MAT

Serie A

Conferencias, seminarios y trabajos de Matemática

ISSN:1515-4904

14

Terceras Jornadas sobre Ecuaciones Diferenciales, Optimización y Análisis Numérico

María C. Maciel

Domingo A. Tarzia (Eds.)

Departamento de Matemática, Rosario, Argentina

Marzo 2007

UNIVERSIDAD AUSTRAL

FACULTAD DE CIENCIAS EMPRESARIALES

ISSN 1515-4904

MAT

SERIE A: CONFERENCIAS, SEMINARIOS Y TRABAJOS DE MATEMÁTICA

No. 14

TERCERAS JORNADAS SOBRE ECUACIONES DIFERENCIALES, OPTIMIZACIÓN Y ANÁLISIS NUMÉRICO

Maria Cristina Maciel - Domingo Alberto Tarzia (Eds.)

INDICE

Rosario, Marzo 2007

LARGE-SCALE ALGORITHMS FOR MINIMIZING A LINEAR FUNCTION WITH A STRICTLY CONVEX QUADRATIC CONSTRAINT[1]

T.I. GIBELLI [2] and M.C MACIEL [3]
Departamento de Matemática, Universidad Nacional del Sur
Av. Alem 1253, 8000 Bahía Blanca, Argentina.

Abstract

The problem of minimizing a linear function subject to convex quadratic constraints appears in many topological optimization problems such as truss and free material optimization. Usually this kind of problems is solved using interior point methods. In this work, the problem with only one constraint is considered. The solution of the problem is analyzed when the constraint matrix is symmetric and positive definite. A simple iterative algorithm is proposed. It is essentially based on the steepest descent direction and the geometric properties of the problem. In this method an iteration goes to the boundary in the negative gradient direction and then it finds the center of the region determined by the intersection of the level hyperplane and the feasible set. The convergence of this algorithm is analyzed. Also, an improved version of the algorithm is established. It comes from the observation that centers are in the same line, so only one center-finding operation is needed. Because of its simplicity, these algorithms are appropriate for very large scale problems. Numerical results are presented.

Key words: Quadratic Constraints, Topological Optimization.
AMS Subject Classification: 49M40, 49N10, 90C25, 90C30.

1 Introduction

Some topologic optimization problems, like truss and free material optimization, can be established as a constrained optimization problem as follows

$$\begin{array}{ll} \min\limits_{s.t.} & c^T x \\ & \frac{1}{2} x^T A_i x \leq b_i, \quad i = 1, ..., m, \end{array} \tag{1}$$

where $c \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{n \times n}$ are symmetric positive semidefinite matrices, and $b_i \in \mathbb{R}$ for all $i = 1, ..., m$ [3].

This kind of problems belongs to the more general class of convex optimization problems:

$$\begin{array}{ll} \min\limits_{s.t.} & f(x) \\ & g_i(x) \leq 0, \quad i = 1, ..., m, \end{array} \tag{2}$$

where $f, g_i : \mathbb{R}^n \to \mathbb{R}$ are twice continuously differentiable convex functions, the set of optimal solutions is a nonempty and compact set and there exists a point $\widehat{x}$ such as $g_i(\widehat{x}) < 0$ for all $i = 1, ..., m$.

Many numerical methods solve the problem under some additional assumptions. The interior point methods are perhaps the most popular [1, 4, 6, 10]. They are used to transform a

[1]This work has been partially supported by Universidad Nacional del Sur, Project 24/069 and Fundación Antorchas, Project 13900-4.
[2]tgibelli@uns.edu.ar. CONICET.
[3]immaciel@criba.edu.ar. Fax: 54-291-4595163.

constrained problem to a sequence of unconstrained problems, which are solved via well known algorithms, for instance the Newton method. Ben-Tal and Zibulesvsky [2] have developed an efficient method for large scale convex problems which are very common in topological design. The algorithm combines ideas of penalty and barrier methods with the Augmented Lagrangian method.

In this work, the problem (1) is analyzed with only one quadratic constraint, and the matrix is symmetric and positive definite. First, the existence of solution and its exact expression is studied. Then, according to the geometric characteristics of the problem an iterative algorithm is proposed to find the solution. The convergence of this algorithm is analyzed. Finally, an improved version of the algorithm is presented and some numerical results are shown to illustrate the behavior of the algorithms.

