Barrier function methods using Matlab

Hailay Weldegiorgis Berhe
Haramaya University Mathematics Department

Accepted 2 October, 2012

The approach in these methods is that to transform the constrained optimization problem into an equivalent unconstrained problem or into a problem with simple constraints, and solved using one (or some variant) of the algorithms for unconstrained optimization problems. Algorithms and MATLAB codes are developed using Powell’s method for unconstrained optimization problems for barrier function methods and then problems that have appeared frequently in the optimization literature which have been solved using different techniques are solved and compared amongst themselves and with other algorithms.

Key Words: Barrier parameter, Barrier problem, Normalization of constraints.

INTRODUCTION

Barrier function methods are class of sequential minimization techniques available to solve the constraint optimization problems. This approach was first proposed by Carroll in 1961 under the name created response surface technique. It is found out in the research that the sequential transformation methods converge to at least to a local minimum in most cases without the need for the convexity assumptions and with no requirement for differentiability of the objective and constraint functions. It is also revealed in the research that the Hessian matrix of the sequential transformation methods become ill-conditioned in a limit as the barrier parameter goes to zero. For problems of non-convex functions (with different local minimum points) it is recommended to solve the problem with different starting points, barrier parameters and barrier multipliers and take the best solution.

Approach was also used to solve nonlinear inequality constrained problems by Box, Davies, and Swam (1969) and Kowalik(1966). The barrier function approach has been thoroughly investigated and popularized by Fiacco and McCormick (1964, 1968).Himmelblau (1972) also discussed effective unconstrained optimization algorithms for solving barrier methods. Similar to penalty functions, barrier functions are also used to transform a constrained problem into unconstrained or into a sequence of unconstrained problems. Barrier methods construct approximations inside the feasible region and set a barrier against leaving it. These require that the interior of the feasible sets be nonempty, which is impossible if equality constraints are present. Therefore they are used with problems having only inequality constraints. This method generates a sequence of feasible points whose limit is an optimal solution to the original problem. A barrier term that prevents the points generated from leaving the feasible region is added to the objective function. The sequence of minimizers is also feasible, therefore, and hence the techniques are sometimes referred to as interior point methods. This method can be advantageous if the objective function is not defined when the constraints are violated. To describe barrier methods, we denote the penalty parameter by

\[ l(\mu) = \mu \geq 0, \text{ which is monotonically decreasing function} \]

and the barrier function by

\[ B(x) = \sum_{i=1}^{m} \Phi(g_i(x)), \]  \hspace{1cm} (1.1a)

where \( \Phi \) is continuous function of one variable over \( \{y : y < 0\} \) and satisfies

\[ \Phi(y) \geq 0 \text{ if } y < 0 \quad \text{and} \quad \lim_{y \to 0^+} \Phi(y) = \infty. \]  \hspace{1cm} (1.1b)

Thus, a typical barrier function is of the form:

\[ B(x) = \sum_{i=1}^{m} \frac{-1}{g_i(x)} \text{or} B(x) = -\sum_{i=1}^{m} \ln\left[ \min\left\{1, -g_i(x)\right\} \right]. \]  \hspace{1cm} (1.2a)

The first is commonly called the inverse barrier function.
(Carroll, 1961). Note the second barrier function in (3.2a) is not differentiable because of the term \( \min (l, -g_i(x)) \). Actually, since the property for \( \phi \) is essential only in a neighborhood of \( y = 0 \), it can be shown that the following barrier function, known as Frisch’s logarithmic barrier function,

\[
B(x) = -\sum_{i=1}^{m} \ln[-g_i(x)],
\]

(1.2b)

also admits convergence in the sense of Theorem 3.2 given below if the non-negativity of \( B(x) \) is ignored or we should define \( \log (-g) = 0 \) when \( g(x) < -1 \) to ensure that \( B \geq 0 \) though this is not a problem when constraints are expressed in normalized form, in which case \( g < -1 \) implies a highly inactive constraint which will not play a role in determining the optimum. The logarithmic barrier function was, according to Fletcher, first introduced by Ragnar Frisch in 1955 (First Norwegian Nobel Prize winner in Economics) and developed by Fiacco and McCormick [1968]. A barrier function \( B \) is one that is continuous and nonnegative over the interior of \( \{x \mid g(x) \leq 0\} \), i.e., over the set \( \{x \mid g(x) < 0\} \), and approaches \( \infty \) as the boundary is approached from the interior.

Definition 1.1: A function \( B : \mathbb{R}^n \rightarrow \mathbb{R} \) is called a barrier function if \( B \) satisfies

\[
B(x) \geq 0 \text{ for all } x \text{ that satisfies } g(x) < 0, \quad \text{and} \quad B(x) \rightarrow \infty \text{ as } \lim_{x} \max_{i} \{g_i(x)\} \rightarrow 0.
\]

Note that as far as they satisfy this definition barrier function can be any form. Consider the following Primal and barrier problems.

**Primal Problem:**

Minimize \( f(x) \)

Subject to \( g(x) \leq 0 \)

\( x \in X, \)

where \( g = (g_1, g_2, \ldots, g_m)^T \in \mathbb{R}^n \) is a vector valued function. Here, \( f, g_1, \ldots, g_m \) are continuous functions on \( \mathbb{R}^n \), and \( X \) is a nonempty set in \( \mathbb{R}^n \). Note that any equality constraints, if present, are accommodated, within the set \( X \). Alternatively, in the case of linear equality constraints, we can possibly eliminate them after solving for some variables in terms of the others, thereby reducing the dimension of the problem. The reason why this treatment is necessary is that the barrier function methods require the set \( \{x \mid g(x) < 0\} \) to be nonempty as explained above, which would obviously not be possible if the equality constraints \( h(x) = 0 \) were accommodated within the set of inequality constraints as \( h(x) \leq 0 \) and \( h(x) \geq 0 \).

**Barrier Problem:**

Assuming that the primal problem (P) has an optimal solution and for every \( \mu > 0 \) the problem to minimize \( \{f(x) + \mu p(x), x \in X\} \) has at least one optimal solution \( x_\mu \). Let \( B \) be a continuous function of the form (3.1a) satisfying the properties stated in (3.1b). The basic barrier function approach attempts to solve the problem:

Minimize \( \theta(\mu) \)

subject to \( \mu \geq 0, \)

Where \( \theta(\mu) = \inf \{f(x) + \mu B(x) : g(x) < 0, x \in X\} \). The main theorem of this section states that

Minimum \( \{f(x) : g(x) \leq 0, x \in X\} = \lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf_{\mu > 0} \theta(\mu). \)

From this result, it is clear that we can get arbitrarily close to the optimal value of the primal problem by computing \( \theta(\mu) \) for a sufficiently small \( \mu > 0 \). This result is established in Theorem 3.1.

