^k_{soc}.$$

Hypotheses H1-H4 allow us to choose a constant $\delta_{max} > 0$, such that, for all k,

$$\|\delta^k_t\| + \|\delta^k_{soc}\| + \|\delta^k_V\| \leq \delta_{max}. \tag{3.2}$$

We can also define a second positive constant ξ_0, such that, for all k, if $\|x - x_c^k\| \le \delta_{max}$, then

$$\|\nabla h_j(x)\| \le \xi_0, \quad j = 1, \dots, m, \tag{3.3}$$

$$\|\nabla^2 h_j(x)\| \le \xi_0, \quad j = 1, \dots, m, \tag{3.4}$$

$$\|\nabla f(x)\| \le \xi_0, \tag{3.5}$$

$$\|\nabla^2 f(x)\| \le \xi_0, \tag{3.6}$$

$$\|\nabla h(x_c^k)^T \nabla h(x_c^k)\| \le \xi_0, \tag{3.7}$$

$$\|B^k\| \le \xi_0, \tag{3.8}$$

$$\|\lambda^k\| \le \xi_0, \tag{3.9}$$

$$\|\delta_{soc}^k\| \le \xi_0 \|\delta_t^k\|^2. \tag{3.10}$$

To simplify our notation, we suppose that the constant ξ_0 is large enough so equation (3.1) can be rewritten as

$$\|{x_c}^{k+1} - x^k\| \le \xi_0 \|h(x^k)\|. \tag{3.11}$$

The main result of this section, presented in Theorem 3.6, is based on five lemmas. The first one gives an upper limit for the increase in the infeasibility produced by the horizontal step.

Lemma 3.1. *The trial iterate x_+ generated at step 4.1.4 of Algorithm 2.1 satisfies*

$$\|h(x_+) - h({x_c}^k)\| \le \overline{\xi_0} \|\delta_t\|^2. \tag{3.12}$$

Proof. Since $x_+ = {x_c}^k + \delta_+ = {x_c}^k + \delta_t + \delta_{soc}$, with $\|\delta_+\| \le \delta_{max}$, we can use a Taylor expansion together with relations (3.4), (3.3), (3.10) and (3.2), and the fact that $\nabla h({x_c}^k){\delta_t}^k = 0$, to show that, for every $j = 1, \dots, m$,

$$\begin{aligned} |h_j(x_+) - h_j(x_c)| &\le |\nabla h_j(x_c)^T(\delta_t + \delta_{soc})| + \frac{\xi_0}{2}\|\delta_t + \delta_{soc}\|^2 \\ &= |\nabla h_j(x_c)^T \delta_{soc}| + \frac{\xi_0}{2}\|\delta_t + \delta_{soc}\|^2 \\ &\le \xi_0^2 \|\delta_t\|^2 + \xi_0(\|\delta_t\|^2 + \|\delta_{soc}\|^2) \\ &\le (\xi_0^2 + \xi_0 + \xi_0^2 \delta_{max})\|\delta_t\|^2. \end{aligned}$$

Setting $\overline{\xi}_0 = \sqrt{m}(\xi_0^2 + \xi_0 + \xi_0^2\delta_{max})$, we get the desired result. □

The second lemma establishes that, under H1-H4, each iteration succeeds and the Lagrangian is sufficiently decreased.

Lemma 3.2. *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, for all k,*

$$\begin{aligned} \Delta L_H^k &= L(x^k, \lambda^k) - L(x_c^k, \lambda^k) \\ &\le -\xi_1 \|g_p(x_c^k)\| \min\left\{\xi_2 \|g_p(x_c^k)\|,\ \xi_3\sqrt{\rho^k}\right\}. \end{aligned} \tag{3.13}$$

Proof. To simplify the notation we will omit here the superscript k. Suppose that 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.

Using a Taylor expansion, Lemma 3.1 and relations (3.6), (3.9), (3.5), (3.2), (3.10) and (3.8), we obtain

$$\begin{aligned}\Delta L_H^+ &= L(x_+,\lambda_+) - L(x_c,\lambda_+) = f(x_+) - f(x_c) + \lambda_+^T(h(x_+) - h(x_c))\\ &\leq g(x_c)^T\delta_t + g(x_c)^T\delta_{soc} + \frac{\xi_0}{2}\|\delta_t+\delta_{soc}\|^2 + \xi_0\overline{\xi}_0\|\delta_t\|^2\\ &\leq q(\delta_t) - \frac{1}{2}\delta_t^T B\delta_t + \overline{\xi}_1\|\delta_t\|^2\\ &\leq q(\delta_t) + \overline{\xi}_2\|\delta_t\|^2, \qquad (3.14)\end{aligned}$$

where $\overline{\xi}_1 = \xi_0^2 + \xi_0 + \xi_0^2\delta_{max} + \xi_0\overline{\xi}_0$ and $\overline{\xi}_2 = \xi_0/2 + \overline{\xi}_1$.

Because δ_{CP}, defined in step 4.1.1 of DCI, is a Cauchy step tangent to the constraints, we have (see, for example, [14])

$$\|\delta_{CP}\| \geq \min\left\{\frac{\|g_p(x_c)\|}{\|B\|},\Delta\right\} \geq \min\left\{\frac{\|g_p(x_c)\|}{\xi_0},\Delta\right\}$$

and

$$q(\delta_{CP}) \leq \frac{1}{2}g(x_c)^T\delta_{CP} \leq -\frac{1}{2}\|g_p(x_c)\|\min\left\{\frac{\|g_p(x_c)\|}{\xi_0},\Delta\right\}. \qquad (3.15)$$

To prove the first part of the lemma, we will show that x_+ is always accepted whenever $\Delta \leq \overline{\Delta}$, where

$$\overline{\Delta} = \min\left\{\frac{\|g_p(x_c)\|}{4\overline{\xi}_2},\sqrt{\frac{\rho}{\overline{\xi}_0}}\right\}. \qquad (3.16)$$

We start noting that, since $\xi_0 < 4\overline{\xi}_2$,

$$\overline{\Delta} \leq \frac{\|g_p(x_c)\|}{\xi_0}. \qquad (3.17)$$

Based on the fact that $\Delta \leq \overline{\Delta}$ and on (3.17), the upper limit of $q(\delta_{CP})$ given by (3.15) can be simplified to

$$q(\delta_{CP}) \leq -\frac{1}{2}\|g_p(x_c)\|\Delta. \qquad (3.18)$$

Combining the conditions $\|\delta_t\| \leq \Delta$ and $q(\delta_t) \leq q(\delta_{CP})$, stated in step 4.1.2 of the DCI algorithm, with (3.16) and (3.18), we obtain

$$\overline{\xi}_2\|\delta_t\|^2 \leq \overline{\xi}_2\Delta^2 \leq \overline{\xi}_2\overline{\Delta}\Delta \leq \frac{1}{4}\|g_p(x_c)\|\Delta \leq -\frac{1}{2}q(\delta_{CP}) \leq -\frac{1}{2}q(\delta_t). \qquad (3.19)$$

Now, from (3.14) and (3.19), we get

$$\Delta L_H^+ \leq \frac{1}{2}q(\delta_t) < 0, \qquad (3.20)$$

which implies that

$$r = \frac{\Delta L_H^+}{q(\delta_t)} \geq \frac{1}{2} \geq \eta_1. \qquad (3.21)$$

Since $\Delta \leq \overline{\Delta}$ and $\|h(x_c)\| \leq \rho$, we can use (3.12) and (3.16) to guarantee that

$$\|h(x_+)\| \leq \rho + \bar{\xi}_0 \|\delta_t\|^2 \leq \rho + \bar{\xi}_0 \overline{\Delta}^2 \leq 2\rho.$$

Therefore, both conditions stated at step 4.2 of the algorithm are satisfied and x_+ is accepted.

