Chapter 6

NEW GRADIENT-BASED TRAJECTORY AND APPROXIMATION METHODS

6.1 Introduction

6.1.1 Why new algorithms?

In spite of the mathematical sophistication of classical *gradient-based* algorithms, certain *inhibiting difficulties remain* when these algorithms are applied to *real-world problems*. This is particularly true in the field of engineering, where unique difficulties occur that have prevented the general application of gradient-based mathematical optimization techniques to design problems.

Optimization difficulties that arise are:

(i) the *functions* are often *very expensive to evaluate*, requiring, for example, the time-consuming finite element analysis of a structure, the simulation of the dynamics of a multi-body system, or a

J.A. Snyman and D.N. Wilke, *Practical Mathematical Optimization*,
Springer Optimization and Its Applications 133,
https://doi.org/10.1007/978-3-319-77586-9_6

computational fluid dynamics (CFD) simulation,

(ii) the *existence of noise*, numerical or experimental, in the functions,

(iii) the presence of *discontinuities* in the functions,

(iv) *multiple local minima*, requiring global optimization techniques,

(v) the existence of regions in the design space where the *functions are not defined*, and

(vi) the occurrence of an *extremely large number* of design *variables*, disqualifying, for example, the SQP method.

6.1.2 Research at the University of Pretoria

All the above difficulties have been addressed in research done at the University of Pretoria over the past twenty years. This research has led to, amongst others, the development of the new optimization algorithms and methods listed in the subsections below.

6.1.2.1 Unconstrained optimization

(i) The *leap-frog* dynamic *trajectory* method: LFOP (Snyman 1982, 1983),

(ii) a *conjugate-gradient method* with Euler-trapezium steps in which a novel gradient-only line search method is used: ETOP (Snyman 1985), and

(iii) a *steepest-descent method* applied to successive spherical quadratic approximations: SQSD (Snyman and Hay 2001).

6.1.2.2 Direct constrained optimization

(i) The leap-frog method for constrained optimization, LFOPC (Snyman 2000), and

(ii) the conjugate-gradient method with Euler-trapezium steps and gradient-only line searches, applied to penalty function formulations of constrained problems: ETOPC (Snyman 2005).

6.1.2.3 Approximation methods

(i) A *feasible descent cone method* applied to successive spherical *quadratic sub-problems*: FDC-SAM (Stander and Snyman 1993; Snyman and Stander 1994, 1996; De Klerk and Snyman 1994), and

(ii) the *leap-frog method* (LFOPC) applied to successive spherical *quadratic sub-problems*: Dynamic-Q (Snyman et al. 1994; Snyman and Hay 2002).

6.1.2.4 Methods for global unconstrained optimization

(i) A *multi-start global minimization* algorithm with *dynamic search trajectories*: SF-GLOB (Snyman and Fatti 1987), and

(ii) a *modified bouncing ball* trajectory method for global optimization: MBB (Groenwold and Snyman 2002).

All of the above methods developed at the University of Pretoria are *gradient-based*, and have the common and unique property, for gradient-based methods, that *no explicit objective function line searches are required.*

In this chapter the LFOP/C unconstrained and constrained algorithms are discussed in detail. This is followed by the presentation of the SQSD method, which serves as an introduction to the Dynamic-Q approximation method. Next the ETOP/C algorithms are introduced, with special reference to their ability to deal with the presence of severe noise in the objective function, through the use of a gradient-only line search technique. Finally the SF-GLOB and MBB stochastic global optimization algorithms, which use dynamic search trajectories, are presented and discussed.

6.2 The dynamic trajectory optimization method

The dynamic trajectory method for unconstrained minimization (Snyman 1982, 1983) is also known as the "leap-frog" method. It has been modified (Snyman 2000) to handle constraints via a penalty function formulation of the constrained problem. The outstanding characteristics of the basic method are:

(i) it uses *only* function *gradient* information $\boldsymbol{\nabla} f$,

(ii) *no* explicit *line searches* are performed,

(iii) it is extremely *robust*, handling steep valleys, and discontinuities and noise in the objective function and its gradient vector, with relative ease,

(iv) the algorithm seeks relatively low local minima and can therefore be used as the basic component in a methodology for *global optimization*, and

(v) when applied to smooth and near quadratic functions, it is not as efficient as classical methods.

6.2.1 Basic dynamic model

Assume a particle of *unit mass* in a n-dimensional *conservative* force field with *potential energy* at $\mathbf{x}$ given by $f(\mathbf{x})$, then at $\mathbf{x}$ the force (acceleration $\mathbf{a}$) on the particle is given by:

$$\mathbf{a} = \ddot{\mathbf{x}} = -\boldsymbol{\nabla} f(\mathbf{x}) \tag{6.1}$$

from which it follows that for the motion of the particle over the time interval $[0, t]$:

$$\tfrac{1}{2}\|\dot{\mathbf{x}}(t)\|^2 - \tfrac{1}{2}\|\dot{\mathbf{x}}(0)\|^2 = f(\mathbf{x}(0)) - f(\mathbf{x}(t)) \tag{6.2}$$

or

$$T(t) - T(0) = f(0) - f(t) \tag{6.3}$$

where $T(t)$ represents the kinetic energy of the particle at time t. Thus it follows that

$$f(t) + T(t) = \text{ constant} \tag{6.4}$$

i.e. conservation of energy along the trajectory. Note that along the particle trajectory the change in the function f, $\Delta f = -\Delta T$, and therefore, as long as T increases, f decreases. This is the underlying principle on which the dynamic leap-frog optimization algorithm is based.

6.2.2 Basic algorithm for unconstrained problems (LFOP)

The basic elements of the LFOP method are as listed in Algorithm 6.1. A detailed flow chart of the basic LFOP algorithm for unconstrained problems is given in Figure 6.1.

Algorithm 6.1 LFOP algorithm

1. Given $f(\mathbf{x})$ and starting point $\mathbf{x}(0) = \mathbf{x}^0$, compute the dynamic trajectory of the particle by solving the *initial value problem*:

$$\begin{aligned} \ddot{\mathbf{x}}(t) &= -\boldsymbol{\nabla} f(\mathbf{x}(t)) \\ \text{with } \dot{\mathbf{x}}(0) &= \mathbf{0};\ \text{and } \mathbf{x}(0) = \mathbf{x}^0. \end{aligned} \tag{6.5}$$

2. Monitor $\mathbf{v}(t) = \dot{\mathbf{x}}(t)$, clearly as long as $T = \frac{1}{2}\|\mathbf{v}(t)\|^2$ increases, $f(\mathbf{x}(t))$ decreases as desired.

3. When $\|\mathbf{v}(t)\|$ decreases, i.e. when the particle moves uphill, apply some *interfering strategy* to gradually extract energy from the particle so as to increase the likelihood of its descent, but not so that descent occurs immediately.

4. In practice the numerical integration of the initial value problem (6.5) is done by the *"leap-frog" method*: compute for $k = 0, 1, 2, \dots$, and given time step Δt:

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathbf{x}^k + \mathbf{v}^k \Delta t \\ \mathbf{v}^{k+1} &= \mathbf{v}^k + \mathbf{a}^{k+1} \Delta t \end{aligned}$$

where

$$\mathbf{a}^k = -\boldsymbol{\nabla} f(\mathbf{x}^k) \text{ and } \mathbf{v}^0 = \tfrac{1}{2}\mathbf{a}^0 \Delta t,$$

to ensure an initial step if $\mathbf{a}^0 \neq \mathbf{0}$.

A typical *interfering strategy* is to continue the trajectory when $\|\mathbf{v}^{k+1}\| \geq \|\mathbf{v}^k\|$, otherwise set $\mathbf{v}^k = \frac{1}{4}(\mathbf{v}^{k+1} + \mathbf{v}^k)$, $\mathbf{x}^{k+1} = \frac{1}{2}(\mathbf{x}^{k+1} + \mathbf{x}^k)$ to compute the new $\mathbf{v}^{k+1}$ and then continue.

In addition, Snyman (1982, 1983) introduced additional *heuristics* to determine a suitable initial time step Δt, to allow for the magnification and reduction of Δt, and to control the magnitude of the step $\mathbf{\Delta x} = \mathbf{x}^{k+1} - \mathbf{x}^k$ by setting a step size limit δ along the computed trajectory. The recommended magnitude of δ is $\delta \approx \frac{1}{10}\sqrt{n} \times$ (maximum variable range).

6.2.3 Modification for constrained problems (LFOPC)

The code LFOPC (Snyman 2000) applies the unconstrained optimization algorithm LFOP to a penalty function formulation of the constrained problem (see Section 3.1) in 3 phases (see Algorithm 6.2).