2 The problem

We consider the optimization problem

$$\begin{array}{ll} \min\limits_{s.t.} & f(x) = c^T x \\ & \dfrac{1}{2} x^T A x - d^T x \leq b, \end{array} \tag{3}$$

where $c \in \mathbb{R}^n$, $c \neq 0$, $d \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix and $b \in \mathbb{R}$, $b > 0$.

The problem (3) can be transformed into the following one

$$\begin{array}{ll} \min\limits_{s.t.} & c^T x \\ & \dfrac{1}{2} x^T A x \leq b, \end{array} \tag{4}$$

via a simple substitution. Such a substitution consists of translating the center of the quadratic $\frac{1}{2} x^T A x - d^T x = b$ to the origin.

The proposed algorithm is based on the geometric properties of the last problem. We state some of them.

Lemma 2.1 *The problem (4) has a unique solution x^* and it satisfies that Ax^* and c are parallel directions.*

Proof: It is straightforward. □

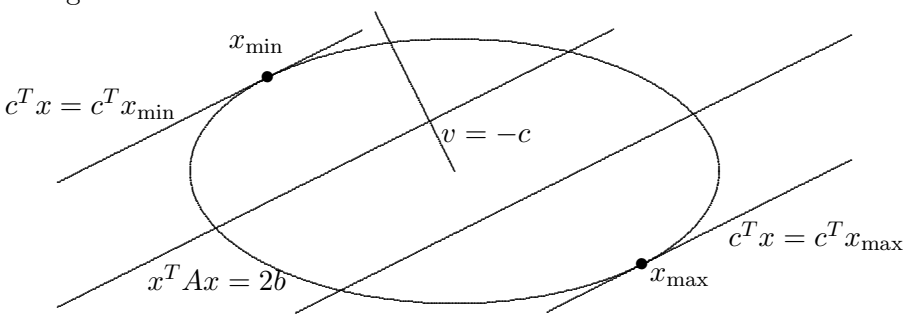

Figure 1: The geometric properties of the problem

Lemma 2.2 *The problem (4), where $c \in \mathbb{R}^n$, $c \neq 0$, $A = I_n$, and b is a positive real number, has a unique solution $x^* = -\sqrt{\dfrac{2b}{c^T c}}\, c$.*

Proof: It is straightforward. □

Theorem 2.1 *The problem (4) has a unique solution,* $x^* = -\sqrt{\dfrac{2b}{c^T A^{-1} c}}\, A^{-1} c$.

Proof: It follows from Lemma 2.1. □

3 The algorithm

The exact solution of problem (4) requires the computation of the inverse matrix of A. It is well known that this computation is not convenient because of the numerical errors and the arithmetic cost. Then, we propose an iterative method which does not require the inverse matrix of A and takes into account the geometric properties of the problem described above.

The main idea is to find alternatively an interior point and a border point following the steepest descent direction of the objective function until the minimizer is reached. The algorithm starts with an interior point. Without of generality, let us take $y_1 = 0$.

Figure 2 shows the sequence of interior and border points generated by this algorithm.

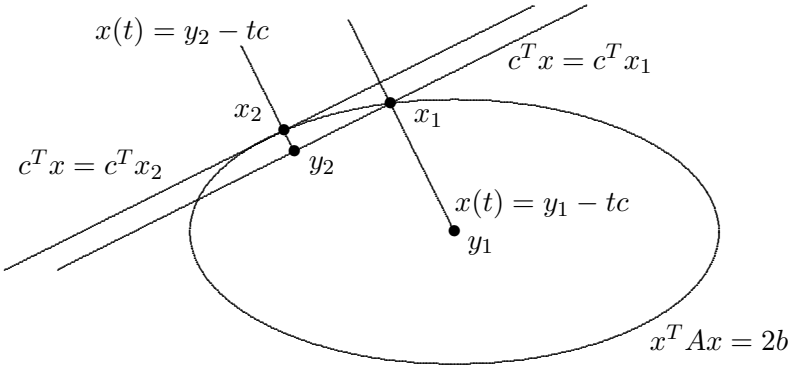

Figure 2: The behavior of the iterative method

Algorithm 1:
Given: $c \in \mathbb{R}^n,\ c \neq 0,\ A \in \mathbb{R}^{n\times n}$ symmetric and positive definite matrix and $b \in \mathbb{R},\ b > 0$.
For $k = 1, 2, \ldots$. "until convergence" repeat

Step 1:

- If $k = 1$, $y_1 = 0$.
- If $k > 1$, find an interior point y_k as the center of the resulting intersection between the level hyperplane $c^T x = c^T x_{k-1}$ and the constraint.