To prove the second part of the lemma, let us recall that, each time the step is rejected, Δ is multiplied by α_R, which means that we can assume the accepted trust region radius satisfies $\Delta \geq \alpha_R \overline{\Delta}$, where $0 < \alpha_R < 1$. Combining this with (3.20), the condition $q(\delta_t) \leq q(\delta_{CP})$, (3.15), (3.17) and (3.16), we get

$$\begin{aligned} \Delta L_H^+ \leq \frac{1}{2} q(\delta_t) \leq \frac{1}{2} q(\delta_{CP}) &\leq -\frac{1}{4} \|g_p(x_c)\| \min\left\{\frac{\|g_p(x_c)\|}{\xi_0}, \Delta\right\} \\ &\leq -\frac{1}{4} \|g_p(x_c)\| \alpha_R \overline{\Delta}. \end{aligned} \tag{3.22}$$

Defining $\xi_1 = \alpha_R/4$, $\xi_2 = 1/(4\bar{\xi}_2)$ and $\xi_3 = 1/\sqrt{\bar{\xi}_0}$, we obtain (3.13) from (3.16) and (3.22). □

Our third lemma defines an upper limit for the (possibly positive) vertical variation of the Lagrangian, ΔL_V^{k+1}.

LEMMA 3.3. *There exists a positive constant* ξ_4, *such that*

$$\Delta L_V^{k+1} \leq \xi_4 \rho_{max}^k \|g_p(x_c^k)\|. \tag{3.23}$$

Proof. Using a Taylor expansion, (3.5), (3.9), (3.11), (2.9) and (2.7), we get, for the vertical variation,

$$\begin{aligned} \Delta L_V^{k+1} &= L(x_c^{k+1}, \lambda^{k+1}) - L(x^k, \lambda^k) \\ &= f(x_c^{k+1}) - f(x^k) + {\lambda^{k+1}}^T h(x_c^{k+1}) - {\lambda^k}^T h(x^k) \\ &\leq \xi_0 \|x_c^{k+1} - x^k\| + \xi_0 \|h(x_c^{k+1})\| + \xi_0 \|h(x^k)\| \\ &\leq (\xi_0^2 + 2\xi_0) \|h(x^k)\| \leq 2(\xi_0^2 + 2\xi_0)\rho^k \leq \xi_4 \rho_{max}^k \|g_p(x_c^k)\|. \end{aligned}$$

Therefore, defining $\xi_4 = 2(\xi_0^2 + 2\xi_0)$, we obtain the desired result. □

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

LEMMA 3.4. *If* $\rho_{max}^{k+1} = \rho_{max}^{k+2} = \ldots = \rho_{max}^{k+j}$, *for* $j \geq 1$, *then*

$$L_c^{k+j} - L_c^k = \sum_{i=k+1}^{k+j} \Delta L_c^i \leq \frac{1}{4} \sum_{i=k}^{k+j-1} \Delta L_H^i + r^k, \tag{3.24}$$

where $r^k = \frac{1}{2}[L_{ref}^k - L_c^k]$.

Proof. Let us 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 (2.1) and the criterion defined in step 3.3 of the algorithm, we have

$$L_c^{k+j_1-1} - L_c^k = \sum_{i=k+1}^{k+j_1-1} (\Delta L_V^i + \Delta L_H^{i-1}) \leq \frac{1}{2} \sum_{i=k}^{k+j_1-2} \Delta L_H^i. \tag{3.25}$$

On the other hand, if L_{ref} changes at iteration $k+j_1$, then the condition stated in step 3.3 of DCI is satisfied. In this case, using the hypothesis that ρ_{max} stays unchanged at iteration $k+j_1$ (so the inequality at step 3.2 is not satisfied) and the fact that $\Delta L_H^k \leq 0$, for all k, we have

$$\begin{aligned} L_c^{k+j_1} - L_c^k &= \Delta L_V^{k+j_1} + L(x^{k+j_1-1}, \lambda^{k+j_1-1}) - L_{ref}^k + [L_{ref}^k - L_c^k] \\ &\leq \frac{1}{2}(L(x^{k+j_1-1}, \lambda^{k+j_1-1}) - L_{ref}^k) + [L_{ref}^k - L_c^k] \\ &= \frac{1}{2}(\Delta L_H^{k+j_1-1} + L_c^{k+j_1-1} - L_c^k) + \frac{1}{2}[L_{ref}^k - L_c^k] \\ &\leq \frac{1}{4}\sum_{i=k}^{k+j_1-1} \Delta L_H^i + r^k. \qquad (3.26) \end{aligned}$$

If $j_1 \geq j$, then (3.25) and (3.26) imply (3.24).

On the other hand, if L_{ref} is updated at iterations $k+j_1, \ldots, k+j_s$, where $0 < j_1 < j_2 < \ldots < j_s \leq j$, then $r^{k+j_1} = r^{k+j_2} = \ldots = r^{k+j_s} = 0$. Therefore, applying the same procedure described above several times and defining $j_0 = 0$, we obtain

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

□

Our fifth lemma establishes the existence of *enough normal space* in the trust cylinders $\mathcal{C}^k$ to guarantee that the Lagrangian can be sufficiently decreased. The idea supporting this lemma is that (3.13) guarantees, asymptotically, that $\|\Delta L_H^k\|$ is bigger than a fraction of $\sqrt{\rho^k}$, while, on the other hand, $\|\Delta L_V^k\| = O(\rho^k)$ (see the proof of Lemma 3.3). This means that a restoration can't, asymptotically, destroy the decrease in the Lagrangian achieved at the horizontal step, and this prevents further ρ_{max} updates.

LEMMA 3.5. *If DCI generates an infinite sequence* $\{x^k\}$, *then*

(i) There are positive constants ξ_5 *and* ξ_6 *such that, whenever*

$$\rho_{max}^k < \min\{\xi_5\|g_p(x_c^k)\|, \xi_6\}, \qquad (3.27)$$

ρ_{max} *doesn't change at iteration* $k+1$.

(ii) Furthermore, if $\liminf \|g_p(x_c^k)\| > 0$, *then there exists* $k_0 > 0$ *such that, for every* $k \geq k_0$,

$$\rho_{max}^k = \rho_{max}^{k_0}. \qquad (3.28)$$

(iii) If the horizontal step and the vector of Lagrange multipliers satisfy

$$\|x^k - x_c^k\| = O(\|g_p(x_c^k)\|) \qquad (3.29)$$

$$\|\lambda^k - \lambda_{LS}(x_c^k)\| = O(\|g_p(x_c^k)\|) \qquad (3.30)$$

then (3.28) is satisfied, regardless of the value of $\liminf \|g_p(x_c^k)\|$. *In other words,* ρ_{max}^k *remains bounded away from zero.*

Proof. Let us consider the first part of the lemma. To prove that ρ_{max} doesn't change at iteration $k+1$, we just need to show that $\Delta L_V^{k+1} < -\Delta L_H^k/2$ (see step 3.2 of Algorithm 2.1). From Lemma 3.2, this result is attained whenever

$$\Delta L_V^k < \frac{\xi_1\xi_2}{2}\|g_p(x_c^k)\|^2 \tag{3.31}$$

and

$$\Delta L_V^k < \frac{\xi_1\xi_3}{2}\sqrt{\rho^k}\|g_p(x_c^k)\|. \tag{3.32}$$

Condition (3.31) can be easily obtained from Lemma 3.3 and (3.27), taking $\xi_5 = \xi_5^a \equiv \xi_1\xi_2/(2\xi_4)$. To obtain (3.32), we need a few more steps. Firstly, we use (2.8) and (3.5) to write

$$\sqrt{\rho_{max}^k} \le 10^2\sqrt{\rho^k}\frac{(\xi_0+1)^{1/2}}{\|g_p(x_c^k)\|^{1/2}}. \tag{3.33}$$

Then, taking the square root from both sides of (3.27) and combining the result with (3.33), we get

$$\rho_{max} \le \sqrt{\xi_5}10^{-2}\sqrt{\xi_0+1}\sqrt{\rho^k}. \tag{3.34}$$

Now, defining $\xi_5 = \xi_5^b \equiv 10^{-4}\xi_1^2\xi_2^2/[4\xi_4^2(\xi_0+1)]$, and using Lemma 3.3 and (3.27), we obtain (3.32). The desired result follows from taking $\xi_5 = \min\{\xi_5^a,\, \xi_5^b\}$.