Algorithm 6.2 LFOPC algorithm

Phase 0:
Given some $\mathbf{x}^0$, then with overall penalty parameter $\rho = \rho_0$, apply LFOP to the penalty function $P(\mathbf{x}, \rho_0)$ to give $\mathbf{x}^*(\rho_0)$.
Phase 1:
With $\mathbf{x}^0 := \mathbf{x}^*(\rho_0)$ and $\rho := \rho_1$, where $\rho_1 \gg \rho_0$, apply LFOP to $P(\mathbf{x}, \rho_1)$ to give $\mathbf{x}^*(\rho_1)$ and identify the set of *active* constraints I_a, such that $g_{i_a}(\mathbf{x}^*(\rho_1)) > 0$ for $i_a \in I_a$.
Phase 2:
With $\mathbf{x}^0 := \mathbf{x}^*(\rho_1)$ use LFOP to minimize

$$P_a(\mathbf{x}, \rho_1) = \sum_{i=1}^{r} \rho_1 h_i^2(\mathbf{x}) + \sum_{i_a \in I_a} \rho_1 g_{i_a}^2(\mathbf{x})$$

to give $\mathbf{x}^*$.

For engineering problems (with convergence tolerance $\varepsilon_x = 10^{-4}$) the choice $\rho_0 = 10$ and $\rho_1 = 100$ is recommended. For extreme accuracy ($\varepsilon_x = 10^{-8}$), use $\rho_0 = 100$ and $\rho_1 = 10^4$.

6.2.3.1 Example

Minimize $f(\mathbf{x}) = x_1^2 + 2x_2^2$ such that $g(\mathbf{x}) = -x_1 - x_2 + 1 \leq 0$ with starting point $\mathbf{x}^0 = [3, 1]^T$ by means of the LFOPC algorithm. Use $\rho_0 = 1.0$ and $\rho_1 = 10.0$. The computed solution is depicted in Figure 6.2.

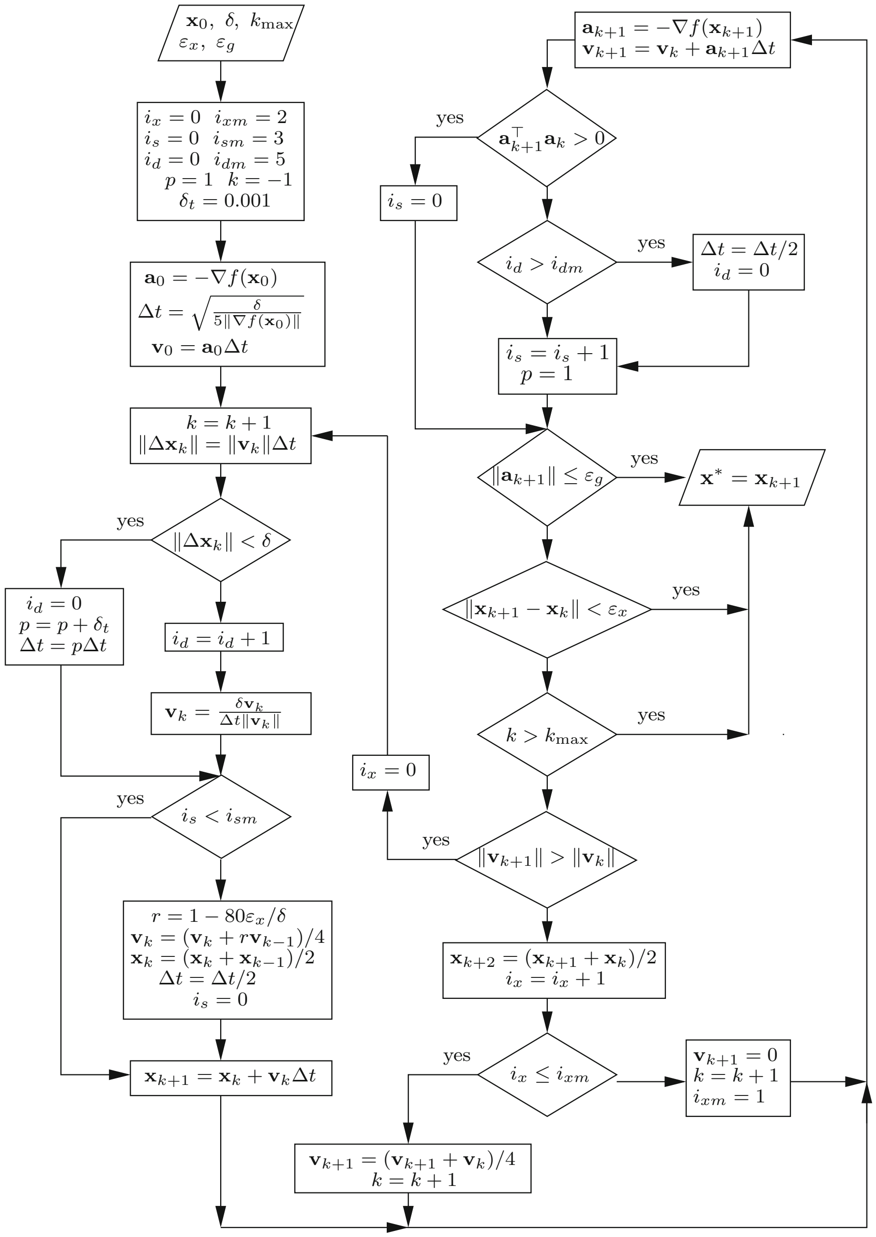
$\mathbf{x}_0,\ \delta,\ k_{\max}$ $\varepsilon_x,\ \varepsilon_g$
$i_x = 0\ \ i_{xm} = 2$ $i_s = 0\ \ i_{sm} = 3$ $i_d = 0\ \ i_{dm} = 5$ $p = 1\ \ k = -1$ $\delta_t = 0.001$
$\mathbf{a}_0 = -\nabla f(\mathbf{x}_0)$ $\Delta t = \sqrt{\frac{\delta}{5\|\nabla f(\mathbf{x}_0)\|}}$ $\mathbf{v}_0 = \mathbf{a}_0 \Delta t$
$k = k+1$ $\|\Delta \mathbf{x}_k\| = \|\mathbf{v}_k\|\Delta t$
$\|\Delta \mathbf{x}_k\| < \delta$
yes
$i_d = 0$ $p = p + \delta_t$ $\Delta t = p\Delta t$
$i_d = i_d + 1$
$\mathbf{v}_k = \frac{\delta \mathbf{v}_k}{\Delta t \|\mathbf{v}_k\|}$
$i_s < i_{sm}$
yes
$r = 1 - 80\varepsilon_x/\delta$ $\mathbf{v}_k = (\mathbf{v}_k + r\mathbf{v}_{k-1})/4$ $\mathbf{x}_k = (\mathbf{x}_k + \mathbf{x}_{k-1})/2$ $\Delta t = \Delta t/2$ $i_s = 0$
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k \Delta t$
$\mathbf{a}_{k+1} = -\nabla f(\mathbf{x}_{k+1})$ $\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{a}_{k+1}\Delta t$
$\mathbf{a}_{k+1}^\top \mathbf{a}_k > 0$
yes
$i_s = 0$
$i_d > i_{dm}$
yes
$\Delta t = \Delta t/2$ $i_d = 0$
$i_s = i_s + 1$ $p = 1$
$\|\mathbf{a}_{k+1}\| \le \varepsilon_g$
yes
$\mathbf{x}^* = \mathbf{x}_{k+1}$
$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_x$
yes
$k > k_{\max}$
yes
$i_x = 0$
$\|\mathbf{v}_{k+1}\| > \|\mathbf{v}_k\|$
yes
$\mathbf{x}_{k+2} = (\mathbf{x}_{k+1} + \mathbf{x}_k)/2$ $i_x = i_x + 1$
$i_x \le i_{xm}$
yes
$\mathbf{v}_{k+1} = 0$ $k = k+1$ $i_{xm} = 1$
$\mathbf{v}_{k+1} = (\mathbf{v}_{k+1} + \mathbf{v}_k)/4$ $k = k+1$

Figure 6.1: Flowchart of the LFOP unconstrained minimization algorithm

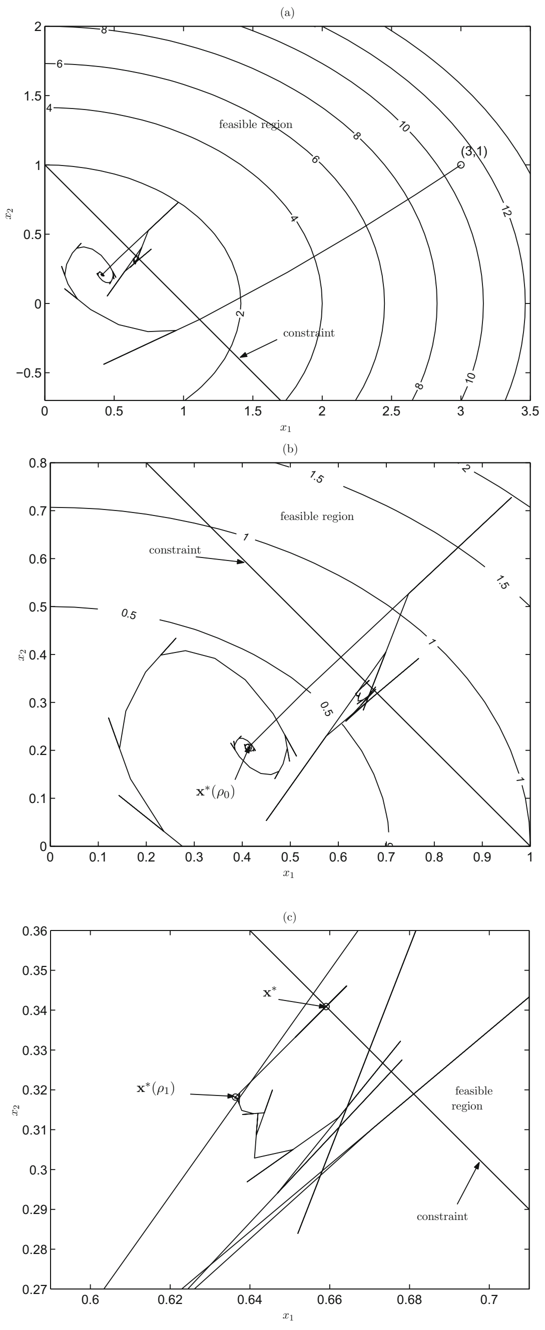

Figure 6.2: The (a) complete LFOPC trajectory for example problem 6.2.3.1, with $\mathbf{x}^0 = [3, 1]^T$, and magnified views of the final part of the trajectory shown in (b) and (c), giving $\mathbf{x}^* \approx [0.659, 0.341]^T$