Step 2: Find a border point x_k as the intersection between the halfline passing through y_k in the direction $v = -c$ and the constraint.

Now let us describe in details Step 1 and 2.

- **Step 1:** $(k > 1)$ Since $c \neq 0$, there exists at least an index i such that $c_i \neq 0$. If $f_{k-1} = c^T x_{k-1}$, then from the hyperplane $c^T x = f_{k-1}$ it is possible to obtain the component

$$x_i = \frac{1}{c_i}\left(f_{k-1} - \sum_{j\neq i,\ j=1}^{n} c_j\, x_j\right).$$

Then,

$$\begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & \dots & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -\frac{c_1}{c_i} & \dots & -\frac{c_{i-1}}{c_i} & -\frac{c_{i+1}}{c_i} & \dots & -\frac{c_n}{c_i} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 1 & 0 \\ 0 & 0 & \dots & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} + \frac{f_{k-1}}{c_i} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Let

- $P \in \mathbb{R}^{n\times(n-1)}$ be the matrix obtained from the identity $I_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$ adding the zero vector as the i-th row . Clearly $PP^T = I_n$ and $P^TP = I_{n-1}$.
- $e = \dfrac{e_i}{c_i}$, where $e_i^T = (0, 0, ..., \underset{\text{place } i}{1}, ..., 0, 0) \in \mathbb{R}^n$.

So $x = B\widehat{x} + f_{k-1}e$ where $B = (I_n - ec^T)P \in \mathbb{R}^{n\times(n-1)}$ and $\widehat{x} = P^Tx \in \mathbb{R}^{n-1}$. The intersection between the hyperplane $x = B\widehat{x} + f_{k-1}e$ with the constraint $\frac{1}{2}x^TAx = b$ is

$$\frac{1}{2}(B\widehat{x} + f_{k-1}e)^TA(B\widehat{x} + f_{k-1}e) = b$$

$$\frac{1}{2}\widehat{x}^TB^TAB\widehat{x} + f_{k-1}e^TAB\widehat{x} + \frac{1}{2}f_{k-1}^2e^TAe = b$$

$$\frac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d}_k^{\,T}\widehat{x} = \widehat{b}_k,$$

where $\widehat{A} = B^TAB$, $\widehat{d_k} = -f_{k-1}B^TAe$ y $\widehat{b}_k = b - \frac{1}{2}f_{k-1}^2e^TAe$.

The center $\widehat{y}_k$ of $\frac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d}_k^{\,T}\widehat{x} = \widehat{b}_k$ can be obtained by solving the linear system $\widehat{A}x = \widehat{d_k}$ or equivalently by obtaining the minimizer of the quadratic function $\frac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d}_k^{\,T}\widehat{x}$. Clearly, that point can be obtained via any minimization algorithm.

Finally, $y_k = B\widehat{y}_k + f_{k-1}e$.

- **Step 2:** The point x_k is obtained as the intersection between the halfline $x(t) = y_k - tc$, $t > 0$, and the quadratic constraint $x^TAx = 2b$. Then,

$$(y_k - tc)^TA(y_k - tc) = 2b$$

$$y_k^TAy_k - 2tc^TAy_k + t^2c^TAc = 2b$$

$$(c^TAc)t^2 + 2(-c^TAy_k)t + (y_k^TAy_k - 2b) = 0$$

$$t_k = \frac{c^TAy_k + \sqrt{(c^TAy_k)^2 - (c^TAc)(y_k^TAy_k - 2b)}}{c^TAc}$$

$$x_k = y_k - t_kc.$$

- **Stopping criterion:** the termination rules to stop the algorithm are the following:
  - if the closeness between interior and border points is less or equal to a given tolerance $\dfrac{\|y_k - x_k\|}{\|x_k\|} \leq$ tolerance, or

– if the closeness between iterates, $\dfrac{\|x_{k-1} - x_k\|}{\|x_k\|} \leq$ tolerance.

After these considerations, **Algorithm 1** can be written as

Algorithm 1:

Given: $c \neq 0 \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ symmetric and positive definite and $b \in \mathbb{R}$, $b > 0$.

Compute: B, e, $\widehat{A} = B^T AB$ and $v = B^T Ae$.