We refer to the function \( f(x) + \mu B(x) \) as the auxiliary function. Ideally, we would like the function \( B \) to take value zero on the region \( \{x : g(x) < 0\} \) and value \( \infty \) on its boundary. This would guarantee that we would not leave the region \( \{x : g(x) \leq 0\} \), provided that the minimization problem started at the interior point. However, this discontinuity poses serious difficulties for any computational procedure. Therefore, this ideal construction of \( B \) is replaced by the more realistic requirement that \( B \) is nonnegative and continuous over the region \( \{x : g(x) < 0\} \) and that it approaches infinity as the boundary is approached from the interior. Note that \( \mu B \) approaches the ideal barrier function described above as \( \mu \) approaches zero. Given \( \mu > 0 \), evaluating \( \theta(\mu) = \inf \{f(x) + \mu B(x) : g(x) < 0, x \in X\} \) seems no simpler than solving the original problem because of the presence of the constraint \( g(x) < 0 \). However, as a result of the structure of \( B \), if we start the optimization from a point in the region \( \Omega = \{x : g(x) < 0\} \) and ignore the constraint \( g(x) < 0 \), we will reach an optimal point in \( \Omega \). This results from the fact that as we approach the boundary of \( \{g(x) \leq 0\} \) from within \( \Omega \), \( B \) approaches infinity, which will prevent us from leaving the feasible set. This is discussed further in the detailed statement of the barrier function method.

**Example 1.1:**

Consider the following problem:

Minimize \( x \)

Subject to \( -x + 1 \leq 0. \)

The constraint \( g(x) = -x + 1 \leq 0 \) is active at \( x = 1 \). By the KKT necessary condition since \( \bar{x} = 1 \) locally solves (P) (\( \bar{x} \) is a regular point), then there exist scalar \( \bar{\nu} \) for \( i \in I \) such that

\[
\nabla f(\bar{x}) + \bar{\nu} \nabla g(\bar{x}) = 0, \quad \bar{\nu} \geq 0
\]
1 - \(\bar{v}(1) = 0\) which follows that 
\(\bar{v} = 1\).

Note that the optimal solution is at \(\bar{x} = 1\) and \(f(\bar{x}) = 1\).

Consider the following barrier function:
\[ B(x) = \frac{-1}{x+1} \text{ for } x \neq 1 \]

Figure 3.1a shows \(\mu B\) for various values of \(\mu > 0\). Note that, as \(\mu \to 0\), \(\mu B\) approaches a function that has value zero over \(x > 1\) and infinity for \(x = 1\). Figure 3.1b shows the auxiliary function \(f(x) + \mu B(x) = x + \mu B\). Note that, for any \(\mu > 0\), the barrier problem is to minimize 
\[ x + \mu B(x) \text{ over } x > 1. \]
Hence, if any of the techniques for unconstrained optimization are used to minimize 
\[ x + \frac{\mu}{x-1} \text{ over } x > 1, \]
the auxiliary function \(f(x) = x + \frac{\mu}{x-1}\) is convex over \(x > 1\). Hence, if any of the techniques for unconstrained optimization are used to minimize 
\[ x + \frac{\mu}{x-1} \text{ over } x > 1, \]
we would obtain the optimal point \(x = 1 + \sqrt{\mu}\) as follows.
\[ (\mu) = x + \frac{\mu}{x-1} \quad \text{and} \quad \forall \theta(\mu) = 0 \]
Thus, \(\theta(\mu) = f(x) + \mu B(x)\) is a decreasing function of \(\mu\), and \(B(x_{\mu})\) is an increasing function of \(\mu\).

**Proof**

1. Fix \(\mu > 0\). By definition of \(\theta\), there exists a sequence \(\{x_{\mu}\}\) with \(x_{\mu} \in X\) and \(g(x_{\mu}) < 0\) such that \(f(x_{\mu}) + \mu B(x_{\mu}) = \theta(\mu)\). By assumption, \(x_{\mu}\) has a convergent subsequence \(\{x_{k}\}\) with limit \(x_{\mu} \in X\). By continuity of \(g\), \(\lim_{k \to \infty} g(x_{k}) = g(x_{\mu}) \leq 0\). We show that \(g(x_{\mu}) < 0\). If not, \(g(x_{\mu}) = 0\) for some \(i\); and since the barrier function \(B\) satisfies (3.1), and \(\forall \theta(\mu) = 0\), it is the only optimal point.

Note that \(f(x_{\mu}) + \mu B(x_{\mu}) = 1 + 2\sqrt{\mu}\). Obviously, as \(\mu \to 0\), \(x_{\mu} \to \bar{x}\) and \(f(x_{\mu}) + \mu B(x_{\mu}) \to f(\bar{x})\).

**Lemma 1.1:**

Let \(f, g\) be continuous functions on \(\mathbb{R}^n\), and let \(X \neq \emptyset\) be closed set in \(\mathbb{R}^n\). Suppose that the set \(S = \{x \in X : g(x) < 0\} \neq \emptyset\) and that \(B\) is a barrier function of the form (3.1) and is continuous on \(\{x : g(x) < 0\}\). Furthermore, Suppose that for any given \(\mu > 0\), if \(\{x_{\mu}\}\) in \(X\) satisfies \(g(x_{\mu}) < 0\) and \(f(x_{\mu}) + \mu B(x_{\mu}) \to \theta(\mu)\), then \(\{x_{\mu}\}\) has a convergent subsequence, then,

1. For each \(\mu > 0\), there exists an \(x_{\mu} \in X\) with \(g(x_{\mu}) < 0\) such that
\[ \theta(\mu) = f(x_{\mu}) + \mu B(x_{\mu}) = \inf\{f(x) + \mu B(x) : g(x) < 0, x \in X\}. \]
2. \(\inf\{f(x) : g(x) \leq 0, x \in X\} \leq \inf\{\theta(\mu) : \mu > 0\}\)
3. For \(\mu > 0\), \(\theta(\mu)\) and \(f(x_{\mu})\) are no decreasing functions of \(\mu\), and \(B(x_{\mu})\) is a no increasing function of \(\mu\).
\[ \theta(\mu) \geq \theta(\lambda). \]