6.3 The spherical quadratic steepest descent method

6.3.1 Introduction

In this section an extremely simple gradient only algorithm (Snyman and Hay 2001) is proposed that, in terms of storage requirement (only 3 n-vectors need be stored) and computational efficiency, may be considered as an alternative to the conjugate gradient methods. The method effectively applies the steepest descent (SD) method to successive simple spherical quadratic approximations of the objective function in such a way that no explicit line searches are performed in solving the minimization problem. It is shown that the method is convergent when applied to general positive-definite quadratic functions. The method is tested by its application to some standard and other test problems. On the evidence presented the new method, called the SQSD algorithm, appears to be reliable and stable and performs very well when applied to extremely ill-conditioned problems.

6.3.2 Classical steepest descent method revisited

Consider the following unconstrained optimization problem:

$$\min f(\mathbf{x}),\ \ \mathbf{x} \in \mathbb{R}^n \tag{6.6}$$

where f is a scalar objective function defined on $\mathbb{R}^n$, the n-dimensional real Euclidean space, and $\mathbf{x}$ is a vector of n real components $x_1, x_2, \dots, x_n$. It is assumed that f is differentiable so that the gradient vector $\boldsymbol{\nabla} f(\mathbf{x})$ exists everywhere in $\mathbb{R}^n$. The solution is denoted by $\mathbf{x}^*$.

The steepest descent (SD) algorithm for solving problem (6.6) may then be stated as follows:

It can be shown that if the steepest descent method is applied to a general positive-definite quadratic function of the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$, then the sequence $\left\{f(\mathbf{x}^k)\right\} \rightarrow f(\mathbf{x}^*)$. Depending, however, on the starting point $\mathbf{x}^0$ and the condition number of $\mathbf{A}$ associated with the quadratic form, the rate of convergence may become extremely slow.

Algorithm 6.3 SD algorithm

Initialization: Specify convergence tolerances ε_g and ε_x, select starting point $\mathbf{x}^0$. Set $k := 1$ and go to main procedure.
Main procedure:

```
1. If $\|\nabla f(\mathbf{x}^{k-1})\| < \varepsilon_g$, then set $\mathbf{x}^* \cong \mathbf{x}^c = \mathbf{x}^{k-1}$ and stop; otherwise
   set $\mathbf{u}^k := -\nabla f(\mathbf{x}^{k-1})$.
2. Let $\lambda_k$ be such that $f(\mathbf{x}^{k-1} + \lambda_k \mathbf{u}^k) = \min_\lambda f(\mathbf{x}^{k-1} + \lambda \mathbf{u}^k)$ subject
   to $\lambda \geq 0$ {line search step}.
3. Set $\mathbf{x}^k := \mathbf{x}^{k-1} + \lambda_k \mathbf{u}^k$; if $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| < \varepsilon_x$, then $\mathbf{x}^* \cong \mathbf{x}^c = \mathbf{x}^k$
   and stop; otherwise set $k := k + 1$ and go to Step 1.
```

It is proposed here that for general functions $f(\mathbf{x})$, better overall performance of the steepest descent method may be obtained by applying it successively to a sequence of very simple quadratic approximations of $f(\mathbf{x})$. The proposed modification, named here the spherical quadratic steepest descent (SQSD) method, remains a first order method since only gradient information is used with no attempt being made to construct the Hessian of the function. The storage requirements therefore remain minimal, making it ideally suitable for problems with a large number of variables. Another significant characteristic is that the method requires no explicit line searches.

6.3.3 The SQSD algorithm

In the SQSD approach, given an initial approximate solution $\mathbf{x}^0$, a sequence of spherically quadratic optimization subproblems $P[k], k = 0, 1, 2, \dots$ is solved, generating a sequence of approximate solutions $\mathbf{x}^{k+1}$. More specifically, at each point $\mathbf{x}^k$ the constructed approximate subproblem is $P[k]$:

$$\min_{\mathbf{x}} \tilde{f}_k(\mathbf{x}) \tag{6.7}$$

where the approximate objective function $\tilde{f}_k(\mathbf{x})$ is given by

$$\tilde{f}_k(\mathbf{x}) = f(\mathbf{x}^k) + \nabla^T f(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k)^T \mathbf{C}_k (\mathbf{x} - \mathbf{x}^k) \tag{6.8}$$

and $\mathbf{C}_k = \text{diag}(c_k, c_k, \ldots, c_k) = c_k\mathbf{I}$. The solution to this problem will be denoted by $\mathbf{x}^{*k}$, and for the construction of the next subproblem $P[k+1]$, $\mathbf{x}^{k+1} := \mathbf{x}^{*k}$.

For the first subproblem the curvature c_0 is set to $c_0 := \left\|\boldsymbol{\nabla} f(\mathbf{x}^0)\right\| / \rho$, where $\rho > 0$ is some arbitrarily specified step limit. Thereafter, for $k \geq 1$, c_k is chosen such that $\tilde{f}(\mathbf{x}^k)$ interpolates $f(\mathbf{x})$ at both $\mathbf{x}^k$ and $\mathbf{x}^{k-1}$. The latter conditions imply that for $k = 1, 2, \ldots$

$$c_k := \frac{2\left[f(\mathbf{x}^{k-1}) - f(\mathbf{x}^k) - \boldsymbol{\nabla}^T f(\mathbf{x}^k)(\mathbf{x}^{k-1} - \mathbf{x}^k)\right]}{\left\|\mathbf{x}^{k-1} - \mathbf{x}^k\right\|^2}. \tag{6.9}$$

Clearly the identical curvature entries along the diagonal of the Hessian, mean that the level surfaces of the quadratic approximation $\tilde{f}_k(\mathbf{x})$, are indeed concentric hyper-spheres. The approximate subproblems $P[k]$ are therefore aptly referred to as spherical quadratic approximations.

It is now proposed that for a large class of problems the sequence $\mathbf{x}^0, \mathbf{x}^1, \ldots$ will tend to the solution of the original problem (6.6), i.e.

$$\lim_{k \to \infty} \mathbf{x}^k = \mathbf{x}^*. \tag{6.10}$$

For subproblems $P[k]$ that are convex, i.e. $c_k > 0$, the solution occurs where $\boldsymbol{\nabla} \tilde{f}_k(\mathbf{x}) = \mathbf{0}$, that is where

$$\boldsymbol{\nabla} f(\mathbf{x}^k) + c_k\mathbf{I}(\mathbf{x} - \mathbf{x}^k) = \mathbf{0}. \tag{6.11}$$

The solution to the subproblem, $\mathbf{x}^{*k}$ is therefore given by

$$\mathbf{x}^{*k} = \mathbf{x}^k - \frac{\boldsymbol{\nabla} f(\mathbf{x}^k)}{c_k}. \tag{6.12}$$

Clearly the solution to the spherical quadratic subproblem lies along a line through $\mathbf{x}^k$ in the direction of steepest descent. The SQSD method may formally be stated in the form given in Algorithm 6.4.

Step size control is introduced in Algorithm 6.4 through the specification of a step limit ρ and the test for $\left\|\mathbf{x}^k - \mathbf{x}^{k-1}\right\| > \rho$ in Step 2 of the main procedure. Note that the choice of c_0 ensures that for $P[0]$ the solution $\mathbf{x}^1$ lies at a distance ρ from $\mathbf{x}^0$ in the direction of steepest descent. Also the test in Step 3 that $c_k < 0$, and setting $c_k := 10^{-60}$ where this condition is true ensures that the approximate objective function is always positive-definite.