For $k = 1, 2, ...$ "until convergence" repeat

Step 1:

- If $k = 1$, $y_1 = 0$.
- If $k > 1$,
  - compute: $f_{k-1} = c^T x_{k-1}$ and $\widehat{d_k} = -f_{k-1}v$.
  - $\widehat{y}_k = argmin\ \dfrac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d_k}^T\widehat{x}$.
  - $y_k = B\widehat{y}_k + f_{k-1}e$.

Step 2:

- $t_k = \dfrac{c^T Ay_k + \sqrt{(c^T Ay_k)^2 - (c^T Ac)(y_k^T Ay_k - 2b)}}{c^T Ac}$.
- $x_k = y_k - t_k c$.

4 Convergence of the algorithm

In this section the behavior of the sequences of interior and border points, $\{y_k\}$ and $\{x_k\}$, generated by the algorithm is analyzed. It will be show that both sequences converge to the solution of the problem (4).

Lemma 4.1 *Let $\{f_k\}$, with $f_k = c^T x_k$ and $\{y_k\}$, the sequences generated by Algorithm 1, and $x^* = -\sqrt{\dfrac{2b}{c^T A^{-1}c}}A^{-1}c$ the solution of problem (4), then*

i) $\{f_k\}$ is a negative decreasing sequence.

$$0 > f_1 > f_2 > ... > f_{k-1} > f_k > f_{k+1} > ...$$

ii) Each term y_k belongs to the halfline $x(t) = t(I_n - B\widehat{A}^{-1}B^T A)e, \quad t < 0$.

iii) $x^{*T}Ax^* = 2b$.

iv) x^* is located to the line $x(t) = t(I_n - B\widehat{A}^{-1}B^T A)e, \quad t < 0$.

Proof:

i) Since $x_1 = -\sqrt{\dfrac{2b}{c^T Ac}}\, c$. Then,

$$f_1 = c^T x_1 = -\sqrt{\frac{2b}{c^T Ac}}\, c^T c < 0.$$

On the other hand, for all k, y_k belongs to the hyperplane $c^T x = c^T x_{k-1}$. Then $c^T y_k = c^T x_{k-1}$. Since $x_k = y_k - t_k c$ with $t_k > 0$ we obtain,

$$f_k = c^T x_k = c^T y_k - t_k c^T c < c^T y_k = c^T x_{k-1} = f_{k-1}.$$

ii) For any iteration k, $\widehat{y}_k = argmin \;\; \frac{1}{2}\widehat{x}^T \widehat{A}\widehat{x} - \widehat{d_k}^T \widehat{x}$, where $\widehat{d_k} = -f_{k-1} B^T Ae$. Therefore, $\widehat{y}_k = \widehat{A}^{-1}\widehat{d_k} = -f_{k-1}\widehat{A}^{-1} B^T Ae$. Since $y_k = B\widehat{y}_k + f_{k-1}e$, it can be written

$$y_k = -f_{k-1} B\widehat{A}^{-1} B^T Ae + f_{k-1}e = f_{k-1}(I_n - B\widehat{A}^{-1} B^T A)e.$$

Therefore for all k, y_k lies on the halfline $x(t) = t(I_n - B\widehat{A}^{-1} B^T A)e$ with $t = f_{k-1} < 0$.

iii) It is straightforward, because

$$x^{*T} Ax^* = (-\sqrt{\frac{2b}{c^T A^{-1} c}} A^{-1} c)^T A(-\sqrt{\frac{2b}{c^T A^{-1} c}} A^{-1} c) = \frac{2b}{c^T A^{-1} c} c^T A^{-1} c = 2b.$$

iv) Let us consider the halfline $x(t) = t(I_n - B\widehat{A}^{-1} B^T A)e$ with $t < 0$ and $t^* = c^T x^*$. Then,

- $t^* < 0$, because

$$t^* = c^T x^* = c^T(-\sqrt{\frac{2b}{c^T A^{-1} c}} A^{-1} c) = -\sqrt{\frac{2b}{c^T A^{-1} c}} c^T A^{-1} c = -\sqrt{2b\, c^T A^{-1} c} < 0.$$

