Optimization Methods and Software
Vol. 21, No. 2, April 2006, 233–245

Taylor & Francis
Taylor & Francis Group

Matrix-free algorithm for the large-scale constrained trust-region subproblem†

ANTHONY J. KEARSLEY*

Mathematical and Computational Sciences Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-8910, USA

(Received 25 May 2004; in final form 30 December 2004)

A new 'matrix-free' algorithm for the solution of linear inequality constrained, large-scale trust-region subproblems is presented. The matrix-free nature of the algorithm eliminates the need for any matrix factorizations and only requires matrix–vector products. Numerical results that demonstrate the viability of the approach are included.

Keywords: Regularization; Constrained quadratic optimization; Trust region; Lanczos method

AMS Classification: Primary: 65F15; Secondary: 65G05

1. Introduction

The trust-region subproblem finds a point that minimizes a quadratic function subject to remaining inside an ellipsoidal region and is an important problem in optimization and linear algebra. Frequently, an application-motivated trust-region subproblem will require that the solution satisfies an additional collection of linear constraints. In this instance, the problem can be written as follows:

$$\begin{aligned} \min_x \quad & c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Qx \\ \text{s.t.} \quad & Ax \leq b \\ & \|Dx\| \leq \Delta, \end{aligned} \tag{1}$$

where $c, x \in \Re^n$, $b \in \Re^m$, $A \in \Re^{m\times n}$, $D, Q \in \Re^{n\times n}$ and Δ is a positive scalar. The aforementioned norm $\|\cdot\|$ is an Euclidean norm. The matrix D is usually selected to scale the problem appropriately and therefore is frequently a diagonal matrix. For simplicity and without loss of generality, we will assume that a linear transformation has been employed so that the

*Corresponding author. Email: ajk@cam.nist.gov

†Contribution of the National Institute of Standards and Technology and not subject to copyright in the United States.

Optimization Methods and Software
ISSN 1055-6788 print/ISSN 1029-4937 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/10556780500094754

matrix D is the identity matrix ($D = I$). There is no specific restriction on n and m, but the method is intended for problems where one or both of these makes the problem large-scale.

Problems of the form (1) arise in a myriad of scientific and engineering fields, for example, regularizing problems in control and optimal control can be realized as trust-region problems [1]. Regularization of many ill-posed problems can also be recognized using the same mathematical tools [see, e.g. the work of Symes 2, 3]. Our primary interest is in the robust generation of steps to be employed, in turn, to locate the minimum of a non-linear function subject to general non-linear constraints. In this case, both the objective function and the linear constraints will be lower order approximation of non-linear functions. The trust-region constraint will then act both as a regularization term and as a mechanism to prevent extremely large steps from being generated when far from the solution [see the survey text by Gould *et al.* 4].

Motivated by recent, successful work on efficient algorithms for the solution of (1) without the linear inequality constraints [see refs. 5–8] and with bound constraints on the variables [see ref. 9], we are hopeful that an effective algorithm can be constructed that does not explicitly use factorizations (approximate or exact) of Q and/or A. These so-called matrix free schemes are very memory cost-effective and can be employed to solve much larger class of problems. Indeed, in many non-linear programming problems born from discretizations of control problems, forming of A^{T} or constructing the product matrix AA^{T} is simply not possible.

The organization of the paper is as follows. In section 2, we describe the algorithm using a top–down approach, first discussing the structure of the major iterations, then discussing the minor iterations. Analysis is provided to show global convergence of the minor iterations. Next, in section 3, we discuss some details of the algorithm which make it implementable and efficient in practice. Some numerical results are presented and discussed in section 4, and we offer some concluding remarks in section 5. Throughout the paper, we will use the notation $M > 0$ to indicate that the symmetric matrix M is positive definite.

2. Basic algorithm

For notational simplicity, we assume throughout the rest of the paper that $D = I$. The general case, i.e. $D > 0$, is handled by a simple change of variables, as was discussed briefly in section 1. Hence, the algorithm presented here tackles the spherically constrained quadratic program

$$\begin{aligned} \min_{x} \quad & c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Qx \\ \text{s.t.} \quad & Ax \leq b, \\ & x^{\mathrm{T}}x \leq \Delta^2. \end{aligned} \tag{SCQP}$$

Let $\mathcal{X}$ denote the feasible set, i.e.

$$\mathcal{X} \triangleq \{x \in \Re^n | Ax \leq b, x^{\mathrm{T}}x \leq \Delta^2\}.$$

The following assumptions will be in force throughout.

ASSUMPTION 1 *The objective function in* (SCQP) *is strictly convex, in particular,* $Q = Q^{\mathrm{T}} > 0$.

ASSUMPTION 2 *The feasible set* $\mathcal{X}$ *for* (SCQP) *has a non-empty interior*, i.e. *there exists* $\hat{x} \in \Re^n$ *such that*

$$A\hat{x} < b, \quad \textit{and} \quad \hat{x}^{\mathrm{T}}\hat{x} < \Delta^2.$$

It is not difficult to show that these assumptions guarantee that a solution to (SCQP) is well defined and unique.