Algorithm 6.4 SQSD algorithm

Initialization: Specify convergence tolerances ε_g and ε_x, step limit $\rho > 0$ and select starting point $\mathbf{x}^0$. Set $c_0 := \left\|\boldsymbol{\nabla} f(\mathbf{x}^0)\right\| / \rho$. Set $k := 1$ and go to main procedure.
Main procedure:

1. If $\left\|\boldsymbol{\nabla} f(\mathbf{x}^{k-1})\right\| < \varepsilon_g$, then $\mathbf{x}^* \cong \mathbf{x}^c = \mathbf{x}^{k-1}$ and stop; otherwise set

$$\mathbf{x}^k := \mathbf{x}^{k-1} - \frac{\boldsymbol{\nabla} f(\mathbf{x}^{k-1})}{c_{k-1}}.$$

2. If $\left\|\mathbf{x}^k - \mathbf{x}^{k-1}\right\| > \rho$, then set

$$\mathbf{x}^k := \mathbf{x}^k - \rho \frac{\boldsymbol{\nabla} f(\mathbf{x}^{k-1})}{\|\boldsymbol{\nabla} f(\mathbf{x}^{k-1})\|};$$

   if $\left\|\mathbf{x}^k - \mathbf{x}^{k-1}\right\| < \varepsilon_x$, then $\mathbf{x}^* \cong \mathbf{x}^c = \mathbf{x}^k$ and stop.

3. Set

$$c_k := \frac{2\left[f(\mathbf{x}^{k-1}) - f(\mathbf{x}^k) - \boldsymbol{\nabla}^T f(\mathbf{x}^k)(\mathbf{x}^{k-1} - \mathbf{x}^k)\right]}{\|\mathbf{x}^{k-1} - \mathbf{x}^k\|^2};$$

   if $c_k < 0$ set $c_k := 10^{-60}$.

4. Set $k := k + 1$ and go to Step 1 for next iteration.

6.3.4 Convergence of the SQSD method

An analysis of the convergence rate of the SQSD method, when applied to a general positive-definite quadratic function, affords insight into the convergence behavior of the method when applied to more general functions. This is so because for a large class of continuously differentiable functions, the behavior close to local minima is quadratic. For quadratic functions the following theorem may be proved.

6.3.4.1 Theorem

The SQSD algorithm (without step size control) is convergent when applied to the general quadratic function of the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}$, *where* $\mathbf{A}$ *is a* $n \times n$ *positive-definite matrix and* $\mathbf{b} \in \mathbb{R}^n$.

Proof. Begin by considering the bivariate quadratic function, $f(\mathbf{x}) = x_1^2 + \gamma x_2^2$, $\gamma \geq 1$ and with $\mathbf{x}^0 = [\alpha, \beta]^T$. Assume $c_0 > 0$ given, and for convenience in what follows set $c_0 = 1/\delta, \delta > 0$. Also employ the notation $f_k = f(\mathbf{x}^k)$.

Application of the first step of the SQSD algorithm yields

$$\mathbf{x}^1 = \mathbf{x}^0 - \frac{\boldsymbol{\nabla} f_0}{c_0} = [\alpha(1-2\delta), \beta(1-2\gamma\delta)]^T \tag{6.13}$$

and it follows that

$$\left\|\mathbf{x}^1 - \mathbf{x}^0\right\|^2 = 4\delta^2(\alpha^2 + \gamma^2\beta^2) \tag{6.14}$$

and

$$\boldsymbol{\nabla} f_1 = [2\alpha(1-2\delta), 2\gamma\beta(1-2\gamma\delta)]^T. \tag{6.15}$$

For the next iteration the curvature is given by

$$c_1 = \frac{2[f_0 - f_1 - \boldsymbol{\nabla}^T f_1(\mathbf{x}^0 - \mathbf{x}^1)]}{\|\mathbf{x}^0 - \mathbf{x}^1\|^2}. \tag{6.16}$$

Utilizing the information contained in (6.13)–(6.15), the various entries in expression (6.16) are known, and after substitution c_1 simplifies to

$$c_1 = \frac{2(\alpha^2 + \gamma^3\beta^2)}{\alpha^2 + \gamma^2\beta^2}. \tag{6.17}$$

In the next iteration, Step 1 gives

$$\mathbf{x}^2 = \mathbf{x}^1 - \frac{\boldsymbol{\nabla} f_1}{c_1}. \tag{6.18}$$

And after the necessary substitutions for $\mathbf{x}^1$, $\boldsymbol{\nabla} f_1$ and c_1, given by (6.13), (6.15) and (6.17) respectively, (6.18) reduces to

$$\mathbf{x}^2 = [\alpha(1-2\delta)\mu_1, \beta(1-2\gamma\delta)\omega_1]^T \tag{6.19}$$

where

$$\mu_1 = 1 - \frac{1+\gamma^2\beta^2/\alpha^2}{1+\gamma^3\beta^2/\alpha^2} \tag{6.20}$$

and

$$\omega_1 = 1 - \frac{\gamma+\gamma^3\beta^2/\alpha^2}{1+\gamma^3\beta^2/\alpha^2}. \tag{6.21}$$

Clearly if $\gamma = 1$, then $\mu_1 = 0$ and $\omega_1 = 0$. Thus by (6.19) $\mathbf{x}^2 = \mathbf{0}$ and convergence to the solution is achieved within the second iteration.

Now for $\gamma > 1$, and for any choice of α and β, it follows from (6.20) that

$$0 \leq \mu_1 < 1 \tag{6.22}$$

which implies from (6.19) that for the first component of $\mathbf{x}^2$:

$$\left|x_1^{(2)}\right| = |\alpha(1-2\delta)\mu_1| < |\alpha(1-2\delta)| = \left|x_1^{(1)}\right| \tag{6.23}$$

or introducing α notation (with $\alpha_0 = \alpha$), that

$$|\alpha_2| = |\mu_1\alpha_1| < |\alpha_1|. \tag{6.24}$$

{Note: because $c_0 = 1/\delta > 0$ is chosen arbitrarily, it cannot be said that $|\alpha_1| < |\alpha_0|$. However α_1 is finite.}

The above argument, culminating in result (6.24), is for the two iterations $\mathbf{x}^0 \rightarrow \mathbf{x}^1 \rightarrow \mathbf{x}^2$. Repeating the argument for the sequence of overlapping pairs of iterations $\mathbf{x}^1 \rightarrow \mathbf{x}^2 \rightarrow \mathbf{x}^3$; $\mathbf{x}^2 \rightarrow \mathbf{x}^3 \rightarrow \mathbf{x}^4; \ldots$, it follows similarly that $|\alpha_3| = |\mu_2\alpha_2| < |\alpha_2|$; $|\alpha_4| = |\mu_3\alpha_3| < |\alpha_3|; \ldots$, since $0 \leq \mu_2 < 1$; $0 \leq \mu_3 < 1; \ldots$, where the value of μ_k is determined by (corresponding to equation (6.20) for μ_1):

$$\mu_k = 1 - \frac{1+\gamma^2\beta_{k-1}^2/\alpha_{k-1}^2}{1+\gamma^3\beta_{k-1}^2/\alpha_{k-1}^2}. \tag{6.25}$$

Thus in general

$$0 \leq \mu_k < 1 \tag{6.26}$$

and

$$|\alpha_{k+1}| = |\mu_k \alpha_k| < |\alpha_k|. \tag{6.27}$$

For large positive integer m it follows that

$$|\alpha_m| = |\mu_{m-1}\alpha_{m-1}| = |\mu_{m-1}\mu_{m-2}\alpha_{m-2}| = |\mu_{m-1}\mu_{m-2}\cdots\mu_1\alpha_1| \tag{6.28}$$

and clearly for $\gamma > 0$, because of (6.26)

$$\lim_{m\to\infty} |\alpha_m| = 0. \tag{6.29}$$

Now for the second component of $\mathbf{x}^2$ in (6.19), the expression for ω_1, given by (6.21), may be simplified to

$$\omega_1 = \frac{1-\gamma}{1+\gamma^3\beta^2/\alpha^2}. \tag{6.30}$$

Also for the second component:

$$x_2^{(2)} = \beta(1-2\gamma\delta)\omega_1 = \omega_1 x_2^{(1)} \tag{6.31}$$

or introducing β notation

$$\beta_2 = \omega_1\beta_1. \tag{6.32}$$

The above argument is for $\mathbf{x}^0 \to \mathbf{x}^1 \to \mathbf{x}^2$ and again, repeating it for the sequence of overlapping pairs of iterations, it follows more generally for $k = 1, 2, \ldots$, that

$$\beta_{k+1} = \omega_k\beta_k \tag{6.33}$$

where ω_k is given by

$$\omega_k = \frac{1-\gamma}{1+\gamma^3\beta_{k-1}^2/\alpha_{k-1}^2}. \tag{6.34}$$

Since by (6.29), $|\alpha_m| \to 0$, it follows that if $|\beta_m| \to 0$ as $m \to \infty$, the theorem is proved for the bivariate case. Make the assumption that $|\beta_m|$

does not tend to zero, then there exists a finite positive number ε such that

$$|\beta_k| \geq \varepsilon \tag{6.35}$$

for all k. This allows the following argument:

$$|\omega_k| = \left|\frac{1-\gamma}{1+\gamma^3\beta_{k-1}^2/\alpha_{k-1}^2}\right| \leq \left|\frac{1-\gamma}{1+\gamma^3\varepsilon^2/\alpha_{k-1}^2}\right| = \left|\frac{(1-\gamma)\alpha_{k-1}^2}{\alpha_{k-1}^2+\gamma^3\varepsilon^2}\right|. \tag{6.36}$$

Clearly since by (6.29) $|\alpha_m| \to 0$ as $m \to \infty$, (6.36) implies that also $|\omega_m| \to 0$. This result taken together with (6.33) means that $|\beta_m| \to 0$ which contradicts the assumption above. With this result the theorem is proved for the bivariate case.