Noting part 1, there exist \( x_\lambda \) and \( x_\mu \) such that
\[ \theta(\lambda) = f(x_\lambda) + \lambda B(x_\lambda) \leq f(x_\mu) + \lambda B(x_\mu), \text{ since } x_\lambda \in X. (1.3a) \]
and
\[ \theta(\mu) = f(x_\mu) + \mu B(x_\mu) \leq f(x_\lambda) + \mu B(x_\lambda), \text{ since } x_\mu \in X. (1.3b) \]
Adding (1.3a) and (1.3b) we have
\[ f(x_\lambda) + \lambda B(x_\lambda) + f(x_\mu) + \mu B(x_\mu) \leq f(x_\mu) + \lambda B(x_\mu) + f(x_\lambda) + \mu B(x_\lambda). \]
By simplifying like terms, we get
\[ \lambda B(x_\lambda) + \mu B(x_\mu) \leq \lambda B(x_\mu) + \mu B(x_\lambda), \]
which implies by rearranging that
\[ (\lambda - \mu)[B(x_\lambda) - B(x_\mu)] \leq 0. \]
Since \( \lambda - \mu \leq 0 \) by assumption, we have \( B(x_\lambda) - B(x_\mu) \geq 0 \). Thus,
\[ B(x_\lambda) \geq B(x_\mu). \]
\( B(x_\mu) \) is a non-increasing function of \( \mu \).
By (1.3a)
\[ f(x_\lambda) + \lambda B(x_\lambda) \leq f(x_\mu) + \lambda B(x_\mu). \]
Then
\[ f(x_\lambda) \leq f(x_\mu), \text{ since } B(x_\lambda) \geq B(x_\mu). \]
Thus part 3 holds and the proof is complete.

From the above, \( \theta \) is a non-decreasing function of \( \mu \) so that,
\[ \inf_{\mu > 0} \theta(\mu) = \lim_{\mu \to 0^+} \theta(\mu) = \inf(\theta(\mu) : \mu > 0). \]

We now show the validity of using barrier functions for solving constrained problems. Theorem 3.2 below shows that the optimal solution to the primal problem is indeed equal to \( \lim_{\mu \to 0^+} \theta(\mu) \), so that it could be solved by a single problem of the form to minimize \( f(x) + \mu B(x) \) subject to \( x \in X \), where \( \mu \) is sufficiently small, or it can be solved through a sequence of problems of the above form with decreasing values of \( \mu \). Similar to exterior penalty functions we don't just choose one small value rather we start with some \( \mu_1 \) and generate a sequence of points.

**Theorem 1.2: (Barrier Convergence Theorem)**

Let \( f : \mathbb{R}^n \to \mathbb{R} \) and \( g : \mathbb{R}^n \to \mathbb{R}^m \) be continuous functions, and let \( \emptyset \neq \mathbb{X} \subseteq \mathbb{R}^n \). Suppose that the set \( \{x \in \mathbb{X} : g(x) < 0\} \neq \emptyset \). Furthermore, suppose that the primal problem to minimize \( f(x) \) subject to \( g(x) \leq 0, \ x \in \mathbb{X} \) has an optimal solution \( \bar{x} \) with the following property. Given any neighborhood \( N \) around \( \bar{x} \), there exists an \( x \in \mathbb{X} \cap N \) such that \( g(x) < 0 \). Then,
\[ \min \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \} = \lim_{\mu \to 0^+} \theta(\mu) = \inf_{\mu > 0} \theta(\mu). \]

Letting \( \theta(\mu) = f(x_\mu) + \mu B(x_\mu) \), where \( x_\mu \in \mathbb{X} \) and \( g(x_\mu) < 0 \), then the limit of any convergent subsequence of \( \{x_\mu\} \) is an optimal solution to the primal problem and, furthermore, \( \mu B(x_\mu) \to 0 \) as \( \mu \to 0^+ \).

**Proof**

Let \( \bar{x} \) be an optimal solution to the primal problem satisfying the stated property, and let \( \varepsilon > 0 \) be arbitrary. By continuity of \( f \) and by the assumption of the Theorem, there is an \( \bar{x} \in \mathbb{X} \) with \( g(\bar{x}) < 0 \) such that \( f(\bar{x}) + \varepsilon > f(\bar{x}). \) Then, for \( \mu > 0 \)
\[ f(\bar{x}) + \varepsilon + \mu B(\bar{x}) > f(\bar{x}) + \mu B(\bar{x}) \geq \theta(\mu). \]
Taking the limit as \( \mu \to 0^+ \), it follows that
\[ f(\bar{x}) + \varepsilon > \lim_{\mu \to 0^+} \theta(\mu). \]
Since this inequality holds for each \( \varepsilon > 0 \), we get \( f(\bar{x}) \geq \lim_{\mu \to 0^+} \theta(\mu) \).

From part 2 of Lemma 3.1 above we have
\[ \theta(\mu) \geq \inf \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \} = f(\bar{x}), \text{ for each } \mu \geq 0. \]

Equating (3.4a) and (3.4b) we have
\[ f(\bar{x}) = \lim_{\mu \to 0^+} \theta(\mu). \]

For \( \mu \to 0^+ \), and since \( B(x_\mu) \geq 0 \) and \( x_\mu \) is feasible to the original problem, it follows that
\[ \theta(\mu) = f(x_\mu) + \mu B(x_\mu) \geq f(x_\mu) \geq f(\bar{x}) \]
\[ \text{this implies that } \theta(\mu) \geq f(\bar{x}) = \inf \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \}, \text{ for each } \mu \geq 0 \text{ and hence} \]
\[ \inf_{\mu > 0} \theta(\mu) \geq f(\bar{x}) = \inf \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \}, \text{ for each } \mu \geq 0 \text{ and hence} \]
\[ \inf_{\mu > 0} \theta(\mu) \geq \min \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \}, \text{ for each } \mu \geq 0 \text{ and hence} \]
In view of part 2 of Lemma 3.1
\[ \inf_{\mu > 0} \theta(\mu) = \min \{ f(x) : g(x) \leq 0, \ x \in \mathbb{X} \} = \lim_{\mu \to 0^+} \theta(\mu). \]
By (1.4d)
\[ \theta(\mu) = f(x_\mu) + \mu B(x_\mu) \geq f(x) \geq f(\bar{x}) \quad (1.4f) \]

Taking the limit as \( \mu \to 0^+ \) and noting that \( f(\bar{x}) = \lim_{\mu \to 0^+} \theta(\mu) \), it follows that

\[ \lim_{\mu \to 0^+} \theta(\mu) = \lim_{\mu \to 0^+} (f(x_\mu) + \mu B(x_\mu)) = f(\bar{x}) = \lim_{\mu \to 0^+} f(x_\mu). \]

Hence, \( \mu B(x_\mu) \to 0 \) as \( \mu \to 0^+ \).