2.1 *Major iterations*

The main idea of our approach is to use the classical logarithmic barrier function [see, e.g. ref. 10] to eliminate the linear inequality constraints while leaving the spherical constraint as is. That is, we repeatedly 'solve' the barrier problem

$$\begin{aligned} \min_x \quad & c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Qx - \tau \sum_{j=1}^{m} \ln(b_j - a_j^{\mathrm{T}}x) \\ \text{s.t.} \quad & x^{\mathrm{T}}x \leq \Delta^2, \end{aligned} \tag{B_τ}$$

where a_j^{T} is the jth row of the matrix A, while simultaneously reducing the barrier parameter $\tau > 0$. Define the feasible set for (B_τ) as

$$\mathcal{S} \triangleq \{x \in \Re^n | x^{\mathrm{T}}x \leq \Delta^2\}.$$

Note that contrary to standard interior point barrier algorithms [11] as applied to problem (SCQP), it is entirely possible that a solution of the subproblem (B_τ) could lie on the surface of $\mathcal{S}$. In view of this fact, denote the set of feasible points in which our 'interior' point iterates may lie by

$$\mathcal{X}^0 \triangleq \{x \in \Re^n | Ax < b, \quad x^{\mathrm{T}}x \leq \Delta^2\}.$$

Under our barrier formulation, the central path for the problem (SCQP) is defined via the parameterization

$$x(\tau) \triangleq \arg\min_{x \in \mathcal{S}} F_\tau(x),$$

for $\tau > 0$, where

$$F_\tau(x) \triangleq \begin{cases} c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Qx - \tau \sum_{j=1}^{m} \ln(b_j - a_j^{\mathrm{T}}x), & x \in \mathcal{X}^0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Finally, a standard result from the theory of barrier algorithms [see, e.g. refs. 11,12] guarantees that the unique global minimizer x^* of (SCQP) is the limit of the central path, i.e.

$$x^* = \lim_{\tau \searrow 0} x(\tau).$$

Thus, a conceptual long-step central path following algorithm may be stated as follows.

ALGORITHM CP

Data c, Q, A, b, Δ, $x^0 \in \mathcal{X}^0$, $\tau^0 > 0$.
Parameter $M > 1$.
Step 0 $k \leftarrow 0$.
Step 1 $x^{k+1} \leftarrow \arg\min_{x \in \mathcal{S}} F_{\tau^k}(x)$.
Step 2 $\tau_{k+1} \leftarrow \tau_k / M$.
Step 3 $k \leftarrow k + 1$. *Go back to* Step 1.

Of course, such an algorithm is not implementable in practice. In particular, the exact minimization required by Step 1 cannot be completed in finite time. Fortunately, it is well known [11,12] that it is not necessary to solve the minimization in Step 1 exactly in order to guarantee convergence to x^*. Subsequently, we describe an iteration scheme that may be used to generate an approximate solution of (B_τ). In section 3, we discuss proximity measures which will allow us to decide when the approximate minimizer of (B_{τ^k}) is 'close enough' to the central path. This will allow us to construct an implementable path following an algorithm based on the model provided by Algorithm CP.

2.2 *Minor iterations*

We now focus our attention on the solution of the barrier subproblem (B_τ) for $\tau > 0$ fixed. To construct a 'matrix-free' algorithm, a QP-based Newton iteration is avoided. Instead, we treat the spherical constraint directly. Such an approach allows us to iteratively apply the Lanczos-based algorithms of ref. [5] or ref. [6], which are 'matrix free' and efficient for very large problems. Specifically, for fixed k and $\tau_k > 0$, in this section we describe an algorithm which generates a sequence $\{x^{k,j}\}_{j\in\aleph}$ satisfying

$$x^{k,j} \longrightarrow x(\tau_k), \quad \text{as } j \to \infty.$$

Consider the following second-order model of (B_{τ^k}) at the point $x^{k,j}$,

$$\begin{aligned} \min_x \quad & \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}}(x - x^{k,j}) + \frac{1}{2}(x - x^{k,j})^{\mathrm{T}}\nabla^2 F_{\tau^k}(x^{k,j})(x - x^{k,j}) \\ \text{s.t.} \quad & x^{\mathrm{T}}x \le \Delta^2. \end{aligned} \qquad (\mathrm{TRS}(x^{k,j}, \tau^k))$$

The minimization of a quadratic function subject to a spherical constraint is typically referred to as a trust-region subproblem (TRS). Note that, under our assumptions, the solution to this problem is always well defined and unique. Let $\hat{x}^{k,j+1}$ be the solution of $(\mathrm{TRS}(x^{k,j}, \tau^k))$ and define the search direction $\delta^{k,j} = \delta(x^{k,j}, \tau^k)$ as

$$\delta(x^{k,j}, \tau^k) \triangleq \hat{x}^{k,j+1} - x^{k,j}.$$

A new iterate $x^{k,j+1}$ may be constructed by performing a line search from $x^{k,j}$ along $\delta^{k,j}$, that is,

$$x^{k,j+1} = x^{k,j} + t^{k,j} \cdot \delta^{k,j},$$

where $t \in (0, 1]$ is chosen so that the new iterate satisfies a descent condition on a suitable merit function. Note that, as long as $x^{k,0} \in \mathcal{S}$, convexity of $\mathcal{S}$ and the direct incorporation of the spherical constraint in $(\mathrm{TRS}(x^{k,j}, \tau^k))$ guarantees that $x^{k,j} \in \mathcal{S}$ for all j, regardless of how $t^{k,j} \in (0, 1]$ is chosen. Thus, we need not incorporate the constraint into the merit function. We now state a conceptual algorithm for the solution of (B_τ).