In order to prove item (*ii*), let us define $b = \liminf(\|g_p(x_c^k)\|)$ and choose an index $\overline{k}_0$ such that $\|g_p(x_c^k)\| > b/2$, for $k \ge \overline{k}_0$. Then, as we proved above, $\rho_{max}^k \ge \min\{\rho_{max}^{\overline{k}_0}, \xi_5 b/2, \xi_6\}$, for $k > k_0$. Thus, ρ_{max} will never be decreased after a certain iteration k_0, as claimed.

To prove the third part of the lemma, we begin observing that (1.3)-(1.4) and H1-H3 imply that $\lambda_{LS}(x)$ and $g_p(x)$ are well defined and of class C^1 in a compact neighborhood of $\overline{\mathcal{Z}}$, the closure of $Z = \{x_c^k\}$. Therefore, $\lambda_{LS}(x)$ and $g_p(x)$ are Lipschitz continuous on the iterates, in the sense that,

$$\|\lambda_{LS}(x_c^{k+1}) - \lambda_{LS}(x_c^k)\| = O(\|x_c^{k+1} - x_c^k\|) \tag{3.35}$$

and

$$\|g_p(x_c^{k+1}) - g_p(x_c^k)\| = O(\|x_c^{k+1} - x_c^k\|). \tag{3.36}$$

From (3.1), (2.9), (2.7) and (3.29) we get

$$\|x_c^{k+1} - x_c^k\| \le \|x_c^{k+1} - x^k\| + \|x^k - x_c^k\| = O(\|g_p(x_c^k)\|), \tag{3.37}$$

and from (3.36) and (3.37) we obtain

$$\|g_p(x_c^{k+1})\| = O(\|g_p(x_c^k)\|). \tag{3.38}$$

Noticing that

$$L(x_c^{k+1}, \lambda^{k+1}) = L(x_c^{k+1}, \lambda_{LS}(x_c^{k+1})) + [\lambda^{k+1} - \lambda_{LS}(x_c^{k+1})]^T h(x_c^{k+1})$$

and

$$L(x^k,\lambda^k) = L(x^k,\lambda_{LS}(x_c^{k+1})) - [\lambda_{LS}(x_c^{k+1}) - \lambda_{LS}(x_c^k)]^T h(x^k) - [\lambda_{LS}(x_c^k) - \lambda^k]^T h(x^k).$$

we get the following decomposition of ΔL_V^{k+1} into a sum of four terms:

$$\begin{aligned}\Delta L_V^{k+1} &= L(x_c^{k+1},\lambda^{k+1}) - L(x^k,\lambda^k)\\ &= [L(x_c^{k+1},\lambda_{LS}(x_c^{k+1})) - L(x^k,\lambda_{LS}(x_c^{k+1}))] +\\ &\quad [\lambda^{k+1} - \lambda_{LS}(x_c^{k+1})]^T h(x_c^{k+1}) +\\ &\quad [\lambda_{LS}(x_c^{k+1}) - \lambda_{LS}(x_c^k)]^T h(x^k) +\\ &\quad [\lambda_{LS}(x_c^k) - \lambda^k]^T h(x^k).\end{aligned} \tag{3.39}$$

Using a Taylor expansion, (1.4), Hypothesis H2, (2.9), (3.1), (3.4)-(3.7) and (3.38), we get

$$\begin{aligned}&L(x_c^{k+1},\lambda_{LS}(x_c^{k+1})) - L(x^k,\lambda_{LS}(x_c^{k+1})) =\\ &g_p(x_c^{k+1})^T(x_c^{k+1} - x^k) + O(\|x_c^{k+1} - x^k\|^2) =\\ &O(\|g_p(x_c^k)\|\rho^k + {\rho^k}^2).\end{aligned} \tag{3.40}$$

Since (2.7) implies that ${\rho^k}^2 \le \rho_{max}^k\|g_p(x_c^k)\|\rho^k$, equation (3.40) assures that the first term in the right side of (3.39) is $O(\|g_p(x_c^k)\|\rho^k)$.

From (3.30) and (3.38), we deduce that $\|\lambda^{k+1} - \lambda_{LS}(x_c^{k+1})\|$ and $\|\lambda_{LS}(x_c^k) - \lambda^k\|$ are $O(\|g_p(x_c^k)\|)$. From (3.35) and (3.37), we also obtain $\|\lambda_{LS}(x_c^{k+1}) - \lambda_{LS}(x_c^k)\| = O(\|g_p(x_c^k)\|)$. Finally, (2.9) assures that $\|h(x_c^{k+1})\| \le \|h(x^k)\| \le 2\rho^k$. This implies that the remaining three terms in the right side of (3.39) are also $O(\|g_p(x_c^k)\|\rho^k)$. Together with (2.7), this ensures that there exists $\xi_7 > 0$, such that

$$\Delta L_V^{k+1} \le \xi_7\rho_{max}^k\|g_p(x_c^k)\|^2. \tag{3.41}$$

Let $\overline{\rho}_{max}$ be defined by

$$\overline{\rho}_{max} = \min\left\{\frac{\xi_1\xi_2}{2\xi_7},\ \frac{10^{-4}}{4\xi_0(\xi_0+1)}\left(\frac{\xi_1\xi_3}{\xi_7}\right)^2\right\}. \tag{3.42}$$

With arguments entirely similar to those used to show (3.31)-(3.32), we can prove, from (3.41) and (3.42), that, if $\rho_{max}^{k_0} < \overline{\rho}_{max}$ and $k \ge k_0$, then $\Delta L_V^{k+1} < -\frac{1}{2}\Delta L_H^k$. Therefore, ρ_{max}^k does not change after k_0. □

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 3.6. *Under H1-H4, either DCI stops at 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 (3.29) and (3.30), then every accumulation point of ${x_c}^k$ is stationary for (1.1).*

Proof. Let us suppose, by contradiction, that $\liminf(\|g_p(x_c^k)\|) = 2b > 0$, and let $\overline{k}_0$ be such that $\|g_p(x_c^k)\|) > b$, for $k \geq \overline{k}_0$. In this case, item *(ii)* from Lemma 3.5 allows us to choose $k_0 \geq \overline{k}_0$, such that, for every $k \geq k_0$, $\rho_{max}^k = \rho_{max}^{k_0}$. Together with (2.8) and (3.5), this implies that $\rho^k \geq 10^{-4}\rho_{max}^{k_0} b/[2(\xi_0+1)]$.