Furthermore, if \( \{x_\mu\} \) has a convergent subsequence with limit \( x' \), then \( f(x') = f(\bar{x}) \). Since \( x_\mu \) is feasible solution to the original problem for each \( \mu \), it follows that \( x' \) is also feasible and, hence, optimal.

Note that since the initial point as well as each of the subsequent points generated in this method lies inside the acceptable region of the design space \( \{x : g(x) \leq 0\} \), for each \( \mu \), the method is classified as interior penalty function formulation.

**Remark:** The assumption of the above Theorem 1.2 holds if \( \{x \in X : g(x) \leq 0\} \) is compact. Assumptions under which such a point \( x_\mu \) exists are given in Lemma 1.1.

### Karush Kuhn Tucker Multipliers at Optimality

Under certain regularity conditions, the barrier interior penalty method also produces a sequence of Lagrange multiplier estimates that converge to an optimal set of Lagrange multipliers. To see this, consider problem (P) to minimize \( f(x) \) subject to \( g_i(x) \leq 0 \) for \( i = 1, \ldots, m \) and \( X = \mathbb{R}^n \). (The case where \( X \) might include additional inequality constraints is easily treated in a likewise fashion. The Barrier function problem is then given by

\[ \min f(x) \text{ subject to } g_i(x) \leq 0, \quad i = 1, \ldots, m, \quad x \in \mathbb{R}^n. \]

Let \( x_\mu \to \bar{x} \) and let \( l = \{i : g_i(\bar{x}) = 0\} \) and \( N = \{i : g_i(\bar{x}) < 0\} \) be defined by (3.8). Then if \( x_\mu \to \bar{x} \), satisfies the linear independence condition for gradient vectors of active constraints, then \( u_\mu \to \bar{u} \), where \( \bar{u} \) is a vector of Karush-Kuhn-Tucker multipliers for the optimal solution \( \bar{x} \) of (P).

**Proof**

Let \( x_\mu \to \bar{x} \) and let \( l = \{i : g_i(\bar{x}) = 0\} \) and \( N = \{i : g_i(\bar{x}) < 0\} \).

For all \( i \in N \),

\[ (u_\mu)_i = \mu g_i'(x_\mu) \to 0, \]

since \( \mu \to 0^+ \) and \( g_i(x_\mu) \to g_i(\bar{x}) \leq 0, \) and \( g_i'(x_\mu) \) is finite. Also \( (u_\mu)_i \geq 0 \) for all \( i \in l \) as \( \mu \to 0^+ \).

Suppose \( u_\mu \to \bar{u} \) as \( \mu \to 0^+ \). Then \( \bar{u}_i \geq 0 \), and \( \bar{u}_i = 0 \) for all \( i \in N \). From the continuity of all functions involved, (3.7) implies that

\[ \nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) = 0, \quad \bar{u}_i \geq 0, \quad \bar{u}^T g(\bar{x}) = 0. \]

Thus \( \bar{u} \) is a vector of Karush-Kuhn-Tucker multipliers. It remains to show that \( u_\mu \to \bar{u} \) for some unique \( \bar{u} \).

Suppose \( \{u_\mu\} \) has no accumulation point, then \( \|u_\mu\| \to \infty \). But then define \( w_\mu = \frac{u_\mu}{\|u_\mu\|} \) and then \( \|w_\mu\| = 1 \) for all \( \mu \), and so the sequence \( \{w_\mu\}_{\mu=1}^{\infty} \) has some accumulation point \( \bar{w} \). For all \( i \in N \), \( (w_\mu)_i = 0 \) for all \( \mu \) large, where by \( \bar{w}_i = 0 \) for all \( i \in N \) and

\[ \sum_{i=1}^{m} (w_\mu)_i \nabla g_i(x_\mu) = \sum_{i=1}^{m} (w_\mu)_i \nabla g_i(x_\mu) = \sum_{i=1}^{m} \left( \frac{u_\mu}{\|u_\mu\|} \right) \nabla g_i(x_\mu) = -\frac{\nabla f(x_\mu)}{\|u_\mu\|} \]

for \( \mu \) small enough. As \( \mu \to 0^+ \) we have \( x_\mu \to \bar{x} \), \( (w_\mu) \to \bar{w} \), and \( \|u_\mu\| \to \infty \) by assumption, and so the above equation becomes...
\[ \sum_{i=1}^{m} \left( w_i \right) \nabla g_i(x_\mu) = 0 \text{ and } ||w|| = 1, \]

which violates the linear independence condition. Therefore \( \{u_\mu\} \) is a bounded sequence, and so has at least one accumulation point.

Now suppose that \( \{u_\mu\} \) has two accumulation points, \( \bar{u} \) and \( \hat{u} \). Note \( \bar{u}_i = 0 \) for \( i \in \mathbb{N} \), and so

\[ \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) = -\nabla f(\bar{x}) = \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) \text{ which follows that} \]
\[ \sum_{i=1}^{m} (\bar{u}_i - \bar{u}) \nabla g_i(\bar{x}) = 0. \]

But by the linear independence condition, \( \bar{u}_i - \bar{u} = 0 \) for all \( i \in I \), and so \( u_i = \bar{u}_i \). This implies that \( \bar{u} = \hat{u} \).

### Computational Difficulties Associated with Barrier Methods

The use of barrier functions for solving constrained nonlinear programming problems also faces several computational difficulties. First, the search must start with a point \( x \in X \) with \( g(x) < 0 \). For some problems, finding such a point may not be easy task. Another problem is that because of the structure of \( \mathbf{B} \), and for small values of the parameter \( \mu \), most search techniques may face serious ill-conditioning and difficulty with round-of errors while solving the problem to minimize \( f(x) + \mu \mathbf{B}(x) \) over \( x \in X \), especially as the boundary of the region \( \{ x : g(x) \leq 0 \} \) is approached. In fact, as the boundary is approached, and since search techniques often use discrete steps, a step leading outside the region \( \{ x : g(x) \leq 0 \} \) may indicate a decrease in the value of \( f(x) + \mu \mathbf{B}(x) \), a false success. Thus, an explicit check of the value of the constraint function \( g \) is needed to guarantee that we do not leave the feasible region.

Let us examine the eigenvalue structure of the Hessian of the objective function (3.5) at the optimum \( x_\mu \) as \( \mu \to 0^+ \) to see the ill-conditioning effect.

Consider the barrier function of the form

\[ \mathbf{B}(x) = \Phi(g(x)), \]

Then Lagrange multipliers and ill-conditioned Hessians are again inevitable. Rather than parallel the earlier analysis of penalty functions, we illustrate the conclusions with an example.