ALGORITHM BP

Data $x^{k,0} \in \mathcal{X}^0$, $\tau^k > 0$.
Parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$.
Step 0 $j \leftarrow 0$.
Step 1 *(Computation of a search direction) Form* $\delta^{k,j}$ *from the solution of* $\mathrm{TRS}(x^{k,j}, \tau^k)$.
If $\delta^{k,j} = 0$, *stop.*

Step 2 *(Line Search) Compute* $t^{k,j}$*, the first number* t *in the sequence* $\{1, \beta, \beta^2, \ldots\}$ *satisfying*

$$F_{\tau^k}(x^{k,j} + t \cdot \delta^{k,j}) \leq F_{\tau^k}(x^{k,j}) + \alpha \cdot t \cdot \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j}.$$

Step 3 (*Updates*)
(i) $x^{k,j+1} \leftarrow x^{k,j} + t^{k,j}\delta^{k,j}$.
(ii) $j \leftarrow j + 1$. Go back to Step 1.

It remains to show that Algorithm BP does indeed generate a sequence of iterates converging to the central path. A point $x \in \Re^n$ is said to be a Karush Kuhn Tucker (KKT) point for (B_{τ^k}) if there exists a multiplier $\mu \in \Re$ satisfying

$$\begin{cases} Qx + c + \tau^k \cdot \sum_{j=1} \dfrac{1}{b_j - a_j^{\mathrm{T}} x} \cdot a_j + 2\mu x = 0, \\ x^{\mathrm{T}} x \leq \Delta^2, \\ \mu \cdot \left(x^{\mathrm{T}} x - \Delta^2\right), \quad \mu \geq 0. \end{cases} \tag{2}$$

In view of Assumption 1, it is not difficult to show that, for $\tau^k > 0$, (B_{τ^k}) is convex with a compact (convex) feasible set [13]. Thus, there exists a unique global minimizer $x^{k,*} = x(\tau_k)$ which, by definition, lies on the central path of (SCQP). Furthermore, (2) is necessary and sufficient, i.e. $x^* = x(\tau^k)$ if, and only if, there exists $\mu^* \in \Re$ such that (x^*, μ^*) satisfies (2).

Throughout the following analysis, we will refer to the KKT conditions for (TRS$(x^{k,j}, \tau^k)$) as well [see also refs. 14,15]. A point $\hat{x} \in \Re^n$ is a KKT point for (TRS$(x^{k,j}, \tau^k)$) if there exists $\hat{\mu} \in \Re$ such that

$$\begin{cases} \nabla^2 F_{\tau^k}(x^{k,j})^{\mathrm{T}}(\hat{x} - x^{k,j}) + \nabla F_{\tau^k}(x^{k,j}) + 2\hat{\mu}\hat{x} = 0, \\ \hat{x}^{\mathrm{T}}\hat{x} \leq \Delta^2, \\ \hat{\mu} \cdot \left(\hat{x}^{\mathrm{T}}\hat{x} - \Delta^2\right), \quad \hat{\mu} \geq 0, \end{cases} \tag{3}$$

Given any $x^{k,j} \in \mathcal{X}^0$ and $\tau^k > 0$, it can be shown that $\nabla^2 F_{\tau^k}(x^{k,j}) > 0$, hence (TRS$(x^{k,j}, \tau^k)$) has a unique global minimizer $\hat{x}^{k,j+1} = \hat{x}(x^{k,j}, \tau^k)$ which satisfies (3) for some $\hat{\mu}^{k,j} > 0$.

LEMMA 1 *The vector* $\delta(x, \tau^k) = 0$ *if, and only if,* x *is a* KKT *point for* (B_{τ^k}).

Proof Suppose $\delta(x, \tau^k) = 0$, then by definition $\hat{x}(x, \tau^k) = x$. Substituting this equality into (3), letting $\hat{\mu}$ be the KKT multiplier and expanding $\nabla F_{\tau^k}(x)$ shows that x satisfies (2) with multiplier $\hat{\mu}$. The converse is proved similarly. ■

LEMMA 2 *Given any* $x \in \mathcal{X}^0$ *and* $\tau^k > 0$, $x + \delta(x, \tau^k) \in \mathcal{S}$. *Further, if* x *is not a* KKT *point for* (B_{τ^k}), *then*

$$\nabla F_{\tau^k}(x)^{\mathrm{T}} \delta(x, \tau^k) < 0.$$

Proof That $x + \delta(x, \tau^k) \in \mathcal{S}$ is clear from the definition of $\delta(x, \tau^k)$ and (TRS(x, τ^k)). Note that $\hat{x} = x$ will always satisfy the constraint for (TRS(x, τ^k)), the optimal value must be non-positive. Thus,

$$\nabla F_{\tau^k}(x)^{\mathrm{T}} \delta(x, \tau^k) \leq -\frac{1}{2}\delta(x, \tau^k)^{\mathrm{T}} \nabla^2 F_{\tau^k}(x) \delta(x, \tau^k).$$

The result follows from Lemma 1 and the fact that $\nabla^2 F_{\tau^k}(x) > 0$. ■

LEMMA 3 *The line search is well defined*, i.e. *Step* 2 *yields a step-length* $t^{k,j} = \beta^i$ *for some finite* $i = i(k, j)$.