Although the algebra becomes more complicated, the above argument can clearly be extended to prove convergence for the multivariate case, where

$$f(\mathbf{x}) = \sum_{i=1}^{n} \gamma_i x_i^2,\ \gamma_1 = 1 \leq \gamma_2 \leq \gamma_3 \leq \cdots \leq \gamma_n. \tag{6.37}$$

Finally since the general quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x},\ \mathbf{A}\ \text{positive} - \text{definite} \tag{6.38}$$

may be transformed to the form (6.37), convergence of the SQSD method is also ensured in the general case. □

It is important to note that, the above analysis does not prove that $\|\mathbf{x}^k - \mathbf{x}^*\|$ is monotonically decreasing with k, neither does it necessarily follow that monotonic descent in the corresponding objective function values $f(\mathbf{x}^k)$, is guaranteed. Indeed, extensive numerical experimentation with quadratic functions show that, although the SQSD trajectory rapidly approaches the minimum, relatively large spike increases in $f(\mathbf{x}^k)$ may occur after which the trajectory quickly recovers on its path to $\mathbf{x}^*$. This happens especially if the function is highly elliptical (poorly scaled).

6.3.5 Numerical results and conclusion

The SQSD method is now demonstrated by its application to some test problems. For comparison purposes the results are also given for the standard SD method and both the Fletcher-Reeves (FR) and Polak-Ribiere (PR) conjugate gradient methods. The latter two methods are implemented using the CG+ FORTRAN conjugate gradient program of Gilbert and Nocedal (1992). The CG+ implementation uses the line search routine of Moré and Thuente (1994). The function and gradient values are evaluated together in a single subroutine. The SD method is applied using CG+ with the search direction modified to the steepest descent direction. The FORTRAN programs were run on a 266 MHz Pentium 2 computer using double precision computations.

The standard (refs. Rao 1996; Snyman 1985; Himmelblau 1972; Manevich 1999) and other test problems used are listed in Section 6.3.6 and the results are given in Tables 6.1 and 6.2. The convergence tolerances applied throughout are $\varepsilon_g = 10^{-5}$ and $\varepsilon_x = 10^{-8}$, except for the extended homogeneous quadratic function with $n = 50000$ (Problem 12) and the extremely ill-conditioned Manevich functions (Problems 14). For these problems the extreme tolerances $\varepsilon_g \cong 0 (= 10^{-75})$ and $\varepsilon_x = 10^{-12}$, are prescribed in an effort to ensure very high accuracy in the approximation $\mathbf{x}^c$ to $\mathbf{x}^*$. For each method the number of function-cum-gradient-vector evaluations (N^{fg}) are given. For the SQSD method the number of iterations is the same as N^{fg}. For the other methods the number of iterations (N^{it}) required for convergence, and which corresponds to the number of line searches executed, are also listed separately. In addition the relative error (E^r) in optimum function value, defined by

$$E^r = \left| \frac{f(\mathbf{x}^*) - f(\mathbf{x}^c)}{1 + |f(\mathbf{x}^*)|} \right| \tag{6.39}$$

where $\mathbf{x}^c$ is the approximation to $\mathbf{x}^*$ at convergence, is also listed. For the Manevich problems, with $n \geq 40$, for which the other (SD, FR and PR) algorithms fail to converge after the indicated number of steps, the infinite norm of the error in the solution vector (I^∞), defined by $\|\mathbf{x}^* - \mathbf{x}^c\|_\infty$ is also tabulated. These entries, given instead of the relative error in function value (E^r), are made in italics.

Inspection of the results shows that the SQSD algorithm is consistently competitive with the other three methods and performs notably well for large problems. Of all the methods the SQSD method appears to be the most reliable one in solving each of the posed problems. As expected, because line searches are eliminated and consecutive search directions are

Prob. #	n	SQSD			Steepest Descent		
		ρ	N^{fg}	E^r	N^{fg}	N^{it}	E^r/I^∞
1	3	1	12	3.E-14	41	20	6.E-12
2	2	1	31	1.E-14	266	131	9.E-11
3	2	1	33	3.E-08	2316	1157	4.E-08
4	2	0.3	97	1.E-15	$>$ 20000		3.E-09
5(a)	3	1	11	1.E-12	60	29	6.E-08
5(b)	3	1	17	1.E-12	49	23	6.E-08
6	4	1	119	9.E-09	$>$ 20000		2.E-06
7	3	1	37	1.E-12	156	77	3.E-11
8	2	10	39	1.E-22	12050*	6023*	26*
9	2	0.3	113	5.E-14	6065	3027	2.E-10
10	2	1	43	1.E-12	1309	652	1.E-10
11	4	2	267	2.E-11	16701	8348	4.E-11
12	20	1.E+04	58	1.E-11	276	137	1.E-11
	200	1.E+04	146	4.E-12	2717	1357	1.E-11
	2000	1.E+04	456	2.E-10	$>$ 20000		2.E-08
	20000	1.E+04	1318	6.E-09	$>$ 10000		8.E+01
	50000	1.E+10	4073	3.E-16	$>$ 10000		5.E+02
13	10	0.3	788	2.E-10	$>$ 20000		4.E-07
	100	1	2580	1.E-12	$>$ 20000		3.E+01
	300	1.73	6618	1.E-10	$>$ 20000		2.E+02
	600	2.45	13347	1.E-11	$>$ 20000		5.E+02
	1000	3.16	20717	2.E-10	$>$ 30000		9.E+02
14	20	1	3651	2.E-27	$>$ 20000		9.E-01
		10	3301	9.E-30			
	40	1	13302	5.E-27	$>$ 30000		*1.E+00*
		10	15109	2.E-33			
	60	1	19016	7.E-39	$>$ 30000		*1.E+00*
		10	16023	6.E-39			
	100	1	39690	1.E-49	$>$ 50000		*1.E+00*
		10	38929	3.E-53			
	200	1	73517	5.E-81	$>$ 100000		*1.E+00*
		10	76621	4.E-81			

* Convergence to a local minimum with $f(\mathbf{x}^c) = 48.9$.

Table 6.1: Performance of the SQSD and SD optimization algorithms when applied to the test problems listed in Section 6.3.6

Prob. #	n	Fletcher-Reeves			Polak-Ribiere		
		N^{fg}	N^{it}	E^r/I^∞	N^{fg}	N^{it}	E^r/I^∞
1	3	7	3	0$	7	3	0$
2	2	30	11	2.E-11	22	8	2.E-12
3	2	45	18	2.E-08	36	14	6.E-11
4	2	180	78	1.E-11	66	18	1.E-14
5(a)	3	18	7	6.E-08	18	8	6.E-08
5(b)	3	65	31	6.E-08	26	11	6.E-08
6	4	1573	783	8.E-10	166	68	3.E-09
7	3	132	62	4.E-12	57	26	1.E-12
8	2	72*	27*	26*	24*	11*	26*
9	2	56	18	5.E-11	50	17	1.E-15
10	2	127	60	6.E-12	30	11	1.E-11
11	4	193	91	1.E-12	99	39	9.E-14
12	20	42	20	9.E-32	42	20	4.E-31
	200	163	80	5.E-13	163	80	5.E-13
	2000	530	263	2.E-13	530	263	2.E-13
	20000	1652	825	4.E-13	1652	825	4.E-13
	50000	3225	1161	1.E-20	3225	1611	1.E-20
13	10	> 20000		2.E-02	548	263	4.E-12
	100	> 20000		8.E+01	1571	776	2.E-12
	300	> 20000		3.E+02	3253	1605	2.E-12
	600	> 20000		6.E+02	5550	2765	2.E-12
	1000	> 30000		1.E+03	8735	4358	2.E-12
14	20	187	75	8.E-24	1088	507	2.E-22
	40	> 30000		*1.E+00*	> 30000		*1.E+00*
	60	> 30000		*1.E+00*	> 30000		*1.E+00*
	100	> 50000		*1.E+00*	> 50000		*1.E+00*
	200	> 100000		*1.E+00*	> 100000		*1.E+00*

* Convergence to a local minimum with $f(\mathbf{x}^c) = 48.9$; $ Solution to machine accuracy.

Table 6.2: Performance of the FR and PR algorithms when applied to the test problems listed in Section 6.3.6

no longer forced to be orthogonal, the new method completely overshadows the standard SD method. What is much more gratifying, however, is the performance of the SQSD method relative to the well-established and well-researched conjugate gradient algorithms. Overall the new method appears to be very competitive with respect to computational efficiency and, on the evidence presented, remarkably stable.