Define

\[ \mathbf{B}(x) = \sum_{i=1}^{m} \frac{-1}{\bar{b}_i(x)} \]

The Barrier objective

\[ \theta(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \frac{1}{\bar{b}_i(x)} \]

has its minimum at a point \( x_\mu \) satisfying

\[ \nabla f(x_\mu) + \mu \sum_{i=1}^{m} \frac{1}{\bar{b}_i(x_\mu)} \nabla g_i(x_\mu) = 0. \]

Thus, we define \( \{ u_\mu \} = \frac{-\mu}{\bar{b}_i(x_\mu)} \). Then (3.11) can be written as

\[ \nabla f(x_\mu) + \sum_{i=1}^{m} (u_\mu)_i \nabla g_i(x_\mu) = 0. \]

Over a convergent subsequence we have \( \{ x_\mu \} \to \bar{x} \) and, assuming that \( \bar{x} \) is a regular point, we have \( u_\mu \to \bar{u} \), the optimal Lagrange multipliers. This implies that if \( g_i \) is an active constraint,

\[ (u_\mu)_i = \frac{-\mu}{\bar{b}_i(x_\mu)^2} - \bar{u}_i < \infty \text{ for all } i = 1, \ldots, m, \]

as \( \mu \to 0^+ \).

Next, evaluating the Hessian \( Q(x, \mu) \) of \( \theta(x, \mu) \), we have

\[ Q(x, \mu) = \nabla^2 f(x_\mu) + \mu \sum_{i=1}^{m} \frac{1}{\bar{b}_i(x_\mu)} \nabla g_i(x_\mu) \nabla g_i(x_\mu)^\top - \mu \sum_{i=1}^{m} \frac{2}{\bar{b}_i(x_\mu)} \nabla g_i(x_\mu) \nabla g_i(x_\mu)^\top = \nabla^2 L(x_\mu) - \mu \sum_{i=1}^{m} \frac{2}{\bar{b}_i(x_\mu)} \nabla g_i(x_\mu) \nabla g_i(x_\mu)^\top. \]

As \( \mu \to 0^+ \) we have

\[ \bar{b}_i(x_\mu)^\top \to \begin{cases} \infty, & \text{if } g_i \text{ is active at } \bar{x} \\ 0, & \text{if } g_i \text{ is inactive at } \bar{x}. \end{cases} \]

So that we may write, from (3.13),

\[ Q(x, \mu) \to \nabla^2 L(\bar{x}) - 2 \sum_{i=1}^{m} \frac{\bar{u}_i}{\bar{b}_i(\bar{x})} \nabla g_i(\bar{x}) \nabla g_i(\bar{x})^\top \]

where \( I \) is the set of indices corresponding to active constraints. Thus, the Hessian of the barrier objective function has exactly the same structure as that of penalty objective functions. Therefore barrier method also faces the difficulties that appear in the penalty methods in the minimization of the corresponding unconstrained optimization problem. According to Fletcher [17], several extensions to barrier function methods have been proposed when solving general nonlinear programming problems in order to avoid the difficulties associated with the ill-conditioning as the barrier parameter approaches to zero and we consider some of them in the next sections.

### General Description of the Barrier Function Method Algorithm

The detail of this and a MATLAB computer program for
implementing the barrier method using Powell’s method of unconstrained minimization is given in the appendix.

**Algorithm 1.1: Algorithm for the Barrier Function Method**

To solve the sequence of unconstrained problems with monotonically decreasing values of \( \mu_k \), let \( \{\mu_k\}, k = 1, \ldots \) be a sequence tending to zero such that \( \mu_k \geq 0 \) and \( \mu_k > \mu_{k+1} \).

Now for each \( k \) we solve the problem:

Minimize \( \{\theta(x, \mu_k), x \in X\} \).

To obtain \( x_k \), the optimum it is assumed that problem \((3.15)\) has a solution for all positive values of \( \mu_k \). A simple implementation known as the sequential unconstrained minimization technique (SUMT) is given below.

**Step 0:** (Initialization) Select a growth parameter \( \beta \in (0, 1) \) and a stopping parameter \( \varepsilon > 0 \) and an initial value of the barrier parameter \( \mu_0 \). Choose a starting point \( x_0 \) which is feasible and formulate the augmented objective function \( \theta(x, \mu_0) \). Let \( k = 1 \).

**Step 1:** (Iterative) Starting from \( x_{k-1} \), use an unconstrained search technique to find the point that minimizes \( \theta(x, \mu_{k-1}) \) and call it \( x_k \).

**Step 2:** (Stopping Rule) If the distance between \( x_{k-1} \) and \( x_k \) is smaller than \( \varepsilon \), i.e., \( |x_{k-1} - x_k| < \varepsilon \) or the difference between two successive objective function values is smaller than \( \varepsilon \), i.e., \( |\theta(x_{k-1}) - \theta(x_k)| < \varepsilon \), stop with \( x_k \) an estimate of the optimal solution. Otherwise, put \( \mu_k = \beta \mu_{k-1} \), and formulate the new \( \theta(x, \mu_k) \) and put \( k = k+1 \) and return to the iterative step.

**Considerations for Implementation of the Barrier Method**

Although the algorithm is straightforward, there are a number of points to be considered in implementing the algorithm. These are

1. The starting feasible point \( x_1 \) may not be readily available in some cases.
2. A suitable value of the initial penalty parameter (\( \mu_1 \)) has to be found.
3. A proper value has to be selected for the multiplication factor, \( \beta \).
4. Suitable convergence criteria have to be chosen to identify the optimum point.
5. The constraints have to be normalized so that each one of them varies between -1 and 0 only or define different penalty parameters for different constraints. All these aspects are discussed in the following paragraphs.