- $x^* = t^*(I_n - B\widehat{A}^{-1} B^T A)e$, because the facts $B = (I_n - ec^T)P$ and $e^T c = 1$ imply that $B^T Ax^* = 0$:

$$\begin{aligned} B^T Ax^* &= -\sqrt{\frac{2b}{c^T A^{-1} c}} B^T c = -\sqrt{\frac{2b}{c^T A^{-1} c}} P^T (I_n - ce^T)c \\ &= -\sqrt{\frac{2b}{c^T A^{-1} c}} P^T (c - c\, e^T c) = -\sqrt{\frac{2b}{c^T A^{-1} c}} P^T (c - c) \\ &= 0. \end{aligned}$$

Then $\widehat{A} P^T x^* = -B^T Aec^T x^*$, because

$$\begin{aligned} \widehat{A} P^T x^* &= B^T ABP^T x^* = B^T A(I_n - ec^T)PP^T x^* \\ &= B^T Ax^* - B^T Aec^T x^* \\ &= -B^T Aec^T x^*. \end{aligned}$$

Therefore, $\widehat{A}^{-1} B^T Aec^T x^* = -P^T x^*$, and

$$\begin{aligned} t^*(I_n - B\widehat{A}^{-1} B^T A)e &= (I_n - B\widehat{A}^{-1} B^T A)ec^T x^* = ec^T x^* - B(\widehat{A}^{-1} B^T Aec^T x^*) \\ &= ec^T x^* + (I_n - ec^T)PP^T x^* = ec^T x^* + x^* - ec^T x^* \\ &= x^*. \end{aligned}$$

□

We establish the convergence result.

Theorem 4.1 *The sequences $\{y_k\}$ and $\{x_k\}$ generated by Algorithm 1 converge to x^*.*

Proof: Since A is symmetric and positive definite, it is possible to consider the elliptic norm $\|z\|_A = \sqrt{z^T A z}$. In first place we show that the sequence $\{\|y_k\|_A\}$ is increasing and upper bounded.

- Since for all k, y_k is an interior point of the feasible region, $\|y_k\|_A^2 = y_k^T A y_k < 2b$. Therefore, the sequence $\{\|y_k\|_A\}$ is upper bounded by $\sqrt{2b}$.
- According to the second part of the Lemma 4.1, $\|y_k\|_A = |f_{k-1}|\|(I_n - B\widehat{A}^{-1}B^T A)e\|_A$ and from the part i) of Lemma 4.1, the sequence $\{|f_k|\}$ is monotone increasing, then the sequence $\{\|y_k\|_A\}$ also results to be monotone increasing.

Therefore, the sequence $\{\|y_k\|_A\}$ is convergent and from the fact that $\{y_k\}$ is contained in a halfline, the sequence $\{y_k\}$ is also convergent. For each k, let us consider the triangle determined by y_k, y_{k+1} and x_k, which is rectangle at x_k. Then

$$\|x_k - y_k\|_2 < \|y_{k+1} - y_k\|_2.$$

The convergence of the sequence $\{y_k\}$ implies

$$\lim_{k\to\infty} x_k = \lim_{k\to\infty} y_k.$$

Moreover, from $\|x_k\|_A = \sqrt{2b}$ for all k, then

$$\lim_{k\to\infty} \|y_k\|_A = \lim_{k\to\infty} \|x_k\|_A = \sqrt{2b}.$$

According to Lemma 4.1 ii), iii) and iv), both x^* and the sequence $\{y_k\}$ are on the halfline passing through the origin and verify $\|y_k\|_A < \sqrt{2b} = \|x^*\|_A$. Thus,

$$\lim_{k\to\infty} \|y_k - x^*\|_A \leq \lim_{k\to\infty} (\|x^*\|_A - \|y_k\|_A) = \sqrt{2b} - \lim_{k\to\infty} \|y_k\|_A = 0.$$

Therefore $\{y_k\}$ converges to x^*, and

$$\lim_{k\to\infty} x_k = \lim_{k\to\infty} y_k = x^*.$$

□

5 Improved version of the algorithm

The Lemma 4.1 suggests a way to simplify the algorithm. Since the interior points generated by the algorithm lie on a line, it is possible to find such a line using two of them. The solution of problem (4) can be found as the intersection of the line and the constraint. It means that the solution can be found by a direct method.

This algorithm has the advantage of being less costly than algorithm 1, but it has the disadvantage of choosing the interior point properly because the accuracy of the solution will depend strongly on the accuracy of this interior point. Figure 3 shows the behavior of the algorithm.

Algorithm 2
Given: $c \neq 0 \in \mathbb{R}^n$, $A \in R^{n\times n}$, symmetric and positive definite matrix and $b \in \mathbb{R}$, $b > 0$.