Proof Follows immediately from Lemma 2. ■

Now suppose that Algorithm BP generates a finite sequence $x^{k,j}$, $j = 1, \ldots, N$. Then, by Step 1, $\delta^{k,N} = 0$ and in view of Lemma 1, $x^{k,N}$ is a KKT point for (B_τ), which implies $x^{k,N} = x(\tau^k)$ by uniqueness. From this point forward, we make the assumption that $\delta^{k,j} \neq 0$, for all j, thus Algorithm BP generates an infinite sequence $\{x^{k,j}\}_{j\in\aleph}$. It will be shown that $x^{k,j} \to x(\tau^k)$, as $j \to \infty$. Note that Lemma 2 guarantee $x^{k,j} \in \mathcal{S}$, for all j, thus the sequence $\{x^{k,j}\}$ is bounded.

LEMMA 4 *Suppose* $\mathcal{J} \subseteq \aleph$ *is an infinite index set such that* $\delta^{k,j} \xrightarrow{j\in\mathcal{J}} 0$, *then* $x(\tau^k)$ *is the unique accumulation point for* $\{x^{k,j}\}_{j\in\mathcal{J}}$.

Proof Denote the multiplier satisfying (3) at the solution of (TRS$(x^{k,j}, \tau^k)$) by $\hat{\mu}^{k,j}$. Suppose that the sequence $\{\hat{\mu}^{k,j}\}_{j\in\mathcal{J}}$ is unbounded. In view of the first equation in (3), we must then have $x^{k,j} \xrightarrow{j\in\mathcal{J}} 0$. But for j sufficiently large, this would contradict the complementary slackness condition of (3), thus $\{\hat{\mu}^{k,j}\}_{j\in\mathcal{J}}$ is bounded. Now, let $\mathcal{J}' \subseteq \mathcal{J}$ be an infinite index set such that $x^{k,j} \xrightarrow{j\in\mathcal{J}'} x^{k,*}$, and suppose without loss of generality that $\hat{\mu}^{k,j} \xrightarrow{j\in\mathcal{J}'} \hat{\mu}^{k,*} \geq 0$. Taking limits on $\mathcal{J}'$ in (3) shows that $x^{k,*}$ is a KKT point for (B_{τ^k}) with multiplier $\hat{\mu}^{k,*}$. Uniqueness of such points proves the result. ■

LEMMA 5 *Suppose* $\mathcal{J} \subseteq \aleph$ *is an infinite index set such that* $x^{k,j} \xrightarrow{j\in\mathcal{J}} x^*$, *then* $\delta^{k,j} \xrightarrow{j\in\mathcal{J}} 0$.

Proof We begin by noting that, in view of Assumption 1 and convexity of the natural logarithm, there exists $\sigma > 0$ such that

$$\inf_{x\in\mathcal{X}^0, \|y\|=1} y^{\mathrm{T}} \nabla^2 F_{\tau^k}(x) y = \sigma.$$

Proceeding by contradiction, suppose there exists an infinite index set $\mathcal{J}' \subseteq \mathcal{J}$ and a constant $\underline{d} > 0$ such that $\|\delta^{k,j}\| \geq \underline{d}$ for all $j \in \mathcal{J}'$. Now, because $\hat{x} = x^{k,j}$ is always feasible for (TRS$(x^{k,j}, \tau^k)$), the optimal value is non-positive, thus

$$\begin{aligned} \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j} &\leq -\frac{1}{2}(\delta^{k,j})^{\mathrm{T}} \nabla^2 F_{\tau^k}(x^{k,j}) \delta^{k,j} \\ &\leq -\frac{\sigma}{2}\underline{d}^2 < 0, \end{aligned}$$

for all $j \in \mathcal{J}'$. Using the identity

$$F_{\tau^k}(x^{k,j} + t\delta^{k,j}) - F_{\tau^k}(x^{k,j}) = t \int_0^1 \nabla F_{\tau^k}(x^{k,j} + t\xi\delta^{k,j})^{\mathrm{T}} \delta^{k,j} d\xi,$$

we can show

$$\begin{aligned} &F_{\tau^k}(x^{k,j} + t\delta^{k,j}) - F_{\tau^k}(x^{k,j}) - \alpha t \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j} \\ &\quad = t \cdot \int_0^1 \left(\nabla F_{\tau^k}(x^{k,j} + t\xi\delta^{k,j}) - \nabla F_{\tau^k}(x^{k,j})\right)^{\mathrm{T}} \delta^{k,j} d\xi + t(1-\alpha)\nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j} \\ &\quad \leq t \cdot \left\{ \sup_{\xi\in[0,1]} \|\nabla F_{\tau^k}(x^{k,j} + t\xi\delta^{k,j}) - \nabla F_{\tau^k}(x^{k,j})\| \cdot \|\delta^{k,j}\| - (1-\alpha)\frac{\sigma}{2}\underline{d}^2 \right\} \end{aligned}$$

for all $j \in \mathcal{J}'$, j large enough. As $F_{\tau^k}(\cdot)$ is continuously differentiable in $\mathcal{X}^0$, and because $\delta^{k,j}$ is bounded, there exists $\underline{t} > 0$ such that for all $j \in \mathcal{J}'$, j sufficiently large,

$$F_{\tau^k}(x^{k,j} + t\delta^{k,j}) - F_{\tau^k}(x^{k,j}) - \alpha t \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j} \leq 0,$$

for all $t \in [0, \underline{t}\,]$. Therefore,

$$\begin{aligned} F_{\tau^k}(x^{k,j+1}) &\leq F_{\tau^k}(x^{k,j}) + \alpha t \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j} \\ &\leq F_{\tau^k}(x^{k,j}) - \alpha \underline{t} \left(\tfrac{\sigma}{2} \underline{d}^2\right), \end{aligned}$$

for all $j \in \mathcal{J}'$, j sufficiently large. Since Lemma 2 and Step 2 of Algorithm BP guarantee that $F_{\tau^k}(x^{k,j})$ is non-increasing in j, this implies $F_{\tau^k}(x^{k,j}) \to -\infty$, as $j \to \infty$. Clearly, this is a contradiction, as $x^{k,j} \in \mathcal{X}^0$ for all j and $F_{\tau^k}(\cdot)$ is continuous in the bounded set $\mathcal{X}^0$. ■