In the implementation of the SQSD method to highly non-quadratic and non-convex functions, some care must however be taken in ensuring that the chosen step limit parameter ρ, is not too large. A too large value may result in excessive oscillations occurring before convergence. Therefore a relatively small value, $\rho = 0.3$, was used for the Rosenbrock problem with $n = 2$ (Problem 4). For the extended Rosenbrock functions of larger dimensionality (Problems 13), correspondingly larger step limit values ($\rho = \sqrt{n}/10$) were used with success.

For quadratic functions, as is evident from the convergence analysis of Section 6.3.4, no step limit is required for convergence. This is borne out in practice by the results for the extended homogeneous quadratic functions (Problems 12), where the very large value $\rho = 10^4$ was used throughout, with the even more extreme value of $\rho = 10^{10}$ for $n = 50000$. The specification of a step limit in the quadratic case also appears to have little effect on the convergence rate, as can be seen from the results for the ill-conditioned Manevich functions (Problems 14), that are given for both $\rho = 1$ and $\rho = 10$. Here convergence is obtained to at least 11 significant figures accuracy ($\|\mathbf{x}^* - \mathbf{x}^c\|_\infty < 10^{-11}$) for each of the variables, despite the occurrence of extreme condition numbers, such as 10^{60} for the Manevich problem with $n = 200$.

The successful application of the new method to the ill-conditioned Manevich problems, and the analysis of the convergence behavior for quadratic functions, indicate that the SQSD algorithm represents a powerful approach to solving quadratic problems with large numbers of variables. In particular, the SQSD method can be seen as an *unconditionally convergent*, *stable* and *economic* alternative iterative method for solving large systems of linear equations, ill-conditioned or not, through the minimization of the sum of the squares of the residuals of the equations.

6.3.6 Test functions used for SQSD

Minimize $f(\mathbf{x})$:

1. $f(\mathbf{x}) = x_1^2 + 2x_2^2 + 3x_3^2 - 2x_1 - 4x_2 - 6x_3 + 6,\ \mathbf{x}^0 = [3,3,3]^T,\ \mathbf{x}^* = [1,1,1]^T,\ f(\mathbf{x}^*) = 0.0.$

2. $f(\mathbf{x}) = x_1^4 - 2x_1^2 x_2 + x_1^2 + x_2^2 - 2x_1 + 1,\ \mathbf{x}^0 = [3,3]^T,\ \mathbf{x}^* = [1,1]^T,\ f(\mathbf{x}^*) = 0.0.$

3. $f(\mathbf{x}) = x_1^4 - 8x_1^3 + 25x_1^2 + 4x_2^2 - 4x_1x_2 - 32x_1 + 16,\ \mathbf{x}^0 = [3,3]^T,\ \mathbf{x}^* = [2,1]^T,\ f(\mathbf{x}^*) = 0.0.$

4. $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,\ \mathbf{x}^0 = [-1.2,1]^T,\ \mathbf{x}^* = [1,1]^T,\ f(\mathbf{x}^*) = 0.0$ (Rosenbrock's parabolic valley, Rao 1996).

5. $f(\mathbf{x}) = x_1^4 + x_1^3 - x_1 + x_2^4 - x_2^2 + x_2 + x_3^2 - x_3 + x_1x_2x_3$, (Zlobec's function, Snyman 1985):

   (a) $\mathbf{x}^0 = [1,-1,1]^T$ and

   (b) $\mathbf{x}^0 = [0,0,0]^T,\ \mathbf{x}^* = [0.57085597, -0.93955591, 0.76817555]^T,$ $f(\mathbf{x}^*) = -1.91177218907.$

6. $f(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,\ \mathbf{x}^0 = [3,-1,0,1]^T,\ \mathbf{x}^* = [0,0,0,0]^T,\ f(\mathbf{x}^*) = 0.0$ (Powell's quartic function, Rao 1996).

7. $f(\mathbf{x}) = -\left\{\frac{1}{1+(x_1-x_2)^2} + \sin\left(\frac{1}{2}\pi x_2 x_3\right) + \exp\left[-\left(\frac{x_1+x_3}{x_2} - 2\right)^2\right]\right\},$ $\mathbf{x}^0 = [0,1,2]^T, \mathbf{x}^* = [1,1,1]^T,\ f(\mathbf{x}^*) = -3.0$ (Rao 1996).

8. $f(\mathbf{x}) = \{-13 + x_1 + [(5 - x_2)x_2 - 2]x_2\}^2 + \{-29 + x_1 + [(x_2 + 1)x_2 - 14]x_2\}^2,\ \mathbf{x}^0 = [1/2,-2]^T,\ \mathbf{x}^* = [5,4]^T, f(\mathbf{x}^*) = 0.0$ (Freudenstein and Roth function, Rao 1996).

9. $f(\mathbf{x}) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2,\ \mathbf{x}^0 = [-1.2,1]^T,\ \mathbf{x}^* = [1,1]^T,$ $f(\mathbf{x}^*) = 0.0$ (cubic valley, Himmelblau 1972).

10. $f(\mathbf{x}) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2,\ \mathbf{x}^0 = [1,1]^T,\ \mathbf{x}^* = [3,1/2]^T,\ f(\mathbf{x}^*) = 0.0$ (Beale's function, Rao 1996).

11. $f(\mathbf{x}) = [10(x_2 - x_1^2)]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2$, $\mathbf{x}^0 = [-3, 1, -3, -1]^T$, $\mathbf{x}^* = [1, 1, 1, 1]^T$, $f(\mathbf{x}^*) = 0.0$ (Wood's function, Rao 1996).

12. $f(\mathbf{x}) = \sum_{i=1}^n ix_i^2$, $\mathbf{x}^0 = [3, 3, \dots, 3]^T$, $\mathbf{x}^* = [0, 0, \dots, 0]^T$, $f(\mathbf{x}^*) = 0.0$ (extended homogeneous quadratic functions).

13. $f(\mathbf{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$, $\mathbf{x}^0 = [-1.2, 1, -1.2, 1, \dots]^T$, $\mathbf{x}^* = [1, 1, \dots, 1]^T$, $f(\mathbf{x}^*) = 0.0$ (extended Rosenbrock functions, Rao 1996).

14. $f(\mathbf{x}) = \sum_{i=1}^n (1 - x_i)^2/2^{i-1}$, $\mathbf{x}^0 = [0, 0, \dots, 0]^T$, $\mathbf{x}^* = [1, 1, \dots, 1]^T$, $f(\mathbf{x}^*) = 0.0$ (extended Manevich functions, Manevich 1999).

6.4 The Dynamic-Q optimization algorithm

6.4.1 Introduction

An efficient *constrained* optimization method is presented in this section. The method, called the Dynamic-Q method (Snyman and Hay 2002), consists of applying the dynamic trajectory LFOPC optimization algorithm (see Section 6.2) to successive quadratic approximations of the actual optimization problem. This method may be considered as an extension of the unconstrained SQSD method, presented in Section 6.3, to one capable of handling general constrained optimization problems.

Due to its efficiency with respect to the number of function evaluations required for convergence, the Dynamic-Q method is primarily intended for optimization problems where function evaluations are expensive. Such problems occur frequently in engineering applications where time consuming numerical simulations may be used for function evaluations. Amongst others, these numerical analyses may take the form of a computational fluid dynamics (CFD) simulation, a structural analysis by means of the finite element method (FEM) or a dynamic simulation of a multibody system. Because these simulations are usually expensive to perform, and because the relevant functions may not be known analytically, standard classical optimization methods are normally not suited to these types of problems. Also, as will be shown, the storage requirements of the Dynamic-Q method are minimal. No Hessian information is required. The method is therefore particularly suitable for problems where the number of variables n is large.

6.4.2 The Dynamic-Q method

Consider the general nonlinear optimization problem:

$$\begin{gathered}\min_{\mathbf{x}} f(\mathbf{x});\ \mathbf{x} = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n \\ \text{subject to} \\ g_j(\mathbf{x}) \leq \mathbf{0};\ j = 1, 2, \dots, p \\ h_k(\mathbf{x}) = \mathbf{0};\ k = 1, 2, \dots, q\end{gathered} \tag{6.40}$$

where $f(\mathbf{x})$, $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ are scalar functions of $\mathbf{x}$.