Step 1: Choose the starting point: $y_1 = 0$.

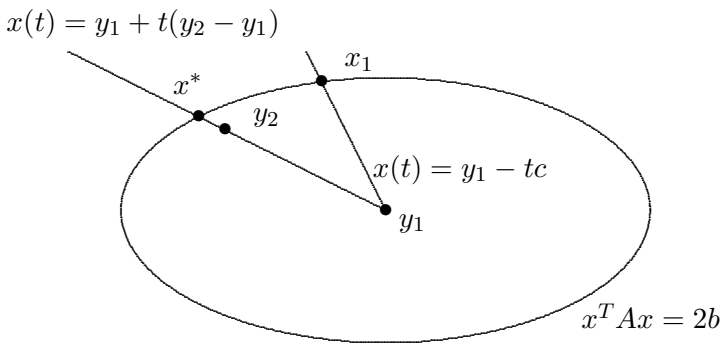

Figure 3: The behavior of the direct method

Step 2: Find a border point x_1 as the intersection between the halfline passing through y_1 in the direction $v = -c$ and the constraint.

Step 3: Find an interior point y_2 as the center of the resulting intersection between the level hyperplane $c^T x = c^T x_1$ and the constraint.

Step 4: Find the solution x^* as the intersection between the halfline passing through y_1 in the direction $v = y_2 - y_1$ and the constraint.

Let us analyze each step of the algorithm. There are no comments about **Step 1**, **Step 2** and **Step 3** because they are similar to those of **Algorithm 1**.

Let us consider **Step 4.** Since $y_1 = 0$, the point x^* is obtained as the intersection between the halfline $x(t) = ty_2,\ t > 0$, and the quadratic constraint $x^T Ax = 2b$. Then,

$$(ty_2)^T A(ty_2) = 2b, \quad t^* = \sqrt{\frac{2b}{y_2^T A y_2}}, \quad x^* = t^* y_2.$$

Then, **Algorithm 2** is established as

Algorithm 2:

Given: $c \neq 0 \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ symmetric and positive definite$b \in \mathbb{R}$, $b > 0$.
Compute: B, e and $\widehat{A} = B^T AB$.

Step 1: Choose a starting interior point: $y_1 = 0$.

Step 2: Find a border point: $x_1 = -\sqrt{\frac{2b}{c^T Ac}}c$.

Step 3: Find an interior point: $y_2 = B\widehat{y}_2 + f_1 e$ with $\widehat{y}_2 = argmin\ \frac{1}{2}\widehat{x}^T \widehat{A}\widehat{x} - \widehat{d}^T\widehat{x}$,
where $f_1 = c^T x_1$ and $\widehat{d} = -f_1 B^T Ae$.

Step 4: Find the solution: $x^* = \sqrt{\frac{2b}{y_2^T A y_2}}y_2$.

6 Numerical experience

In this section implementations of Algorithms 1 and 2 are described. The algorithms have been implemented in Matlab in an environment PC with a Pentium 4 processor with 512 MB RAM

and 2.40GHz.

The specific procedure used is given below.

- The stopping criterion used in **Algorithm 1** is: $e_k = \|y_k - x_k\| < 10^{-12}$.
- In Step 1 of Algorithm 1 and Step 3 of Algorithm 2 the point $\widehat{y}_k = argmin\ \frac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d_k}^T\widehat{x}$ is obtained by applying the spectral gradient method. It is a relatively novel non-descent method, appropriate to large scale optimization problems and competitive with the traditional conjugate gradient method. For more details see [5, 7, 8, 9].

In the tables, the following abbreviations are used:

- **n**: the number of variable of the problem.
- **it**: the amount of iterations carried out by **Algorithm 1**.
- **Nit**: the number of iteration considered from **Algorithm 1**.
- **it SG**: the number of iterations required by spectral gradient method to obtain $\widehat{y}_k = argmin\ \frac{1}{2}\widehat{x}^T\widehat{A}\widehat{x} - \widehat{d_k}^T\widehat{x}$ in the k-th iteration.
- $f_k = c^T x_k$: the objective function value at the k-th iteration.
- $e_k = \dfrac{\|y_k - x_k\|}{\|x_k\|}$: the relative error in the k-th iteration.
- $f^* = c^T x^*$: the objective value function at the estimate x^*, given by the algorithm.
- $r = \|\frac{1}{2}{x^*}^T A x^* - b\|$: the error constraint evaluated at x^*. We observe that r should be zero since the solution must satisfy the constraint.
- **CPU time**: the computing time required by the algorithm.