THEOREM 1 *The sequence $x^{k,j} \to x(\tau^k)$, as $j \to \infty$.*

Proof It follows immediately from Lemmas 4 and 5 that there exists an infinite index set $\mathcal{J} \subseteq \aleph$ such that $x^{k,j} \xrightarrow{j \in \mathcal{J}} x(\tau^k)$. Further, it is a simple consequence of the descent properties of the algorithm and the strict convexity of $F_{\tau^k}(\cdot)$ that $x(\tau^k)$ is the only accumulation point of $\{x^{k,j}\}_{j \in \aleph}$. ■

3. Algorithm specifics

In this section, we discuss some modifications of the algorithm outlined in the previous section which allow it to be implementable in practice.

3.1 *The line search*

Before the line search in Step 2 of Algorithm BP can be carried out, in practice, we will need to compute an upper bound on the step-length so that all linear constraints from (SCQP) will be strictly satisfied. If this is not done, numerical problems (domain errors) are likely to result when evaluating the natural logarithms of the (negative) constraint violations. Of course, an upper bound on the step size is easily computed analytically. Consider

$$\bar{t}^{k,j} \triangleq \begin{cases} 1, & \text{if } A\delta^{k,j} \leq 0, \\ \nu \cdot \min\left\{1, \dfrac{b_i - a_i^{\mathrm{T}} x^{k,j}}{a_i^{\mathrm{T}} \delta^{k,j}} : i \text{ s.t. } a_i^{\mathrm{T}} \delta^{k,j} > 0\right\}, & \text{otherwise.} \end{cases}$$

For $\nu = 1$, such a bound guarantees that $x^{k,j} + t\delta^{k,j} \in \mathcal{X}'$, for all $t \in [0, \bar{t}^{k,j}]$, but we require the trial points strictly satisfy the linear constraints. Thus, we must choose $\nu \in (0, 1)$ in order to guarantee $x^{k,j} + t\delta^{k,j} \in \mathcal{X}^0$, for all $t \in [0, \bar{t}^{k,j}]$. Note that typically ν is chosen very close to 1 (e.g. $\nu = 0.95$).

Once an upper bound has been established, we may proceed with the Armijo-type line search as in Step 2 of Algorithm BP, the only difference now being that $t^{k,j}$ is chosen as the first number t in the sequence $\{\bar{t}^{k,j}, \beta\bar{t}^{k,j}, \beta^2\bar{t}^{k,j}, \ldots\}$ satisfying

$$F_{\tau^k}(x^{k,j} + t \cdot \delta^{k,j}) \leq F_{\tau^k}(x^{k,j}) + \alpha \cdot t \cdot \nabla F_{\tau^k}(x^{k,j})^{\mathrm{T}} \delta^{k,j}.$$

3.2 *Proximity measures and barrier parameter updates*

In practice, the minor iterations must be stopped after a finite number of steps, hence the central path is never actually reached. Of course, it is well known that it is not only unnecessary but also computationally wasteful to expend too much effort in the minor iterations trying to get as close as possible to the central path. In general, with a well-chosen proximity measure, only a few minor iterations are required for each major iteration in order to ensure global convergence.

Proximity measures $p(\cdot,\cdot) : \mathcal{X}^0 \times \Re_+ \to \Re_+$ take many forms, but typically they satisfy

- $p(x,\tau) \geq 0, \forall x \in \mathcal{X}^0, \forall \tau \geq 0.$
- $p(x,\tau) = 0$ if, and only if, $x = x(\tau)$.

For our algorithm, we chose

$$p(x,\tau) \stackrel{\Delta}{=} \|\delta(x,\tau)\|,$$

i.e. the Euclidean norm of the search direction. In view of Lemma 1, such a choice satisfies the properties mentioned earlier. Of course, equally important as the exact form of the proximity measure itself is the choice of threshold used to determine when we are close enough to the central path for a given major iteration. We chose to reduce the threshold with the barrier parameter to ensure that we get closer and closer to the central path as the major iterations increase. In particular, we stop the minor iterations when

$$\left\|\delta(x^{k,j},\tau^k)\right\| \leq C \cdot \tau^k,$$

where $C > 0$ is a constant. Note that, to be complete, we should demonstrate that using such a proximity measure does not affect global convergence of the major iterations to the solution of (SCQP). Although we do not include such analysis here, we have outlined the arguments and believe the result to hold. A detailed examination of this issue remains to be done.

Many modern interior point algorithms base the barrier parameter on approximate complementarity and compute τ^k as a function of the dual and slack variables. As our algorithm does not produce a dual variable estimate, we resort to simpler schemes such as those used in classical barrier algorithms. Specifically, we use the long-step update and let

$$\tau^{k+1} \longleftarrow \frac{\tau^k}{M},$$

where $M > 1$. Of course, using such an update may limit the ultimate rate of convergence in a neighborhood of the solution.