Now, using (3.24) and (3.13), we can assure that, for $k > k_0$,

$$\begin{aligned} L(x_c^k, \lambda^k) - L(x_c^{k_0}, \lambda^{k_0}) &= \sum_{i=k_0+1}^{k} \Delta L_c^i \leq \frac{1}{4} \sum_{i=k_0}^{k-1} \Delta L_H^i + r^{k_0} \\ &\leq -(k-k_0)\theta + r^{k_0} \to -\infty, \end{aligned}$$

where

$$\theta = \frac{1}{4}\xi_1 b \min\left\{\xi_2 b,\ 10^{-2}\xi_3\sqrt{\frac{b\rho_{max}^{k_0}}{\xi_0+1}}\right\} > 0. \tag{3.43}$$

This contradicts H1-H2, imposing $\liminf(\|g_p(x_c^k)\|) = 0$.

For the second part of the theorem, let's assume that (3.29) and (3.30) apply. In this case, Lemma 3.5 ensures that $\rho_{max}^k = \rho_{max}^{k_0}$, for some k_0 and every $k \geq k_0$.

Suppose, by contradiction, that $\|g_p(x_c)^{k_\ell})\| \geq b > 0$ for an infinite subsequence $\{k_\ell\}$. Let n_k be the number of iterations between k_0 and k, for some index $k \in \{k_\ell\}$. In this case, using (3.24) and (3.13) again, and taking $n_k \to \infty$, we have,

$$\begin{aligned} L(x_c^k, \lambda^k) - L(x_c^{k_0}, \lambda^{k_0}) &= \sum_{i=k_0+1}^{k} \Delta L_c^i \leq \frac{1}{4} \sum_{i=k_0}^{k-1} \Delta L_H^i + r^{k_0} \\ &\leq -n_k\theta + r^{k_0} \to -\infty, \end{aligned} \tag{3.44}$$

where θ is given by (3.43). This also contradicts H1-H2, implying that $\|g_p(x_c^{k_\ell})\| \to 0$ for every subsequence of x_c^k. □

Theorem 3.6 can equally be proved if we admit inexact solutions for the subproblems associated with Algorithm 2.1, using fairly loose conditions on the residues for accepting the step. For instance, we could relax the condition $\nabla h(x_c)\,\delta_t = 0$, or admit inexact computations of $g_p(x)$ and the solution of the quadratic subproblem (2.4). Although this modification can be interesting for large-scale problems and would not change the proofs significantly, we preferred not to present it in this article, since its details might look rather messy at a first reading. We also believe that the second order correction would play a very interesting role if inexact methods were used.

4. Local Convergence.

Let $N(M)$ represent the null space of M. Let also $\{x_c^k\}$ and $\{x^k\}$ be sequences generated by Algorithm 2.1, converging to x^*, a "good minimizer" of problem 1.1. By "good minimizer" we mean that $\nabla h(x^*)$ has full row rank, $\nabla f(x^*) = -\nabla h(x^*)^T\lambda^*$, with $\lambda^* = \lambda_{LS}(x^*)$, and there is a constant $\mu_1 > 0$, such that, for $y \in N(\nabla h(x^*))$,

$$\mu_1\|y\|^2 \leq y^T\nabla_{xx}^2 L(x^*, \lambda^*)y. \tag{4.1}$$

In this section, we will restrict our attention to a neighborhood V^* of x^*, where, due to the fact that h is C^2 and $\nabla h(x^*)$ has full row rank, the orthogonal projector onto $N(\nabla h(x))$, i.e. $P(x) = I - \nabla h(x)^T(\nabla h(x)\nabla h(x)^T)^{-1}\nabla h(x)$, is Lipschitz continuous. Sometimes we will use the term δ_c to represent the "full" step taken by the algorithm, i.e., $\delta_c = x_c^{k+1} - x_c^k = \delta_H^k + \delta_V^{k+1}$.

Besides considering hypotheses H1-H4, our analysis of the local convergence of x_c^k and x^k, will be based on four additional local assumptions. The first three of these assumptions are used in the proof of Lemma 4.1, and are described below.

A1: $\lambda^k - \lambda_{LS}(x_c^k) = O(\|g_p(x_c^k)\|)$.

A2: B^k is assimptotically uniformly positive-definite in the tangent space to the restrictions, which means that, in some neighbourhood of x^*, we can redefine μ_1 so

$$\mu_1\|y\|^2 \le y^T B^k y \le \mu_2\|y\|^2 \tag{4.2}$$

for $y \in N(\nabla h(x_c^k))$, where μ_2 is just the constant ξ_0 defined in (3.8).

A3: Let δ_{HN}^k be the minimizer of the quadratic model (2.6) without the trust region constraint. We assume that, whenever δ_{HN}^k is within the trust region ($\|\delta_{HN}^k\| \le \Delta$), it is the the first horizontal step tried by the algorithm. Besides, we also suppose that it satisfies

$$P(x_c^k)(B^k - \nabla_{xx}^2 L(x^*, \lambda^*))\delta_{HN}^k = o(\|\delta_{HN}^k\|).$$

Assumption A1 is not a stringent condition. Usual estimates for the Lagrange multipliers (see, for example, [42]) satisfy $\|\lambda^k - \lambda^*\| = O(\|x_c^k - x^*\|)$, so A1 can be guaranteed by our Lemma 4.2, presented below, along with (1.5) and (1.2).

Assumptions A2 and A3 are essentially equivalent to standard conditions for superlinear convergence in two steps of SQP quasi-Newton methods, like those established by Powell in [35]. These assumptions are satisfied, for example, if we define $B^k = \nabla_{xx} L(x_c^k, \lambda^k)$. In a future paper, we intend to incorporate in our analysis the use of secant reduced Hessian approximation schemes, as well as the inexact solution of the subproblems involved, in a way that A2 and A3 are satisfied.

From now on, we also suppose that $\delta_{soc} = 0$. This is done only to simplify the exposition. In fact, the arguments presented below are still valid if we consider $\delta_{soc} = O(\|\delta_t\|^2)$.

Since $\nabla h(x)$ and $\nabla_{xx}^2 L(x, \lambda)$ are continuous and $\nabla h(x^*)$ has full row rank, our assumptions imply that there is a constant $\mu_3 > 0$ and a neighborhood V^* of x^* such that, for x, $x_c^k \in V^*$,

$$\mu_3\|\lambda\| \le \|\nabla h(x)^T \lambda\|, \quad \text{for } \lambda \in I\!R^m, \tag{4.3}$$

$$P(x_c^k)(B^k - \nabla_{xx}^2 L(x_c^k, \lambda_{LS}(x_c^k)))\delta_{HN}^k = o(\|\delta_{HN}^k\|), \quad \text{and} \tag{4.4}$$

$$P(x_c^k)(B^k - \nabla_{xx}^2 L(x_c^k, \lambda^k))\delta_{HN}^k = o(\|\delta_{HN}^k\|). \tag{4.5}$$

Let Z^k be a matrix whose columns form an orthonormal basis for the null space $N(\nabla h(x_c^k))$. We can define the global minimizer of the quadratic model in the tangent space as $\delta_{HN}^k = Z^k \nu^k \in N(\nabla h(x_c^k))$. This point clearly satisfies

$$(Z^k)^T(B^k \delta_{HN}^k + \nabla_x f(x_c^k)) = (Z^k)^T B^k Z^k \nu^k + (Z^k)^T g_p(x_c^k) = 0. \tag{4.6}$$

From (4.2) and the fact that $(Z^k)^T Z^k = I$, matrix $((Z^k)^T B^k Z^k)^{-1}$ satisfies, in the neighborhood V^*, and for all $u \in R^{n-m}$,

$$\frac{1}{\mu_2}\|u\|^2 \le u^T((Z^k)^T B^k Z^k)^{-1} u \le \frac{1}{\mu_1}\|u\|^2. \tag{4.7}$$

In the next lemma we will prove the eventual acceptance, in Algorithm 2.1, of

$$\delta_{HN}^k = -Z^k \nu^k = -Z^k((Z^k)^T B^k Z^k)^{-1}(Z^k)^T g_p(x_c^k). \tag{4.8}$$

LEMMA 4.1. *δ_{HN}^k is accepted by Algorithm 2.1, for k sufficiently large.*

Proof. Combining (4.7) and (4.8), we have that

$$\|\delta_{HN}^k\| \le \frac{1}{\mu_1}\|g_p(x_c^k)\|, \tag{4.9}$$

for $x_c^k \in V^*$. Because the trust region radius satisfies $\Delta \ge \Delta_{min}$ at the beginning of each iteration, assumption A3 and (4.9) imply that, in a suitable V^*, δ_H^+ will be the first horizontal step tried by Algorithm 2.1.