Now, we consider two problems, each of them with different number of variables. Both problems are considered without the linear term ($d = 0$) because any other problem can be reduced to this case.

Problem 1:

Let us consider problem (4) with $c = (1, 1, ..., 1)^T \in \mathbb{R}^n$, $A = diagonal(1, 2, ..., n) \in \mathbb{R}^{n\times n}$ and $b = 1$.

Table 1 shows the results obtained at each iteration of the iterative method with $n = 100$. Tables 2 and 3 show the results obtained applying Algorithms 1 and 2 with different values of n (n =100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000).

The fast convergence of the algorithm can be observed in Table 1: the error decreases quite fast. Also, it is possible to observe that the number of iterations required by the spectral gradient method to obtain y_k decreases as k increases.

Table 2 shows that the number of iterations required by Algorithm 1 do not increase when the number of variable is augmented. The time required to run the algorithm increases because of the operations needed at each iterations. Clearly, more computing time is needed if the number of variable is large.

Comparing Tables 2 and 3, there are not significant differences between the results obtained by Algorithms 1 and 2, despite the considerable decrease in computing time.

Nit	$f_k = c^T x_k$	$e_k = \frac{\|y_k - x_k\|}{\|x_k\|}$	**it SG**
1	-1.99007438041998	1	0
2	-2.96984668055811	0.18218129703840	132
3	-3.20683507409673	0.03193981609218	123
4	-3.22093665958930	0.00178265648084	111
5	-3.22098665492903	$6.297052951447885e-006$	94
6	-3.22098665555746	$7.915125095980291e-011$	76
7	-3.22098665555746	$5.035955884656709e-017$	18

Table 1: Iterations of Algorithm 1 - Problem 1, $n = 100$

n	$f^* = c^T x^*$	$r = \|\frac{1}{2} x^{*T} A x^* - b\|$	**CPU time**	**It**
100	-3.22098665555746	$3.330669073875470e-016$	0.0630	7
200	-3.42871140463044	$2.220446049250313e-016$	0.3430	7
300	-3.54476060695204	$3.330669073875470e-016$	1.4530	7
400	-3.62489439602770	$2.220446049250313e-015$	2.9850	7
500	-3.68587124842703	$9.992007221626409e-016$	5.9680	7
600	-3.73496410209512	$2.220446049250313e-015$	11.1250	7
700	-3.77597937377608	$1.332267629550188e-015$	18.9220	7
800	-3.81115527428657	$9.992007221626409e-016$	27.3750	7
900	-3.84191771929092	$1.110223024625157e-015$	38.9690	7
1000	-3.86923011994643	$2.220446049250313e-016$	56.9370	7

Table 2: Algorithm 1 applied to Problem 1 and different values of n

n	$f^* = c^T x^*$	$r = \|\frac{1}{2} x^{*T} A x^* - b\|$	**CPU time**
100	-3.22098665555746	$2.220446049250313e-016$	0.0320
200	-3.42871140463045	$8.881784197001252e-016$	0.1250
300	-3.54476060695204	$1.887379141862766e-015$	0.6400
400	-3.62489439602770	$8.881784197001252e-016$	1.4530
500	-3.68587124842704	$8.881784197001252e-016$	2.9850
600	-3.73496410209512	$1.332267629550188e-015$	5.3910
700	-3.77597937377609	$8.881784197001252e-016$	9.0470
800	-3.81115527428656	$5.551115123125783e-016$	12.5630
900	-3.84191771929092	$1.110223024625157e-016$	18.7970
1000	-3.86923011994643	$9.992007221626409e-016$	24.8750

Table 3: Algorithm 2 applied to Problem 1 and different values of n

Problem 2:

Let us consider problem (4) with $c = (1, 1, ..., 1)^T \in \mathbb{R}^n$, $b = 1$ and $A = \dfrac{H(v)^T H(v)}{n^3} \in \mathbb{R}^{n \times n}$, where $v = (1, 2, 3, ..., n) \in \mathbb{R}^n$ and $H(v)$ is a square Hankel matrix whose first column is v and whose elements are zero below the first anti-diagonal. This matrix is symmetric and positive definite.