In the Dynamic-Q approach, successive subproblems $P[i],\ i = 0, 1, 2, \dots$ are generated, at successive approximations $\mathbf{x}^i$ to the solution $\mathbf{x}^*$, by constructing *spherically quadratic* approximations $\tilde{f}(\mathbf{x})$, $\tilde{g}_j(\mathbf{x})$ and $\tilde{h}_k(\mathbf{x})$ to $f(\mathbf{x})$, $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$. These approximation functions, evaluated at a point $\mathbf{x}^i$, are given by

$$\begin{aligned}\tilde{f}(\mathbf{x}) &= f(\mathbf{x}^i) + \boldsymbol{\nabla}^T f(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^i)^T \mathbf{A}(\mathbf{x} - \mathbf{x}^i) \\ \tilde{g}_j(\mathbf{x}) &= g_j(\mathbf{x}^i) + \boldsymbol{\nabla}^T g_j(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i) \\ &\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}^i)^T \mathbf{B}_j(\mathbf{x} - \mathbf{x}^i),\ j = 1, \dots, p \\ \tilde{h}_k(\mathbf{x}) &= h_k(\mathbf{x}^i) + \boldsymbol{\nabla}^T h_k(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i) \\ &\quad + \frac{1}{2}(\mathbf{x} - \mathbf{x}^i)^T \mathbf{C}_k(\mathbf{x} - \mathbf{x}^i),\ k = 1, \dots, q\end{aligned} \tag{6.41}$$

with the Hessian matrices $\mathbf{A}$, $\mathbf{B}_j$ and $\mathbf{C}_k$ taking on the simple forms

$$\begin{aligned}\mathbf{A} &= \operatorname{diag}(a, a, \dots, a) = a\mathbf{I} \\ \mathbf{B}_j &= b_j\mathbf{I} \\ \mathbf{C}_k &= c_k\mathbf{I}.\end{aligned} \tag{6.42}$$

Clearly the identical entries along the diagonal of the Hessian matrices indicate that the approximate subproblems $P[i]$ are indeed spherically quadratic.

For the first subproblem ($i = 0$) a linear approximation is formed by setting the curvatures a, b_j and c_k to zero. Thereafter a, b_j and c_k are chosen so that the approximating functions (6.41) interpolate their corresponding actual functions at both $\mathbf{x}^i$ and $\mathbf{x}^{i-1}$. These conditions imply that for $i = 1, 2, 3, \ldots$

$$\begin{aligned} a &= \frac{2\left[f(\mathbf{x}^{i-1}) - f(\mathbf{x}^i) - \boldsymbol{\nabla}^T f(\mathbf{x}^i)(\mathbf{x}^{i-1} - \mathbf{x}^i)\right]}{\|\mathbf{x}^{i-1} - \mathbf{x}^i\|^2} \qquad (6.43) \\ b_j &= \frac{2\left[g_j(\mathbf{x}^{i-1}) - g_j(\mathbf{x}^i) - \boldsymbol{\nabla}^T g_j(\mathbf{x}^i)(\mathbf{x}^{i-1} - \mathbf{x}^i)\right]}{\|\mathbf{x}^{i-1} - \mathbf{x}^i\|^2},\ j = 1, \ldots, p \\ c_k &= \frac{2\left[h_k(\mathbf{x}^{i-1}) - h_k(\mathbf{x}^i) - \boldsymbol{\nabla}^T h_k(\mathbf{x}^i)(\mathbf{x}^{i-1} - \mathbf{x}^i)\right]}{\|\mathbf{x}^{i-1} - \mathbf{x}^i\|^2},\ k = 1, \ldots, q. \end{aligned}$$

If the gradient vectors $\boldsymbol{\nabla}^T f$, $\boldsymbol{\nabla}^T g_j$ and $\boldsymbol{\nabla}^T h_k$ are not known analytically, they may be approximated from functional data by means of first-order forward finite differences.

The particular choice of spherically quadratic approximations in the Dynamic-Q algorithm has implications on the computational and storage requirements of the method. Since the second derivatives of the objective function and constraints are approximated using function and gradient data, the $O(n^2)$ calculations and storage locations, which would usually be required for these second derivatives, are not needed. The computational and storage resources for the Dynamic-Q method are thus reduced to $O(n)$. At most, $4 + p + q + r + s$ n-vectors need be stored (where p, q, r and s are respectively the number of inequality and equality constraints and the number of lower and upper limits of the variables). These savings become significant when the number of variables becomes large. For this reason it is expected that the Dynamic-Q method is well suited, for example, to engineering problems such as structural optimization problems where a large number of variables are present.

In many optimization problems, additional bound constraints of the form $\hat{k}_i \leq x_i \leq \check{k}_i$ occur. Constants $\hat{k}_i$ and $\check{k}_i$ respectively represent lower and upper bounds for variable x_i. Since these constraints are of a simple form (having zero curvature), they need not be approximated in the Dynamic-Q method and are instead explicitly treated as special linear inequality constraints. Constraints corresponding to lower and upper limits are respectively of the form

$$\begin{aligned} \hat{g}_l(\mathbf{x}) &= \hat{k}_{vl} - x_{vl} \leq 0,\ l = 1, 2, \ldots, r \leq n \\ \check{g}_m(\mathbf{x}) &= x_{wm} - \check{k}_{wm} \leq 0,\ m = 1, 2, \ldots, s \leq n \end{aligned} \tag{6.44}$$

where $vl \in \hat{I} = (v1, v2, \ldots, vr)$ the set of r subscripts corresponding to the set of variables for which respective lower bounds $\hat{k}_{vl}$ are prescribed, and $wm \in \check{I} = (w1, w2, \ldots, ws)$ the set of s subscripts corresponding to the set of variables for which respective upper bounds $\check{k}_{wm}$ are prescribed. The subscripts vl and wm are used since there will, in general, not be n lower and upper limits, i.e. usually $r \neq n$ and $s \neq n$.

In order to obtain convergence to the solution in a controlled and stable manner, move limits are placed on the variables. For each approximate subproblem $P[i]$ this move limit takes the form of an additional single inequality constraint

$$g_\rho(\mathbf{x}) = \left\|\mathbf{x} - \mathbf{x}^i\right\|^2 - \rho^2 \leq 0 \tag{6.45}$$

where ρ is an appropriately chosen step limit and $\mathbf{x}^i$ is the solution to the previous subproblem.

The approximate subproblem, constructed at $\mathbf{x}^i$, to the optimization problem (6.40) (plus bound constraints (6.44) and move limit (6.45)), thus becomes $P[i]$:

$$\begin{gathered} \min_{\mathbf{x}} \tilde{f}(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^n \\ \text{subject to} \\ \begin{aligned} \tilde{g}_j(\mathbf{x}) &\leq 0,\ j = 1, 2, \ldots, p \\ \tilde{h}_k(\mathbf{x}) &= 0,\ k = 1, 2, \ldots, q \\ \hat{g}_l(\mathbf{x}) &\leq 0,\ l = 1, 2, \ldots, r \\ \check{g}_m(\mathbf{x}) &\leq 0,\ m = 1, 2, \ldots, s \\ g_\rho(\mathbf{x}) &= \left\|\mathbf{x} - \mathbf{x}^i\right\|^2 - \rho^2 \leq 0 \end{aligned} \end{gathered} \tag{6.46}$$

with solution $\mathbf{x}^{*i}$. The Dynamic-Q algorithm is given by Algorithm 6.5. In the Dynamic-Q method the subproblems generated are solved using the dynamic trajectory, or "leap-frog" (LFOPC) method of Snyman (1982, 1983) for unconstrained optimization applied to penalty function formulations (Snyman et al. 1994; Snyman 2000) of the constrained

Algorithm 6.5 Dynamic-Q algorithm

```
Initialization: Select starting point $\mathbf{x}^0$ and move limit $\rho$. Set $i := 0$.
Main procedure:
    1. Evaluate $f(\mathbf{x}^i)$, $g_j(\mathbf{x}^i)$ and $h_k(\mathbf{x}^i)$ as well as $\nabla f(\mathbf{x}^i)$, $\nabla g_j(\mathbf{x}^i)$ and
       $\nabla h_k(\mathbf{x}^i)$. If termination criteria are satisfied set $\mathbf{x}^* = \mathbf{x}^i$ and stop.
    2. Construct a local approximation $P[i]$ to the optimization problem
       at $\mathbf{x}^i$ using expressions (6.41) to (6.43).
    3. Solve the approximated subproblem $P[i]$ (given by (6.46)) using
       the constrained optimizer LFOPC with $\mathbf{x}^0 := \mathbf{x}^i$ (see Section 6.2)
       to give $\mathbf{x}^{*i}$.
    4. Set $i := i + 1$, $\mathbf{x}^i := \mathbf{x}^{*(i-1)}$ and return to Step 1.
```

problem. A brief description of the LFOPC algorithm is given in Section 6.2.

The LFOPC algorithm possesses a number of outstanding characteristics, which makes it highly suitable for implementation in the Dynamic-Q methodology. The algorithm requires only gradient information and no explicit line searches or function evaluations are performed. These properties, together with the influence of the fundamental physical principles underlying the method, ensure that the algorithm is extremely robust. This has been proven over many years of testing (Snyman 2000). A further desirable characteristic related to its robustness, and the main reason for its application in solving the subproblems in the Dynamic-Q algorithm, is that if there is no feasible solution to the problem, the LFOPC algorithm will still find the best possible compromised solution. The Dynamic-Q algorithm thus usually converges to a solution from an infeasible remote point without the need to use line searches between subproblems, as is the case with SQP. The LFOPC algorithm used by Dynamic-Q is identical to that presented in Snyman (2000) except for a minor change to LFOP which is advisable should the subproblems become effectively unconstrained.