For $\nu \in I\!R^{n-m}$, the *reduced* polynomial

$$\overline{q}(\nu) = q(Z^k\nu) = ((Z^k)^T g_p(x_c^k))^T\nu + \nu^T((Z^k)^T B^k Z^k)\nu$$

has degree 2, with positive definite quadratic form. Therefore, its minimum, $\overline{q}(\nu^k)$, satisfies (see [14]),

$$\begin{aligned} q(\delta_{HN}^k) &= q(Z^k\nu^k) = \overline{q}(\nu^k) \\ &= -\frac{1}{2}((Z^k)^T g_p(x_c^k))^T((Z^k)^T B^k Z^k)^{-1}(Z^k)^T g_p(x_c^k) \\ &\le -\frac{1}{2\mu_2}\|g_p(x_c^k)\|^2, \end{aligned} \tag{4.10}$$

where the last inequality comes from (4.7).

Now, using a Taylor expansion, the fact that $\delta_{HN}^k = P(x_c^k)\delta_{HN}^k$, (4.5) and (4.9), we get

$$\begin{aligned} \Delta L_H^+ &= L(x_c^k + \delta_{HN}^k, \lambda^k) - L(x_c^k, \lambda^k) \\ &= g_p(x_c^k)^T\delta_{HN}^k + \frac{1}{2}\delta_{HN}^{k\,T}\nabla_{xx}{}^2 L(x_c^k)\delta_{HN}^k + o(\|\delta_{HN}^k\|^2) \\ &= g_p(x_c^k)^T\delta_{HN}^k + \frac{1}{2}\delta_{HN}^{k\,T} B^k\delta_{HN}^k + o(\|\delta_{HN}^k\|^2) \\ &= q(\delta_{HN}^k) + o(\|g_p(x_c^k)\|^2). \end{aligned} \tag{4.11}$$

It follows from (4.10-4.11) that

$$|r| = |\Delta L_H^+/q(\delta_{HN}^k)| = 1 + \frac{o(\|g_p(x_c^k)\|^2)}{\|g_p(x_c^k)\|^2/(2\mu_2)} \to 1,$$

so one of the acceptance conditions stated at step 4.2 of Algorithm 2.1 is satisfied for k sufficiently large.

Let us now prove that the other acceptance condition, $\|h(x_c^k + \delta_{HN}^k)\| \le 2\rho^k$, also holds. From (4.9), assumption A1 and Lemma 3.5, there exists k_0 sufficiently large so $\rho_{max}^k = \rho_{max}^{k_0} > 0$ for $k \ge k_0$. Therefore, (2.8) and (3.5) guarantee that, for $k \ge k_0$, $\|g_p(x_c^k)\| \le \beta\rho^k$, where $\beta = 10^4(1+\xi_0)/\rho_{max}^{k_0}$. Together with (3.12) and (4.9), this implies that, for k sufficiently large,

$$\begin{aligned} \|h(x_c^k + \delta_{HN}^k)\| \le \|h(x_c^k)\| + \overline{\xi}_0\|\delta_{HN}^k\|^2 &\le \|h(x_c^k)\| + \frac{\overline{\xi}_0}{\mu_1{}^2}\|g_p(x_c^k)\|^2 \\ &\le \rho^k(1 + \beta\frac{\xi_0}{\mu_1{}^2}\|g_p(x_c^k)\|). \end{aligned}$$

Since, for k sufficiently large, $\beta\dfrac{\xi_0}{{\mu_1}^2}\|g_p(x_c^k)\| < 1$, the step δ_{HN}^k will eventually be accepted. □

This lemma is based on the fact that $\Delta \geq \Delta_{min}$ at the beginning of an iteration. This condition can be removed if we replace A3 by the more restrictive assumption

A3′: Let δ_+ be obtained as a positive linear combination of δ_{CP} and δ_{HN}. Let also δ_+ satisfy

$$P(x_c^k)(B^k - \nabla^2_{xx}L(x^*, \lambda^*))\delta_+ = o(\|\delta_+\|).$$

In this case, we can also prove that δ_+ is accepted whenever x_c^k and x^k are in a suitable neighborhood V^* of x^*. Therefore, there exists k_1 such that Δ^k is not reduced for $k > k_1$, so we can restrict our attention to the case where $\delta_H^k = \delta_{HN}^k$.

Notice that the dynamic control of the infeasibility might force us to compute more than one single vertical step δ_V^+, starting from x^k, if $\|g_p(x^k + \delta_V^+)\|$ is too small.

At the beginning of iteration $k+1$, we have $x_c = x^k$, while the vertical step ends at $x_c = x_c^{k+1}$. In order to avoid unnecessary updates of $\nabla h(x_c^k)$, we state our fourth local assumption:

A4: Each non-zero vertical step $\delta_V^{k+1} = x_c^{k+1} - x^k$, is computed taking one or more steps in the form

$$\delta_V^+ = -A^T(AA^T)^{-1}h(x_c), \tag{4.12}$$

where A satisfies

$$\|A - \nabla h(x_c)\| = O(\|g_p(x_c^k)\|). \tag{4.13}$$

Vector δ_V^+ given by (4.12) is the usual Gauss-Newton step for solving $h(x) = 0$, with an approximation A for the Jacobian $\nabla h(x_c)$. Using a Taylor expansion, (4.3), (4.12), (4.13) and the continuity of $\nabla h(x)$, it is easy to show that, if $x_c^{k+1} \neq x^k$, then the first vertical step δ_V^+ of iteration $k+1$ satisfies

$$\|\delta_V^+\| = O(\|h(x^k)\|), \quad \text{and} \tag{4.14}$$

$$\|h(x_c^{k+1})\| \leq \|h(x^k + \delta_V^+)\| = o(\|h(x^k)\|). \tag{4.15}$$

As $g_p(x^k)$ becomes small, it is natural to force a restoration after each horizontal step. This can be done, for instance, by choosing ρ^k slightly smaller than $\|h(x^k)\|$. In [11] and [12], Colemann and Conn analyze algorithms that alternate a horizontal step with a single vertical restoration step. Under local assumptions similar to those presented here, these so called horizontal-vertical algorithms are superlinear convergent in 2 steps. Colemann and Conn also point out in [11] that the restoration step adopted by their methods differ from the usual SQP vertical step, since it is based on $x^k + \delta_H$ while, in the SQP framework, x^k is used to define the vertical subproblem.

One difference between these horizontal-vertical algorithms and ours is that we admit more than one vertical step like (4.12) at each iteration, as mentioned above. Our main result on the local behavior of the algorithm is based on the following lemma that, in a sense, expresses analytically the "good" structure we have in the neighborhood of a "good" KKT point. This approach is similar to the one used by Powell in [35], although his focus was restricted to each sequence generated by an SQP algorithm.