3.3 *TRS solvers and the 'hard case'*

In this section, we show that the subproblems (TRS$(x^{k,j}, \tau^k)$) generated by the minor iterations of our algorithm tend towards the so-called 'hard case' for problems of this type [for detailed discussions of the hard case; see, e.g. refs. 5,9,15,16]. To simplify the discussion, consider the TRS

$$\begin{aligned} \min_x \quad & g^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx \\ \text{s.t.} \quad & x^{\mathrm{T}}x \leq s^2, \end{aligned} \tag{TRS}$$

where $g \in \Re^n$ and $0 \prec H = H^{\mathrm{T}} \in \Re^{n\times n}$. Let $\lambda^1 \in \Re$ be the smallest eigenvalue of H and let $\Lambda_1 \subseteq \Re^n$ be the corresponding eigenspace. For our purposes, it suffices[†] to say that (TRS) is in the hard case when $g \in \Lambda_1^{\perp}$. Note that even when this condition is close to being satisfied, numerical difficulties are likely to result.

In the case of (TRS($x^{k,j}$, τ^k)), we have

$$H = Q + \tau^k \cdot \sum_{i=1}^{m} \frac{1}{(a_i^{\mathrm{T}} x^{k,j} - b_i)^2} \cdot a_i a_i^{\mathrm{T}}$$

$$g = c - \tau^k \cdot \sum_{i=1}^{m} \frac{2a_i^{\mathrm{T}} x^{k,j} - b_i}{(a_i^{\mathrm{T}} x^{k,j} - b_i)^2} \cdot a_i.$$

Sacrificing rigor, we now discuss how TRS($x^{k,j}$, τ^k) may tend towards the hard case as $x^{k,j}$ tends toward a solution. For k fixed, let $\mathcal{I}_k \subseteq \{1, \ldots, m\}$ be the set of indices of linear constraints whose boundaries are approached by $x^{k,j}$ as $j \to \infty$. Of course, the barrier term prevents all linear constraints from being active, even asymptotically in j, but as k gets large, some will get close. Now suppose that k is 'large enough' (and fixed), and consider TRS($x^{k,j}$, τ^k) as j gets large. In particular, note that g will tend towards

$$\bar{g}_k \stackrel{\Delta}{=} \sum_{i\in\mathcal{I}_k} v_i a_i,$$

for some $v_i \in \Re$, $i \in \mathcal{I}_k$, and H will tend towards

$$\bar{H}_k \stackrel{\Delta}{=} \sum_{i\in\mathcal{I}_k} w_i a_i a_i^{\mathrm{T}} \geq 0,$$

for some $0 < w_i \in \Re$, $i \in \mathcal{I}_k$. The eigenspace of $\bar{H}_k$ corresponding to the non-zero (hence, largest) eigenvalues is $\mathrm{sp}\{a_i : i \in \mathcal{I}_k\}$. Because $\bar{H}_k = \bar{H}_k^{\mathrm{T}}$, $\mathcal{S}_1 = \mathrm{sp}\{a_i : i \in \mathcal{I}_k\}^{\perp}$, where $\mathcal{S}_1$ is the eigenspace corresponding to the smallest eigenvalue, i.e. zero. Thus,

$$\bar{g}_k \in \mathcal{S}_1^{\perp},$$

which is one of the conditions of the hard case. Although this discussion dealt primarily with asymptotic properties, it is clear that for k and j large enough, (TRS($x^{k,j}$, τ^k)) will be 'close' to the hard case, which could cause problems numerically.

Recently, there has been a great deal of effort towards developing algorithms which more effectively deal with the hard case. Two promising approaches are due to Rojas *et al.* [5] and Rendl and Wolkowicz [6]. Although other work has been done, what makes these algorithms attractive in our context is the fact that they are 'matrix free', i.e. do not rely on matrix factorizations, and are well-suited for large-scale problems.

4. Numerical results

Numerical experiments that demonstrate the efficacy of this approach are presented in this section. The algorithm has been implemented in MATLAB 6.0. Eigenvalue computations required by the algorithm were performed using an implementation of the implicitly restarted

[†]We will assume in the following discussion that the other condition for the hard case holds, namely, the solution y of $(H - \lambda_1 I)y = -g$ satisfies $\|y\| < s$.

Arnoldi method [17] available in ARPACK [18]. All numerical experiments presented here were run on a Pentium 1.80 (GHz) personal computer with 512 MB of RAM running a Linux operating system. Double precision IEEE floating point arithmetic with machine precision approximately 2.2×10^{-16} was employed.

To demonstrate the efficacy of the algorithm, use the algorithm to solve two sets of problems. In the first set of problems, the objective function matrix Q was chosen to be different powers of the standard discrete, five-point, equally spaced stencil approximation to the two-dimensional Laplacian operator on a unit square, say L, where $Q = \{L, L^2, L^3\}$. The matrix A was selected to be a uniformly distributed sparse (pseudo) random matrix with elements chosen from $[-10, 10]$ and similarly the vectors c and b were selected to be a uniformly distributed (pseudo) random vectors whose elements were chosen from $[-1, 1]$ (using intrinsic MATLAB commands). For the second set of problems, we report on the algorithm's performance on an application problem from computational material science described at the end of this section.