**Starting Feasible Point \( x_1 \)**

In most engineering problems, it will not be very difficult to find an initial point \( x_1 \) satisfying all the constraints, \( g_i(x_1) \). As an example, consider the problem of minimum weight design of a beam whose deflection under a given loading condition has to remain less than or equal to a specified value. In this case one can always choose the cross section of the beam to be very large initially so that the constraint remains satisfied. The only problem is that the weight of the beam (Objective) corresponding to this initial design will be very large. Thus in most of the practical problems, we will be able to find a feasible starting point at the expense of a large value of the objective function. However, there may be some situations where the feasible design points could not be found so easily. In such cases, the required feasible starting points can be found by using the barrier function method itself as follows:

1. Choose an arbitrary point \( x_1 \), and evaluate the constraints \( g_i(x) \) at the point \( x_1 \). Since the point \( x_1 \) is arbitrary, it may not satisfy all the constraints with strict inequality sign. If \( r \) out of a total of \( m \) Constraints are violated; renumber the constraints such that the last \( r \) constraints will become the violated ones, that is,

\[
g(x_1) < 0, \quad i = 1, 2, \ldots, m-r
\]

\[
g(x_1) \geq 0, \quad i = m-r +1, m-r + 2, \ldots, m.
\]

2. Identify the constraint which is violated most at the point \( x_1 \), that is find the integer \( k \) such that

\[
g(x_1) = \max[g_i(x_1)] \text{ for } i = m - r + 1, \ m - r + 2, \ldots, m.
\]

3. Now formulate a new optimization problem as:

Find \( x \) which minimizes \( g_i(x) \)
Subject to

\[
g_i(x) \leq 0, \text{ for } i = 1, 2, \ldots, m-r
\]

\[
g_i(x) - g_k(x_i) \leq 0, \text{ for } i = m - r + 1, m-r + 2, \ldots, k-1, k+1, \ldots, m.
\]

4. Solve the optimization problem formulated in step (3) by taking the point \( x_1 \) as a feasible point using the barrier penalty function method. Note that this optimization method can be terminated whenever the value of the objective function \( g_i(x) \) drops below zero. Thus the solution obtained \( x_1 \) will satisfy at least one more constraint than did the original point \( x_1 \).

5. If all the constraints are not satisfied at the point \( x_1 \), set the new starting point as \( x_1 = x_1 \), and renumber the constraints such that the last \( r \) constraints will be

6. unsatisfied ones (this value of \( r \) will be different from the
previous value), and go to step 2.

This procedure is repeated until all the constraints are satisfied, and a point $x_1 = x_M$ is obtained for which $g_i(x) < 0$, i = 1, . . ., m.

If the constraints are consistent, it should be possible to obtain, by applying the above procedure, a point $x_1$ that satisfy all the constraints. However, there may exist situations in which the solution of the problem formulated in step 3 gives the unconstrained or constrained local minimum of $g_i(x)$ that is positive. In such cases, one has to start afresh with a new point $x_1$ from step 1 onward.

**Initial Value of the Barrier Parameter $\mu_1$**

Since the unconstrained minimization of $\theta(x, \mu_k)$ is to be carried out for a decreasing sequence of $\mu_k$, it might appear that by choosing a very small value of $\mu_1$, we can avoid an excessive number of minimizations of the function $\theta$. But from computational point of view, it will be easier to minimize the unconstrained function $\theta(x, \mu_k)$ if $\mu_k$ is large. This can be seen qualitatively from Fig.3.1a. As the value of $\mu_k$ becomes smaller, the value of the function changes more rapidly in the vicinity of the minimum $\theta_k$. Since it is easier to find the minimum of a function whose graph is smoother, the unconstrained minimization of $\theta_k$ will be easier if $\mu_k$ is large. However, the minimum of $\theta_k$, $x_k$, will be farther away from the desired minimum $\bar{x}$ if $\mu_k$ is large. Thus it requires an excessive number of unconstrained minimization of $\theta(x, \mu_k)$ (for several values of $\mu_k$) to reach the point if $\mu_1$ is selected to be very large. Thus a moderate value has to be chosen for the initial penalty parameter $\mu_1$. In practice, a value of $\mu_1$ which gives the value of $\theta(x, \mu_i)$ approximately equal to 1.1 to 2.0 times the value of $f(x_1)$ has been found quite satisfactory in achieving quick convergence of the process. Thus for any initial feasible starting point $x_1$, the value of $\mu_1$ can be taken as

$$\mu_1 \approx 0.1 \text{ to } 1.0 \frac{f(x_1)}{\sum_{i=1}^{m} g_i(x_1)} \quad (1.18)$$

The most common used initial barrier parameters in the literature are 1000, 100, 10, 1, and 0.1.

**Subsequent Values of the Barrier Parameter**

Once the initial value of the $\mu_k$ is chosen, the subsequent values of $\mu_k$ have to be chosen such that

$$\mu_{k+1} < \mu_k. \quad (1.19)$$

For convenience, the value of $\mu_k$ is chosen according to the relation

$$\mu_{k+1} = \beta \mu_k, \quad (1.20)$$

where $0 < \beta < 1$. The value of $\beta$ can be taken as 0.1, 0.2, 0.5, etc.

**Convergence Criteria**

Since the unconstrained minimization of $\theta(x, \mu_k)$ has to be carried out for a decreasing sequence of values $\mu_k$, it is necessary to use proper convergence criteria to identify the optimum point and to avoid an unnecessary large number of unconstrained minimizations. The process can be terminated whenever the following conditions are satisfied:

The relative difference between the values of the objective function obtained at the end of any two consecutive unconstrained minimizations falls below a small number $\varepsilon$, that is

$$\left| \frac{f(x_k) - f(x_{k-1})}{f(x_k)} \right| \leq \varepsilon. \quad (1.21)$$

The difference between the optimum points $x_k$ and $x_{k-1}$ becomes very small, this can be judged in several ways as:

$$|\Delta x| \leq \varepsilon_2. \quad (1.22)$$

where $\Delta x = x_k - x_{k-1}$, and $(\Delta x)_i$ is the $i$th component of the vector $\Delta x$.

Max $|\Delta x| \leq \varepsilon_3. \quad (1.23)$

$$|\Delta x| = |(\Delta x)^2 + (\Delta x)^2 + \cdots + (\Delta x)^2|^{1/2} \leq \varepsilon_4. \quad (1.24)$$

Note that the values of $\varepsilon_2$ to $\varepsilon_4$ have to be chosen depending on the characteristics of the problem at hand.