Table 4 shows the results obtained at each iteration of the algorithm when $n = 100$. Tables

5 and 6 show the results obtained via Algorithms 1 and 2.

Nit	$f_k = c^T x_k$	$e_k = \frac{\|y_k - x_k\|}{\|x_k\|}$	**it SG**
1	-3.82499150774954	1	0
2	-7.25025696447532	0.09187993351090	875
3	-10.07701880238112	0.04035743849082	705
4	-12.17872638695642	0.02164347403184	753
5	-13.51681325907164	0.01141576228632	878
6	-14.16487459323087	0.00498514448544	695
7	-14.34312557007998	0.00130895964353	732
8	-14.35752222964941	$1.044190646555198e-004$	710
9	-14.35761670656568	$6.845619643013136e-007$	580
10	-14.35761671063453	$2.948193549052210e-011$	332
11	-14.35761671063453	0	27

Table 4: Iterations of Algorithm 1 - Problem 2, $n = 100$

n	$f^* = c^T x^*$	$r = \|\frac{1}{2} x^{*T} A x^* - b\|$	**CPU time**	**It**
100	-14.35761671063453	0	0.4060	11
200	-20.15598398495877	0	5.8910	13
300	-24.62326461541155	$1.110223024625157e-016$	25.4060	15
400	-28.39588023323513	0	78.4690	17
500	-31.72283979772807	$2.220446049250313e-016$	198.0780	18

Table 5: Algorithm 1 applied to Problem 2 and different values of n

n	$f^* = c^T x^*$	$r = \|\frac{1}{2} x^{*T} A x^* - b\|$	**CPU time**
100	-14.35761671063453	$2.220446049250313e-016$	0.0630
200	-20.15598398495877	0	0.7500
300	-24.62326461541155	$2.220446049250313e-016$	2.1880
400	-28.39588023323513	0	7.4060
500	-31.72283979772806	$2.220446049250313e-016$	13.3750

Table 6: Algorithm 2 applied to Problem 2 and different values of n

The observations done above are valid for Problem 2. However, it is possible to observe an increment in the number of iterations required by Algorithm 1 to achieve convergence. Moreover, it is possible to observe that the computing time required for running Problem 2 is higher than the one required for Problem 1. It is because in Problem 1, the matrix is sparse.

7 Conclusions

The problem of minimizing a linear function subject to a strictly convex quadratic constraint has been analyzed. The convergence of the algorithm is guaranteed via simple geometric considerations. The simplicity and the good behavior of proposed algorithm make it adequate to large

scale optimization problems. Since many topologic optimization problems are established as (1) the next objective is to extend the algorithm when the strictly convex quadratic constraints appear showing a block structure. The truss optimization problems, if many bars are involved, show this block structure and the extended algorithm might be successfully used.

References

[1] A. BEN-TAL and M.P. BENDSØE. A new method for optimal truss topological design. *SIAM Journal on Optimization*, 3(2):322–358, 1993.

[2] A. BEN-TAL and M. ZIBULESVSKY. Penalty/barrier multiplier methods for convex programming problems. *SIAM Journal on Optimization*, 7(2):347–366, 1997.

[3] M.P. BENDSØE and O. SIGMUND. *Topology Optimization. Theory, Methods and Applications*. Springer Verlag, Berlin, 2003.

[4] M.P. BENDSØE, A. BEN TAL, and J. ZOWE. Optimization methods for truss geometry and topology design. *Structural Optimization*, 7:141–159, 1994.

[5] R. FLETCHER. On the Barzilai-Borwein method. Technical Report NA/207, Department of Mathematics, University of Dundee, Dundee, Scotland, 2001.

[6] F. JARRE, M. KOČVARA, and J. ZOWE. Optimal truss design by interior-point methods. *SIAM Journal on Optimization*, 8(4):1084–1107, 1998.

[7] M. RAYDAN. *Convergence properties of the Barzilai and Borwein gradient method*. PhD thesis, Dept. of Mathematical Science, Rice University, Houston, Texas, 1991.

[8] M. RAYDAN. On the Barzilai and Borwein choice of the steplength for the gradient method. *IMA Journal Numerical Analysis*, 13:321–326, 1993.

[9] M. RAYDAN. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7:26–33, 1997.

[10] J. ZOWE, M. KOČVARA, and M.P. BENDSØE. Free material optimization via mathematical programming. *Computers & Structures*, 79:445–466, 1997.