In some cases, the generated linear constraints did not admit a feasible point, in which case the vector b was increased until the feasible region was large enough to posses an non-empty interior. Finally, all numerical results report only on starting points that were feasible

Table 1. Algorithm performance on test problems.

n	m	Q	Δ	Sparse(A)(%)	ITS(its)	Hard case	Fails	Mat∗vec
1,000	1,000	L	1.e − 1	10	102(13)	22	0/10	2,688
1,000	1,000	L	1.e + 1	10	100(13)	17	0/10	2,113
1,000	1,000	L	1.e − 1	15	108(12)	34	1/10	2,659
1,000	1,000	L	1.e + 1	15	119(14)	19	0/10	2,002
1,000	1,000	L	1.e − 1	20	141(18)	58	0/10	2,713
1,000	1,000	L	1.e + 1	20	141(17)	32	0/10	1,957
1,000	1,000	L^2	1.e − 1	10	129(21)	29	1/10	2,934
1,000	1,000	L^2	1.e + 1	10	121(17)	31	0/10	2,712
1,000	1,000	L^2	1.e − 1	15	145(28)	43	2/10	2,840
1,000	1,000	L^2	1.e + 1	15	147(19)	49	0/10	2,008
1,000	1,000	L^2	1.e − 1	20	178(35)	50	2/10	3,163
1,000	1,000	L^2	1.e + 1	20	199(39)	67	1/01	4,070
1,000	1,000	L^3	1.e − 1	10	289(39)	101	3/10	5,841
1,000	1,000	L^3	1.e + 1	10	291(38)	142	2/10	5,982
1,000	1,000	L^3	1.e − 1	15	316(43)	174	3/10	6,956
1,000	1,000	L^3	1.e + 1	15	320(48)	163	3/10	6,675
1,000	1,000	L^3	1.e − 1	20	352(42)	191	5/10	7,071
1,000	1,000	L^3	1.e + 1	20	393(44)	184	4/10	8,008
1,000	10,000	L	1.e − 1	10	132(15)	28	0/10	3,469
1,000	10,000	L	1.e + 1	10	98(15)	27	0/10	3,468
1,000	10,000	L	1.e − 1	15	189(19)	45	0/10	5,711
1,000	10,000	L	1.e + 1	15	135(18)	36	0/10	5,402
1,000	10,000	L	1.e − 1	20	197(23)	67	0/10	6,734
1,000	10,000	L	1.e + 1	20	158(24)	41	0/10	6,091
1,000	10,000	L^2	1.e − 1	10	129(21)	29	1/10	3,453
1,000	10,000	L^2	1.e + 1	10	121(17)	31	0/10	2,999
1,000	10,000	L^2	1.e − 1	15	191(28)	61	3/10	5,827
1,000	10,000	L^2	1.e + 1	15	152(30)	57	2/10	5,715
1,000	10,000	L^2	1.e − 1	20	211(23)	84	4/10	7,133
1,000	10,000	L^2	1.e + 1	20	209(29)	82	4/10	7,002
1,000	10,000	L^3	1.e − 1	10	313(59)	201	5/10	9,988
1,000	10,000	L^3	1.e + 1	10	310(58)	195	5/10	9,972
1,000	10,000	L^3	1.e − 1	15	396(43)	222	8/10	14,410
1,000	10,000	L^3	1.e + 1	15	427(48)	176	3/10	15,812
1,000	10,000	L^3	1.e − 1	20	453(41)	209	9/10	16,899
1,000	10,000	L^3	1.e + 1	20	471(49)	200	8/10	17,001

with respect to the linear constraints, a so-called Big-M method was employed to attain these feasible starting points. This method calculation was inexpensive compared with the cost of solving the problem. The major iteration was halted when the relative change in iterates was below 1.e − 5. The trust-region problems were solved using large-scale trust-region subproblem (LSTRS) [19] and the parameters used in our numerical tests were $\tau_0 = 1$, $M = 2$ and $C = 10^{-9}$.

In table 1, we summarize the results of our algorithm applied to 10 different problems with varying sizes, condition number of matrices and trust-region radii. The selection of trust-region radii was made to ensure that the numerical results contained examples were the trust-region constraint was both active and inactive. The first two columns denote the number of variables and constraints. The matrices Q and A are described in the third and sixth columns. In the fifth column, we report on the average number of major iterations and, in parenthesis the average number of inner iterations when the algorithm converged (rounded to the nearest unit). In the seventh and eighth columns, the average number of hard cases encountered and failures. The ninth column records the average number of matrix–vector products performed when the algorithm succeeded in finding the optimal solution. All matrix–vector products were calculated without storing the matrix.

4.1 *An application problem*

In table 2, we summarize the results of our algorithm applied to an optimization problem from material science. This particular problem arises from a two-dimensional scalar model of Austenite-twined-Martensite interface proposed by Kohn and Müller [20,21]. They derive an model problem that includes an equipartitioning principle which enters the problem as a constraint relating elastic and surface energies. This constraint cannot be numerically evaluated because it involves a sup over all possible partitions of the physical domain, but it can sequentially approximated through a family of linear inequality constraints [see refs. 22,23].

The first three columns in table 2 denote the number of variables, constraints and the trust-region radius employed in the model problem. In the fourth column, we report on the average number of major iterations and in parenthesis the average number of inner iterations when the algorithm converged (rounded to the nearest unit). The fifth column records the number of hard cases encountered. Finally, the sixth and seventh columns report the number of failures and average number of matrix–vector products performed when the algorithm succeeded in finding the optimal solution.