It is well known that the function $\phi(x) = \|h(x)\| + \|g_p(x)\|$ can be used to measure how close $x \in V^*$ is to x^*. However, we need a stronger result. We want to say that, in a vicinity of x^*, $\phi(x)$ is *equivalent* to $\|x - x^*\|$, in the sense that $\|x - x^*\| = \Theta(\phi(x))$, i.e. $\|x - x^*\| = O(\phi(x))$ and $\phi(x) = O(\|x - x^*\|)$.

LEMMA 4.2. *There is a neighborhood V^* of x^*, where*

$$\|x - x^*\| = \Theta(\|h(x)\| + \|g_p(x)\|) \tag{4.16}$$

Proof. We just have to show that

$$\|x - x^*\| = O(\|h(x)\|) + O(\|g_p(x)\|), \tag{4.17}$$

The converse follows trivially from the fact that $h(x)$ and $g_p(x)$ are Lipschitz continuous in V^*.

In [19], Fletcher shows that the SQP method has quadratic convergence to a good minimizer x^*. The same argument can be used to prove that, if δ_x is an SQP step from $x \in V^*$ to $x_+ = x + \delta_x$, where V^* is a suitable neighborhood of x^*, then we have $x_+ - x^* = O(\|x - x^*\|^2)$. It is also easy to show (see (10.1.11)-(10.1.13) in [19]) that δ_x satisfies $\delta_x = x_+ - x = O(\|g_p(x)\|) + O(\|h(x)\|)$. From these two relations, it follows that $x - x^* = (x_+ - x^*) - (x_+ - x) = O(\|x_+ - x\|) = O(\|g_p(x)\|) + O(\|h(x)\|)$. □

Byrd [7] and Yuan [46] give examples showing that we can't expect superlinear convergence in one step for x_c^k. However, Byrd [8] points to the possibility of obtaining superlinear convergence in one step for x^k. To understand why this happens, notice that a vertical step that moves from x^k to x_c^{k+1} approaches the feasible set in a "superlinear" way. After that, the horizontal step superlinearly pushes x_c^{k+1} towards the dual manifold $\mathcal{L}^* = \{x \in V^* : g_p(x) = 0\}$, with δ_H^{k+1} tangent to the feasible directions. Therefore, this horizontal step does not destroy the "vertical superlinear approximation". On the other hand, if we start at x_c^k, the superlinear convergence in a single step cannot be guaranteed since the vertical step δ_V^{k+1} usually isn't tangent to $\mathcal{L}^*$ and, for this reason, δ_V^{k+1} can partly spoil the good approach to $\mathcal{L}^*$ obtained by δ_H^k.

To close this section, we present our main theorem, showing that the algorithm is 2-step superlinearly convergent. Besides, convergence in one step can also be obtained if we call a restoration at each iteration.

THEOREM 4.3. *Under H1-H4 and A1-A4, x^k and x_c^k are 2-step superlinearly convergent to x^*. If a restoration is computed at each x^k, then x^k converges superlinearly to x^*.*

Proof. Since $\|x^{k+1} - x^*\| \le \|x^{k+1} - x_c^{k+1}\| + \|x_c^{k+1} - x^*\|$, equations (4.9) and (4.16) imply that

$$\|x^{k+1} - x^*\| = O(\|x_c^{k+1} - x^*\|). \tag{4.18}$$

Besides, observing that $\|x_c^{k+1} - x^*\| \le \|x^k - x_c^{k+1}\| + \|x^k - x^*\|$, and using (3.1) and (4.16), we have

$$\|x_c^{k+1} - x^*\| = O(\|x^k - x^*\|). \tag{4.19}$$

In order to prove the 2-step superlinear convergence, we shall use the following relations:

$$g_p(x^k) = o(\|x_c^k - x^*\|), \tag{4.20}$$

$$g_p(x_c^{k+1}) = o(\|x_c^{k-1} - x^*\|), \tag{4.21}$$
$$h(x^k) = o(\|x_c^{k-1} - x^*\|), \text{ and} \tag{4.22}$$
$$h(x_c^{k+1}) = o(\|x_c^k - x^*\|). \tag{4.23}$$

Let us show that these relations are valid, starting with (4.20). Using a Taylor expansion, along with (1.4), we have, for $x^k \in V^*$,

$$\begin{aligned} \|g_p(x^k)\| &= \|P(x^k)\Gamma^k\| + o(\|\delta_H^k\|) \\ &\leq \|(P(x^k) - P(x_c^k))\Gamma^k\| + \|P(x_c^k)\Gamma^k\| + o(\|\delta_H^k\|), \end{aligned} \tag{4.24}$$

where $\Gamma^k = g_p(x_c^k) + \nabla_{xx}^2 L(x_c^k, \lambda_{LS}(x_c^k))\delta_H^k$.

The continuity of $P(x)$ in V^* and (4.9) give us

$$\|(P(x^k) - P(x_c^k))\Gamma^k\| = o(\|\Gamma^k\|) = o(\|g_p(x_c^k)\|). \tag{4.25}$$

Besides, (4.4), (4.8) and (4.9) imply that

$$\|P(x_c^k)\Gamma^k\| = \|P(x_c^k)(g_p(x_c^k) + B^k\delta_H^k)\| + o(\|\delta_H^k\|) = o(\|g_p(x_c^k)\|). \tag{4.26}$$

Replacing (4.25) and (4.26) into (4.24) and considering also (4.16), we get (4.20).

To prove (4.23), we need to consider separately two situations. First, let k_i be an infinite subsequence at which no vertical step was made, i.e. $x_c^{k_i+1} = x^{k_i}$. In this case, the dynamic control of the infeasibility, together with (4.20), imply that

$$\|h(x_c^{k_i+1})\| = O(\|g_p(x_c^{k_i+1})\|) = O(\|g_p(x^{k_i})\|) = o(\|x_c^{k_i} - x^*\|). \tag{4.27}$$

Let's now consider an infinite subsequence of iterations k_j at which at least one vertical step δ_V^+ satisfying A4 was made. In this case, (4.15) and (4.16) imply that

$$\|h(x_c^{k_j+1})\| \leq \|h(x_c^{k_j} + \delta_V^+)\| = o(\|h(x^{k_j})\|) = o(\|x^{k_j} - x^*\|). \tag{4.28}$$

Equation (4.23) follows directly from (4.18), (4.27) and (4.28).

Combining (3.12), (4.9), (4.16), (4.18), (4.19) and (4.23), we can write

$$\begin{aligned} \|h(x^k)\| &= \|h(x_c^k)\| + O(\|\delta_H{}^k\|^2) = h(x_c^k) + O(\|g_p(x_c^k)\|^2) \\ &= \|h(x_c^k)\| + O(\|x_c^k - x^*\|^2) = o(\|x_c^{k-1} - x^*\|), \end{aligned} \tag{4.29}$$

so (4.22) has been proved.