**Normalization of the Constraints**

A structural optimization problem, for example, might have constraints on the deflection ($\delta$) and the stress ($\sigma$) as:

$$g_1(x) = \delta(x) - \delta_{\text{max}} \leq 0 \quad (1.25)$$
$$g_2(x) = \sigma(x) - \sigma_{\text{max}} \leq 0 \quad (1.26)$$

where the maximum allowable values are given by $\delta_{\text{max}} = 0.5$ in. and $\sigma_{\text{max}} = 20,000$ psi. If a design vector $x_1$ gives the values of $g_1$ and $g_2$ as $-0.2$ and $-10,000$, the contribution of $g_1$ will be much larger than that of $g_2$ (by an order of $10^4$) in the formulation of the $\theta$-function. This will badly affect the convergence rate during the minimization of $\theta$-function. Thus it is advisable to normalize the constraints so that they vary between $-1$ and 0 as far as possible. For the constraints shown in
(3.25) and (3.26), the normalization can be done as
\[ g_1(x) = \frac{g_i(x)}{\delta_{\max}(x)} - 1 \leq 0 \]  
(1.27)

\[ g_2(x) = \frac{g_2(x)}{\sigma_{\max}(x)} - 1 \leq 0. \]  
(1.28)

If the constraints are not normalized as shown in (1.27) and (1.28), the problem can still be solved effectively by defining different penalty parameters for different constraints as
\[ \theta(x, \mu_k) = f(x) - \sum_{i=1}^{m} \frac{R_i}{\delta_i(x)} \]

where \( R_1, R_2, \ldots, R_m \) are selected such that the contributions of different \( g_i(x) \) to the \( \theta \)-function will be approximately the same at the initial point \( x_i \). When the unconstrained minimization of \( \theta(x, \mu_k) \) is carried for a decreasing sequence of values of \( \mu_k \), the values of \( R_1, R_2, \ldots, R_m \) will not be altered; however, they are expected to be effective in reducing the disparities between the contributions of the various constraints to the \( \theta \)-function.

Test Problems (Testing Practical Examples)

Example 1:

At Gotham City airport terminal there are 10 arrival gates (A to J respectively). A pictorial representation of the terminal is given below with the location of the gates being:

<table>
<thead>
<tr>
<th>Gate</th>
<th>x coordinate</th>
<th>y coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>F</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>G</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>H</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>I</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>20</td>
<td>2</td>
</tr>
</tbody>
</table>

Luggage from arriving flights is unloaded at these gates and moved to a passenger luggage pick-up point. It is estimated that the number of pieces of luggage arriving per day at each gate (A to J respectively) is: 3600, 2500, 1800, 2200, 1000, 4500, 5600, 1400, 1800 and 3000 respectively. Where should the passenger luggage pick-up point be located in order to minimize movement of luggage?

SOLUTION

In order to logically locate the passenger luggage pick-up point we need to make use of the amount of luggage flowing from the gates to the pick-up point. Logically a gate from which there is a large flow should be nearer to the pick-up point than a gate with a small flow. Informally therefore we would like to position the pick-up point so as to minimize the sum over all gates \( g \) (distance between \( g \) and the pick-up point) multiplied by (flow between \( g \) and the pick-up point). Note here the package terminology is somewhat peculiar:

1. Existing facilities - are points where we know in advance exactly where they are and they are fixed in position.
2. New facilities - are points where we do not know where they are and their location is what we have to determine (using the package).

Note here that in solving the problem we need to specify the appropriate distance model. This is because as we do not yet know where the new facility (luggage pick-up point) is to be we cannot specify the distance between it and the gates without a general expression for calculating the distance between two locations. If \((x_i, y_i)\) and \((x_j, y_j)\) represent the coordinates of two locations \( i \) and \( j \) then the distance model measures can be:

1. rectilinear - distance between \( i \) and \( j \) is: \(|x_i-x_j| + |y_i-y_j|\)
2. Euclidean - distance between \( i \) and \( j \) is: \((x_i-x_j)^2 + (y_i-y_j)^2\)^{0.5}
3. squared Euclidean - distance between \( i \) and \( j \) is: \((x_i-x_j)^2 + (y_i-y_j)^2\)

The rectilinear distance measure is often used for factories, American cities, etc which are laid out in the form of a rectangular grid. For this reason it is sometimes called the Manhattan distance measure. The Euclidean distance measure is used where genuine straight line travel is possible. The squared Euclidean distance measure is used where straight line travel is possible but where we wish to discourage excessive distances (squaring large distance number results in an even larger distance number and recall that we use the distance number in the objective which we are trying to minimize). Here we used the Euclidean distance to solve the problem using MATLAB. This problem is solved using a package. The output from the package for each of the distance measures is shown below.

From the package output we can see that the location for the luggage pick-up point should be:

<table>
<thead>
<tr>
<th>Distance measure</th>
<th>x coordinate</th>
<th>y coordinate</th>
<th>objective value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectilinear</td>
<td>247700</td>
<td>10.12</td>
<td>8.98</td>
</tr>
<tr>
<td>Euclidean</td>
<td>189,847.06</td>
<td>9.05</td>
<td>7.67</td>
</tr>
<tr>
<td>Squared Euclidean</td>
<td>1,594,206.25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the package output we can see that the location for the luggage pick-up point should be:
To solve the problem using MATLAB the mathematical model is:

\[
\text{Minimize } f(x) = 3600\sqrt{(x_1)^2 + (x_2 - 2)^2} + 2500\sqrt{(x_1 - 2)^2 + (x_2 - 4)^2} + 1800\sqrt{(x_1 - 5)^2 + (x_2 - 6)^2} + 2200\sqrt{(x_1 - 5)^2 + (x_2 - 10)^2} + 1000\sqrt{(x_1 - 7)^2 + (x_2 - 15)^2} + 4500\sqrt{(x_1 - 10)^2 + (x_2 - 15)^2} + 5600\sqrt{(x_1 - 12)^2 + (x_2 - 10)^2} + 1400\sqrt{(x_1 - 12)^2 + (x_2 - 6)^2} + 1800\sqrt{(x_1 - 15)^2 + (x_2 - 4)^2} + 3000\sqrt{(x_1 - 20)^2 + (x_2 - 2)^2}.
\]

Subject to 
\[-x_1 \leq 0, \quad x_2 \leq 0.
\]

The corresponding unconstrained optimization problem is:

\[
\theta(x, \mu) = f(x) - \mu\left(\frac{1}{x_1} + \frac{1}{x_2}\right).
\]

The barrier penalty function method, coupled with the Powell's method for unconstrained minimization, and golden bracket and golden search method for one-dimensional search, is used to solve this problem.

Optimum solution point using Package is \(x = (10.12, 8.98)\)

Optimum solution is at \(f^* = 189,847.06\)

Optimum solution point using MATLAB is \(x = (10.134797162042, 8.992485968315)\) and optimum solution is at \(f^* = 189,846.8168417850\).