Table 2. Algorithm performance on application problems.

n	m	Δ	ITS(its)	Hard case	Fails	Mat∗vec
4,096	1	1.e − 1	69(21)	18	0/10	2,993
4,096	1	1.e + 1	59(18)	0	0/10	1,192
4,096	1,000	1.e − 1	117(23)	10	1/10	5,080
4,096	1,000	1.e + 1	121(18)	9	0/10	4,178
4,096	1,000,000	1.e − 1	589(32)	106	5/10	10,981
4,096	1,000,000	1.e + 1	232(16)	29	0/10	9,347
16,384	1	1.e − 1	309(20)	102	2/10	8,524
16,384	1	1.e + 1	291(18)	16	1/10	6,754
16,384	1,000	1.e − 1	781(31)	112	5/10	9,134
16,384	1,000	1.e + 1	628(14)	24	4/10	7,086
16,384	1,000,000	1.e − 1	–	–	10/10	–
16,384	1,000,000	1.e + 1	790(23)	97	8/10	13,012

In both the first test problem and the material science model problem it appears that the algorithm encounters the hard case as iterates draw closer to the solution, especially for ill-conditioned problems. The ill conditioning that tends to increase the frequency with which the hard case was encountered can result from a larger condition number of the matrix Q or through an increase in the number of constraints, m. The analysis presented here supports these observations.

5. Conclusions

A new algorithm for the solution of large-scale, inequality constrained trust-region subproblems has been presented. The algorithm follows work on embedding the LSTRS [24] into parameterized eigenvalue problems [25] and appears to work well for problems of moderate condition number and may eventually be employed in tandem with preconditioners for ill-conditioned problems.

References

[1] Dennis, J.E., Heinkenschloss, M. and Vicente, L.N., 1998, Trust-region interior-point algorithms for a class of nonlinear programming problems. *SIAM Journal on Control and Optimization*, **36**, 1750–1794.
[2] Symes, W.W., 1990, Velocity inversion: a case study in infinite-dimensional optimization. *Mathematical Programming*, **48**, 71–102.
[3] Symes, W.W., 1993, A differential semblance criterion for inversion of multioffset seismic reflection data. *Journal of Geophysical Research*, **98**, 2062–2073.
[4] Gould, N.I.M., Conn, A.R. and Toint, Ph.L., 2000, *Trust-Region Methods* (Philadelphia, Pennsylvania: SIAM).
[5] Rojas, M., Santos, S.A. and Sorensen, D.C., 2000, A new matrix free algorithm for the large scale trust region subproblem. *SIAM Journal on Optimization*, **11**(3), 611–646.
[6] Rendl, F. and Wolkowicz, H., 1997, A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming*, **77**(2), 273–299.
[7] Golub, G.H. and von Matt, U., 1991, Quadratically constrained least squares and quadratic problems. *Numerical Mathematics*, **59**, 561–580.
[8] Gould, N.I.M., Lucidi, S., Roma, M. and Toint, Ph.L., 1999, Solving the trust-region subproblem using the Lanzos method. *SIAM Journal on Optimization*, **9**, 504–524.
[9] Rojas, M. and Steihaug, T., 2002, An interior-point trust-region-based method for large-scale non-negative regularization. *Inverse Problems*, **18**, 1291–1307.
[10] Fiacco, A.V. and McCormick, G.P., 1968, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* (New York: John Wiley and Sons; republished by Philadelphia: SIAM).
[11] Wright, S.J., 1997, *Primal-Dual Interior Point Methods* (Philadelphia: SIAM).
[12] Ye, Y., 1997, *Interior Point Algorithms: Theory and Analysis* (New York: John Wiley and Sons).
[13] Bertsekas, D.P., 1999, *Nonlinear Programming* (Belmont, Massachusetts: Athena Scientific).
[14] Gay, D.M., 1981, Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, **2**(2), 186–197.
[15] Sorensen, D.C., 1982, Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, **19**, 409–426.
[16] Moré, J.J. and Sorensen, D.C., 1983, Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, **4**, 553–572.
[17] Sorensen, D.C., 1992, Implicit application of polynomial filters in a K-step arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, **13**, 357–385.
[18] Lehoucq, R.B., Sorensen, D.C. and Yang, C., 1998, *ARPACK User's Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* (Philadelphia: SIAM).
[19] Rojas, M., Santos, S.A. and Sorensen, D.C., 2003, LSTRS: Matlab software for large-scale trust-region subproblems and regularization. Technical Report TR2003–04, Department of Mathematics, Wake Forest University.
[20] Kohn, R.V. and Müller, S., 1992, Branching of twins near an austenite-twinned-martensite interface. *Philosophical Magazine A*, **66**, 697–715.
[21] Kohn, R.V. and Müller, S., 1994, Surface energy and microstructure in coherent phase transitions. *Communications in Pure and Applied Mathematics*, **47**, 405–435.

[22] Kearsley, A.J. and Melara, L.A., 2003, Simulation of an austenite-twinned-martensite interface. *NIST Journal of Research*, **108**(6), 413–427.
[23] Kearsley, A.J. and Melara, L.A., 2004, Constrained simulation of an austenite-twinned-martensite interface. *NIST Journal of Research* (submitted for publication).
[24] Steihaug, T., 1983, The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, **20**, 626–637.
[25] Sorensen, D.C., 1997, Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM Journal on Optimization*, **7**, 141–161.