Finally, to obtain equation (4.21), we use a Taylor expansion, (3.1), (4.18), (4.19), (4.20) and (4.22), so

$$\begin{aligned} \|g_p(x_c^{k+1})\| &= \|g_p(x^k)\| + O(\|x^k - x_c^{k+1}\|) \\ &= \|g_p(x^k)\| + O(\|h(x^k)\|) = o(\|x_c^{k-1} - x^*\|). \end{aligned}$$

The 2-step superlinear convergence of x_c^k and x^k follows from (4.16) and (4.18)-(4.23), since these equations imply that

$$\|x^{k+1} - x^*\| = O(\|g_p(x^{k+1})\| + \|h(x^{k+1})\|) = o(\|x^{k-1} - x^*\|), \text{ and} \tag{4.30}$$
$$\|x_c^{k+1} - x^*\| = O(\|g_p(x_c^{k+1})\| + \|h(x_c^{k+1})\|) = o(\|x_c^{k-1} - x^*\|). \tag{4.31}$$

In order to conclude the proof, let's assume a non zero restoration step is done at each iteration. Then, (3.12) and (4.15), together with (4.9), (4.16), and (4.19), allow us to improve (4.22), obtaining

$$\|h(x^k)\| = \|h(x_c^k)\| + O(\|g_p(x_c^k)\|^2) = o(\|x^{k-1} - x^*\|). \tag{4.32}$$

Replacing (4.20) and (4.32) into (4.30), and considering also (4.19), we get

$$\begin{aligned}\|x^{k+1} - x^*\| &= O(\|g_p(x^{k+1})\| + \|h(x^{k+1})\|)\\ &= o(\|x_c^{k+1} - x^*\| + \|x^k - x^*\|) = o(\|x^k - x^*\|),\end{aligned}$$

so the desired superlinear convergence of x^k to x^* is attained. □

5. Numerical experience.

The success of an algorithm is based not only on its theoretical convergence results, but also on its practical behavior. 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 [22].

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.

5.1. A practical implementation of the algorithm.

We begin the detailed description of the 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.

5.1.1. Vertical step.

Whenever $\|h(x_c)\| > \rho$ at the beginning of an iteration, we need to reduce the infeasibility. Unfortunately, this test is tricky to perform, since ρ depends on $n_p(x_c)$, and this term, in turn, depends on the matrix $\nabla h(x_c)$. Naturally, it would not be wise to compute $\nabla h(x_c)$ 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

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

The restoration is done applying Powell's *dogleg* method [34] to the constrained linear least squares problem (2.5), replacing x by x_c. Again, the solution of this problem depends on $\nabla h(x_c)$. Therefore, the first time we try to solve (2.5), we use $A = \nabla h(x_c^{k-1})$. If the infeasibility is not sufficiently reduced, we define $A = \nabla h(x_c)$ and recompute the step.

To find an approximate solution for the 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

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

where

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

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

$$s_{NS} = -A(x_c)^T (A(x_c)A(x_c)^T)^{-1} h(x_c). \tag{5.1}$$

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 which A is recalculated.

After the restoration, $\nabla h(x_c)$ 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 $10^{-4} n_p\, \rho_{max} \leq \rho \leq n_p\, \rho_{max}$, we keep this value. Otherwise, we simply define

$$\rho = \min\{n_p\, \rho_{max},\, 0.75\rho_{max}\}.$$

The reduction obtained by the dogleg method may be small depending on the curvature of h. When this happens, we abandon the constrained linear least squares problem and try to apply the Moré and Thuente line search algorithm [32] to the unconstrained nonlinear least squares problem

$$\textit{minimize} \quad \|h(x)\|^2, \tag{5.2}$$

using a BFGS approximation for the second-order part of the Hessian of the objective function [15].

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.

5.1.2. Horizontal step. The horizontal step of the method consists in solving the quadratic programming problem (2.6). If Z is a matrix that spans the null space of $\nabla h(x_c)$, then it is possible to rewrite (2.6) as the constrained nonlinear programming problem

$$\begin{aligned} &\textit{minimize} \;\; g(x_c)^T Z v + \frac{1}{2} v^T Z^T B Z v \\ &\textit{subject to} \;\; \|Zv\|_\infty \leq \Delta, \end{aligned} \tag{5.3}$$

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 (5.3), as we did in the vertical step. Instead of that, we use the Steihaug-Toint method [41, 43], 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 [24].

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 on the boundary of the trust region is chosen.

5.1.3. 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 (5.1), in the dogleg method;
- the Lagrange multipliers (1.3) and, consequently, the projected gradient (1.4);
- the second order correction (see (5.4) 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.

As an attractive alternative, we could use the augmented system approach to solve such systems, since it reduces the fill-in produced by dense rows in A, and keeps the condition number of the matrix under control. Direct methods for solving symmetric indefinite augmented systems are presented, for example, in [6, 40], while iterative approaches are introduced in [20, 5], just to cite a few references. We plan to include one or more of these algorithms into our code in a near future.

5.1.4. 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

$$\begin{aligned} \delta_{soc} &= argmin\{\|\nabla h(x_c)\,\delta + (h(x_c+\delta_t) - h(x_c))\|\} \\ &= -\nabla h(x_c)^T(\nabla h(x_c)\nabla h(x_c)^T)^{-1}(h(x_c+\delta_t) - h(x_c)). \end{aligned} \tag{5.4}$$

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

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.

5.2. 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 [22]. The selected problems are presented in Table 5.1. The number of variables of the problem is given by n, while m is the number of constraints.

TABLE 5.1
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	6402	3201	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	13284	6724	ORTHRGDM	2003	1000
GRIDNETE	13284	6724	ORTHRGDS	1003	500
GRIDNETH	13284	6724			

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 with two freely available nonlinear programming solvers. The first is Lancelot-B, the well known algorithm distributed along with the Galahad library [21]. The second is Ipopt (version 3.3.3) [44], an interior point method that also tackles equality constrained problems quite well. Both codes include a nice 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. Exact first and second derivatives were computed by all of the methods.

The DCI algorithm was designed to declare convergence when both $\|h(x)\| < \epsilon_h$ and $n_p < \epsilon_p$, as well as when $\rho_{max} < \epsilon_r$. However, since Lancelot-B 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-B. The Ipopt stopping tolerances were changed accordingly. Default values were used for the remaining Ipopt parameters. The default settings were also used in Lancelot-B, except for the maximum number of iterations that was increased to 10000.

Other parameters used in the DCI algorithm are

$$\begin{aligned} \rho^0_{max} &= \max\{10^{-5},\, 5.1\|h(x^0)\|,\, 50\, n_p(x^0)\}, \\ \Delta^0 = \Delta^0_{VS} &= \max\{10\|x^0\|,\, 10^5\}, \end{aligned} \tag{5.5}$$

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

The comparison of the methods were done using the performance profiles defined by Dolan and Moré [17]. 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 Lancelot-B, Ipopt and DCI and are shown in Figure 5.1. One can deduce from this figure that the DCI algorithm took less time than Lancelot-B and Ipopt to obtain the solution of 47% of the problems, while Ipopt was the best solver for 45% of the problems and Lancelot-B took less time in only 9.4% of the cases. Ipopt outperforms DCI for t between 1.5 and 4.5 but, in general, we may say that DCI presented the best performance among the solvers.

DCI and Lancelot-B obtained an optimal solution (i.e. an stationary point for (1.1)) for all of the problems. The Ipopt code, in turn, converged to a point of local infeasibility when solving the LUKVLE16 problem, and ran out of memory after spending 2920 seconds searching the solution of the LUKVLE11 problem. For all of the remaining problems, Ipopt also obtained an optimal solution.