And the iteration step using MATLAB for barrier method and the necessary data are given as follows:

Initial:

\(x_1 = [100; 100];\)
\(\mu = 100; \beta = 0.1;\)
\(\text{tol} = 1.0e-3; \text{tol1} = 1.0e-3; h = 0.1; N = 10;\)

<table>
<thead>
<tr>
<th>(\mu)</th>
<th>(\text{xmin})</th>
<th>(\text{fmin})</th>
<th>(\text{aug}_{\text{min}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0</td>
<td>((100.00000000000, 100.00000000000))</td>
<td>3554019.377700200</td>
<td>3554017.377700200</td>
</tr>
<tr>
<td>10.0</td>
<td>((10.134145350628, 8.991963447856))</td>
<td>189846.8174827870</td>
<td>189844.7186155895</td>
</tr>
<tr>
<td>1.00</td>
<td>((10.134732528902, 8.992434187711))</td>
<td>189846.8168418902</td>
<td>189846.6069730092</td>
</tr>
<tr>
<td>0.10</td>
<td>((10.134790904938, 8.992481220972))</td>
<td>189846.8168418520</td>
<td>189846.7958544489</td>
</tr>
<tr>
<td>0.01</td>
<td>((10.134797150735, 8.992485967800))</td>
<td>189846.8168417850</td>
<td>189846.8147430459</td>
</tr>
<tr>
<td>0.001</td>
<td>((10.134797162042, 8.992485968315))</td>
<td>189846.8168417850</td>
<td>189846.8166319111</td>
</tr>
</tbody>
</table>
Example 2:

A company has three factories that are located at the points \((-16, 4), (6, 5), \) and \((3, -9), \) respectively, in the \(x_1x_2\)-plane, and the numbers of deliveries to those factories are 5, 6, and 10 per month, respectively. The company has a plan to build a new warehouse in its site bounded by \(|x_1-1| + |x_2-1| \leq 2\) and is trying to minimize the monthly mileage of delivery trucks in determining the location of a new warehouse on the assumption that the distance between two points represents the driving distance.

(a) What is the objective function that must be defined in the program?
(b) What is the statement defining the inequality constraint?
(c) Use Barrier function method to get the optimum location of the new warehouse.

Solution

The mathematical model of this warehouse problem is:

\[
\begin{align*}
\text{Minimize} & \quad f(x) = 5\sqrt{(x_1+16)^2 + (x_2-4)^2} + 6\sqrt{(x_1-6)^2 + (x_2-5)^2} + 10\sqrt{(x_1-3)^2 + (x_2+9)^2} \\
\text{subject to} & \quad |x_1-1| + |x_2-1| \leq 2
\end{align*}
\]

The corresponding unconstrained optimization problem is:

\[
\theta(x, \mu) = f(x) - \mu \left( \frac{1}{|x_1-1| + |x_2-1|} \right)
\]

The barrier function method, coupled with the Powell method of unconstrained minimization and golden bracket and golden search method of one-dimensional search, is used to solve this problem.
clc;  % clears the screen
clear all;  % clears all values of variables for memory advantage.
global x µ V
x = [100; 100];  % initial starting point
µ = 100;beta = 0.1;
tol = 1.0e-3; tol1 = 1.0e-3; h = 0.1;N = 10;
if size(x,2) > 1; x = x'; end  % x must be column vector
n = length(x);    % Number of design variables
df = zeros(n,1);  % Decreases of f stored here
u = eye(n);       % Columns of u store search directions V
for k=1:N % loop for the penalty function method
    [c,ceq]=constrainedairport(x);
    obj=obairport(x);
    f=unconstrainedairport(x,µ);
    disp(sprintf('%1.5f       (%3.12f,%3.12f)       %2.10f
                  %2.10f ',µ,x,obj,f))
    for j = 1:30      % Allow up to 30 cycles for Powell's method
        xold = x;
        fold = feval(@unconstrainedlan,xold,µ);
        % First n line searches record the decrease of f
        for i = 1:n
            V = u(1:n,i);
            [a,b] = goldbracket(@(fline,x)@fline(x+V,µ),0.0,h);
            [s,fmin] = goldsearch(@(fline,a,b);
            df(i) = fold - fmin;
        end
        fold = fmin;
        x = x + s*V;
    end
    % Last line search in the cycle
    V = x - xold;
    [a,b] = goldbracket(@(fline,0,0,h),V);
    [s,fmin] = goldsearch(@(fline,a,b);
    x = x + s*V;
    if sqrt(dot(x-xold,x-xold)/n) < tol
        y = x;
    end
    % Identify biggest decrease of f & update search directions
    imax = 1; dfmax = df(1);
    for i = 2:n
        if df(i) > dfmax
            imax = i; dfmax = df(i);
        end
    end
    x = y;
    µ=beta*µ;
    sqrt(dot(f -obj,(f-obj));
    if sqrt(dot(f -obj, f -obj)) < tol1
        return
    end
end  % end of SUMT iteration.

function z = flines(s) % f in the search direction V
function [a,b] = goldbracket(func,x1,h)
function [s,fmin] = goldsearch(func,a,b);
% Start of golden bracketing for the minimum.

% output:
% a, b = limits on x at the minimum point.
c = 1.618033989;
f1 = feval(func,x1);
x2 = x1 + h; f2 = feval(func,x2);
% Determine downhill direction & change sign of h if needed.
if f2 > f1
    h = -h;
    x2 = x1 + h; f2 = feval(func,x2);
% Check if minimum is between x1 - h and x1 + h
if f2 > f1
    a = x2; b = x1 - h;
    return
end
% Search loop
for i = 1:100
    h = c*h;
    x3 = x2 + h; f3 = feval(func,x3);
    if f3 > f2
        a = x1; b = x3;
        return
    end
    x1 = x2; x2 = x3; f2 = f3;
end
error('goldbracket did not find a minimum please try another starting point')

% Start of golden search for the minimum.
function [xmin,fmin] = goldsearch(func,a,b,tol2)
% Golden section search for the minimum of f(x).
% The minimum point must be bracketed in a <= x <= b.
% usage: [fmin,xmin] = goldsearch(func,xstart,h)

% input:
% func = handle of function that returns f(x).
% a, b = limits of the interval containing the minimum.
% tol2 = error tolerance for goldsearch.
if nargin < 4; tol2 = 1.0e-6; end
nIter = ceil(-2.078087*log(tol2/abs(b-a)));
R = 0.618033989; % R is called golden ratio.
C = 1.0 - R;
% First telescoping
x1 = R*a + C*b;
x2 = C*a + R*b;
f1 = feval(func,x1);
f2 = feval(func,x2);
% Main loop
for i = 1:nIter
    if f1 > f2
        a = x1; x1 = x2; f1 = f2;
x2 = C*a + R*b;
f2 = feval(func,x2);
    else
        b = x2; x2 = x1; f2 = f1;
x1 = R*a + C*b;
f1 = feval(func,x1);
    end
end
if f1 < f2; fmin = f1; xmin = x1; else
    fmin = f2; xmin = x2;
end

REFERENCES