To close this section, let us focus our analysis on the behavior of the restoration scheme adopted in DCI, summarized in Figure 5.2. Each point in the figure represents

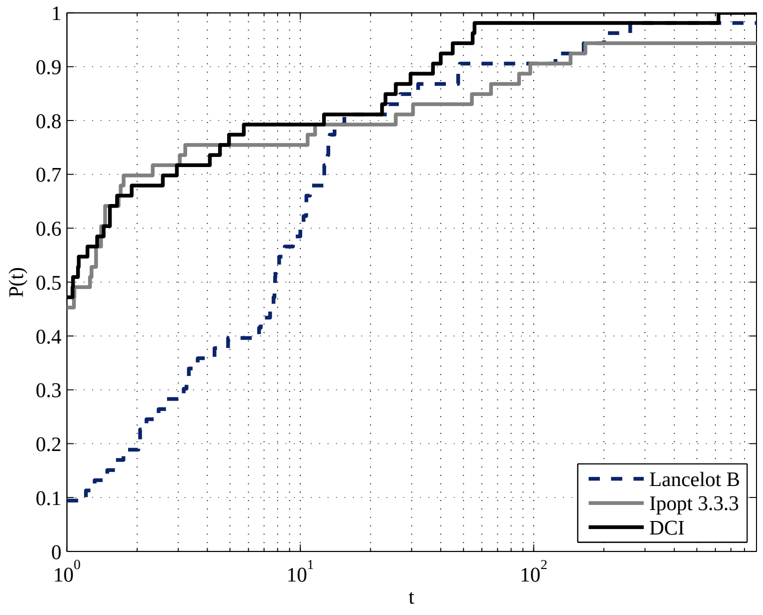

FIG. 5.1. *Performance profiles for 53 CUTEr problems.*

one CUTEr problem. The horizontal coordinate of a point is the percentage of the number of iterations in which only one restoration was done. The vertical coordinate is the percentage of iterations in which more than one restoration was needed. Diagonal lines were included to group the problems by the percentage of iterations with one or more restorations.

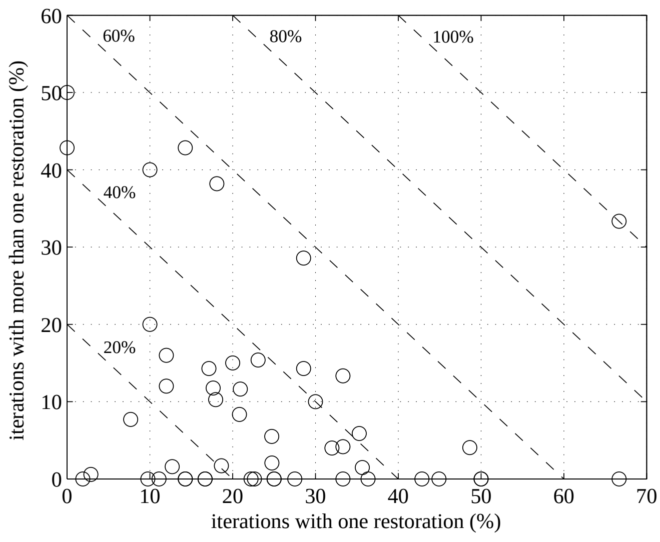

FIG. 5.2. *Percentage of iterations with one or more restorations for the 53 CUTEr problems.*

The results obtained for the 53 CUTEr problems showed that, on average, the DCI algorithm performed one restoration in 24.3% of the iterations, while it was necessary to perform more than one restoration in only 9.2% of the iterations. Summing the figures, we observe that no restoration was made in about two thirds of the iterations, on average. Besides, if we consider only the iterations in which more than one restoration was done, the number of restorations was equal to 2 in 62,1% of the cases.

Our experiments with the CUTEr problems also revealed 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 (5.5). 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.

6. 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). Besides, it is also superlinearly convergent under some mild assumptions.

Our current implementation of the algorithm works well when applied to medium-sized problems, so we believe that 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:

- the use of an augmented system approach to solve the linear systems;
- the reformulation of the algorithm so inexact solutions for the linear subroutines are admitted;
- the use of BFGS approximations to the Hessian of the Lagrangian when computing the horizontal step;
- the definition of clever rules for choosing the initial value of ρ_{max}.

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, and C. Schmid, *A reduced Hessian method for large scale constrained optimization*, SIAM J. Optim., 5 (1995), pp. 314–347.

[5] S. Bonettini, V. Ruggiero and F. Tinti, *On the solution of indefinite systems arising in nonlinear programming problems*, Numer. Lin. Alg. Appl., 14 (2007), pp. 807–831.

[6] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric indefinite linear systems*, Math. Comput., 31 (1977), pp. 163.179.

[7] R. H. Byrd, *An example of irregular convergence in some contrained optimization methods that use the projected hessian*, Math. Programming, 32 (1985), pp. 232–237.

[8] R. H. Byrd, *On the convergence of constrained optimization method with accurate hessian information on a subspace*, SIAM J. Num. Anal., 27 (1990), pp. 141–153.

[9] R. H. Byrd, J. C. Gilbert and J. Nocedal, *A trust region method based on interior point techniques for nonlinear programming*, Math. Programming, 89 (2000), pp. 149–186.

[10] R. H. Byrd, M. E. Hribar and J. Nocedal, *An interior point algorithm for large scale nonlinear programming*, SIAM J. Optim., 9 (2000), pp. 877–900.

[11] T. Coleman and A. R. Conn, *Nonlinear programming via an exact penalty function: asymptotic analysis*, Math. Programming, 24 (1982), pp. 123–136.

[12] T. Coleman and A. R. Conn, *On the local convergence of a quasi-Newton method for the nonlinear programming problem*, SIAM J. Num. Anal., 21 (1984), pp. 755–669.

[13] T. Coleman and A. Liao, *The local convergence of Byrd-Schnabel algorithm for contrained optimization*, SIAM J. Num. Anal., 21 (1984), pp. 755–669.

[14] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Trust-region methods*. SIAM, Philadelphia, USA, 2000.

[15] J. E. Dennis, H. J. Martinez and R. A. Tapia, *Convergence theory for the structured BFGS secant method with application to nonlinear least squares*, J. Optim. Theory Applics., 61 (1989), pp. 161-178.

[16] J. E. Dennis and L. N. Vicente *On the convergence theory of trust-region-based algorithms for equality-constrained optimization*, SIAM J. Optim., 7 (1997), pp. 927-950.

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

[18] M. El-Alem, *A global convergence theory for a general class of trust-region-based algorithms for constrained optimization without assuming regularity*, SIAM J. Optim., 9 (1999), pp. 965–990.

[19] R. Fletcher, *Practical methods of optimization*, 2nd. ed., John Wiley & Sons, Chichester, UK, 1987.

[20] 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.

[21] 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.

[22] 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.

[23] F. A. M. Gomes, M. C. Maciel and J. M. Martínez, *Nonlinear programming algorithms using trust regions and augmented Lagrangians with nonmonotone penalty parameters*, Math. Programming, 84 (1999), pp. 161–200.

[24] 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.

[25] 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 Verlag, New York, NY, 1997, pp. 246–255.

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

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

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

[29] J. M. Martínez and E. A. Pilotta, *Inexact restoration methods for nonlinear programming: advances and perspectives*, in Optimization and Control with Applications, L. Qi, K. Teo and X. Yang, eds., Springer Verlag, New York, NY, 2005, pp. 271–292.

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

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

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

[33] 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.

[34] 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.

[35] M. J. D. Powell, *The convergence of variable metric methods for nonlinearly contrained optimization calculations*, in Nonlinear Programming 3, O. L. Mangasarian, R. R. Meyer and S. M. Robinson, eds., Academic Press, New York, NY, 1978, pp. 27–63.

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

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

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

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

[40] O. Schenk and K. Gärtner, *On fast factorization pivoting methods for symmetric indefinite systems*, Electron. Trans. Numer. Anal., 23 (2006), pp. 158–179.

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

[42] R. A. Tapia, *Quasi-Newton methods for equality contrained optimization: equivalence of existing methods and a new implementation*, in Nonlinear Programming 3, O. L. Mangasarian, R. R. Meyer and S. M. Robinson, eds., Academic Press, New York, NY, 1978, pp. 125–164.

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

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

[45] 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.

[46] Y. Yuan, *An only 2-step q-superlinear convergence example for some algorithms that use reduced hessian approximations*, Math. Programming, 32 (1985), pp. 224-231.