6.4.3 Numerical results and conclusion

The Dynamic-Q method requires very few parameter settings by the user. Other than convergence criteria and specification of a maximum number of iterations, the only parameter required is the step limit ρ. The algorithm is not very sensitive to the choice of this parameter, however, ρ should be chosen of the same order of magnitude as the diameter of the region of interest. For the problems listed in Table 6.3 a step limit of $\rho = 1$ was used except for problems 72 and 106 where step limits $\rho = \sqrt{10}$ and $\rho = 100$ were used respectively.

Given specified positive tolerances ε_x, ε_f and ε_c, then at step i termination of the algorithm occurs if the normalized step size

$$\frac{\|\mathbf{x}^i - \mathbf{x}^{i-1}\|}{1 + \|\mathbf{x}^i\|} < \varepsilon_x \tag{6.47}$$

or if the normalized change in function value

$$\frac{|f^i - f_{\text{best}}|}{1 + |f_{\text{best}}|} < \varepsilon_f \tag{6.48}$$

where f_{best} is the lowest previous feasible function value and the current $\mathbf{x}^i$ is feasible. The point $\mathbf{x}^i$ is considered feasible if the absolute value of the violation of each constraint is less than ε_c. This particular function termination criterion is used since the Dynamic-Q algorithm may at times exhibit oscillatory behavior near the solution.

In Table 6.3, for the same starting points, the performance of the Dynamic-Q method on some standard test problems is compared to results obtained for Powell's SQP method as reported by Hock and Schittkowski (1981). The problem numbers given correspond to the problem numbers in Hock and Schittkowski's book. For each problem, the actual function value f_{act} is given, as well as, for each method, the calculated function value f^* at convergence, the relative function error

$$E^r = \frac{|f_{\text{act}} - f^*|}{1 + |f_{\text{act}}|} \tag{6.49}$$

and the number of function-gradient evaluations (N^{fg}) required for convergence. In some cases it was not possible to calculate the relative

Prob. #	n	f_{act}	SQP			Dynamic-Q		
			N^{fg}	f^*	E^r	N^{fg}	f^*	E^r
2	2	5.04E-02	16∼	2.84E+01	2.70E+01	7*	4.94E+00	<1.00E-08
10	2	-1.00E+00	12	-1.00E+00	5.00E-08	13	-1.00E+00	<1.00E-08
12	2	-3.00E+01	12	-3.00E+01	<1.00E-08	9	-3.00E+01	<1.00E-08
13	2	1.00E+00	45	1.00E+00	5.00E-08	50$	9.59E-01	2.07E-02
14	2	1.39E+00	6	1.39E+00	8.07E-09	5	1.39E+00	7.86E-07
15	2	3.07E+02	5	3.07E+02	<1.00E-08	15*	3.60E+02	5.55E-07
16	2	2.50E-01	6*	2.31E+01	<1.00E-08	5*	2.31E+01	<1.00E-08
17	2	1.00E+00	12	1.00E+00	<1.00E-08	16	1.00E+00	<1.00E-08
20	2	3.82E+01	20	3.82E+01	4.83E-09	4*	4.02E+01	<1.00E-08
22	2	1.00E+00	9	1.00E+00	<1.00E-08	3	1.00E+00	<1.00E-08
23	2	2.00E+00	7	2.00E+00	<1.00E-08	5	2.00E+00	<1.00E-08
24	2	-1.00E+00	5	-1.00E+00	<1.00E-08	4	-1.00E+00	1.00E-08
26	3	0.00E+00	19	4.05E-08	4.05E-08	27	1.79E-07	1.79E-07
27	3	4.00E-02	25	4.00E-02	1.73E-08	28	4.00E-02	9.62E-10
28	3	0.00E+00	5	2.98E-21	2.98E-21	12	7.56E-10	7.56E-10
29	3	-2.26E+01	13	-2.26E+01	8.59E-11	11	-2.26E+01	8.59E-11
30	3	1.00E+00	14	1.00E+00	<1.00E-08	5	1.00E+00	<1.00E-08
31	3	6.00E+00	10	6.00E+00	<1.00E-08	10	6.00E+00	1.43E-08
32	3	1.00E+00	3	1.00E+00	<1.00E-08	4	1.00E+00	<1.00E-08
33	3	-4.59E+00	5*	-4.00E+00	<1.00E-08	3*	-4.00E+00	<1.00E-08
36	3	-3.30E+03	4	-3.30E+03	<1.00E-08	15	-3.30E+03	<1.00E-08
45	5	1.00E+00	8	1.00E+00	<1.00E-08	7	1.00E+00	1.00E-08
52	5	5.33E+00	8	5.33E+00	5.62E-09	12	5.33E+00	1.02E-08
55	6	6.33E+00	1∼	6.00E+00	4.54E-02	2*	6.66E+00	1.30E-09
56	7	-3.46E+00	11	-3.46E+00	<1.00E-08	20	-3.46E+00	6.73E-08
60	3	3.26E-02	9	3.26E-02	3.17E-08	11	3.26E-02	1.21E-09
61	3	-1.44E+02	10	-1.44E+02	1.52E-08	10	-1.44E+02	1.52E-08
63	3	9.62E+02	9	9.62E+02	2.18E-09	6	9.62E+02	2.18E-09
65	3	9.54E-01	11∼	2.80E+00	9.47E-01	9	9.54E-01	2.90E-08
71	4	1.70E+01	5	1.70E+01	1.67E-08	6	1.70E+01	1.67E-08
72	4	7.28E+02	35	7.28E+02	1.37E-08	30	7.28E+02	1.37E-08
76	4	-4.68E+00	6	-4.68E+00	3.34E-09	8	-4.68E+00	3.34E-09
78	5	-2.92E+00	9	-2.92E+00	2.55E-09	6	-2.92E+00	2.55E-09
80	5	5.39E-02	7	5.39E-02	7.59E-10	6	5.39E-02	7.59E-10
81	5	5.39E-02	8	5.39E-02	1.71E-09	12	5.39E-02	1.90E-10
100	7	6.80E+02	20	6.80E+02	<1.00E-08	16	6.80E+02	1.46E-10
104	8	3.95E+00	19	3.95E+00	8.00E-09	42	3.95E+00	5.26E-08
106	8	7.05E+03	44	7.05E+03	1.18E-05	79	7.05E+03	1.18E-05
108	9	-8.66E-01	9*	-6.97E-01	1.32E-02	26	-8.66E-01	3.32E-09
118	15	6.65E+02	∼	∼	∼	38	6.65E+02	3.00E-08
Svan	21	2.80E+02	150	2.80E+02	9.96E-05	93	2.80E+02	1.59E-06

* Converges to a local minimum - listed E^r relative to function value at local minimum; ∼ Fails; $ Terminates on maximum number of steps.

Table 6.3: Performance of the Dynamic-Q and SQP optimization algorithms

function error due to rounding off of the solutions reported by Hock and Schittkowski. In these cases the calculated solutions were correct to at least eight significant digits. For the Dynamic-Q algorithm, convergence tolerances of $\varepsilon_f = 10^{-8}$ on the function value, $\varepsilon_x = 10^{-5}$ on the step size and $\varepsilon_c = 10^{-6}$ for constraint feasibility, were used. These were chosen to allow for comparison with the reported SQP results.

The result for the 12-corner polytope problem of Svanberg (1999) is also given. For this problem the results given in the SQP columns are for Svanberg's Method of Moving Asymptotes (MMA). The recorded number of function evaluations for this method is approximate since the results given correspond to 50 outer iterations of the MMA, each requiring about 3 function evaluations.

A robust and efficient method for nonlinear optimization, with minimal storage requirements compared to those of the SQP method, has been proposed and tested. The particular methodology proposed is made possible by the special properties of the LFOPC optimization algorithm (Snyman 2000), which is used to solve the quadratic subproblems. Comparison of the results for Dynamic-Q with the results for the SQP method show that equally accurate results are obtained with comparable number of function evaluations.

6.5 A gradient-only line search method for conjugate gradient methods

6.5.1 Introduction

Many engineering design optimization problems involve numerical computer analyses via, for example, FEM codes, CFD simulations or the computer modeling of the dynamics of multi-body mechanical systems. The computed objective function is therefore often the result of a complex sequence of calculations involving other computed or measured quantities. This may result in the presence of numerical noise in the objective function so that it exhibits non-smooth trends as design parameters are varied. It is well known that this presence of numerical noise in the design optimization problem inhibits the use of classical and traditional gradient-based optimization methods that employ line searches, such as for example, the conjugate gradient methods. The numerical noise may prevent or slow down convergence during optimization. It may also promote convergence to spurious local optima. The computational expense of the analyses, coupled to the convergence difficulties created by the numerical noise, is in many cases a significant obstacle to performing multidisciplinary design optimization.

In addition to the anticipated difficulties when applying the conjugate gradient methods to noisy optimization problems, it is also known that standard implementations of conjugate gradient methods, in which conventional line search techniques have been used, are less robust than one would expect from their theoretical quadratic termination property. Therefore the conjugate gradient method would, under normal circumstances, not be preferred to quasi-Newton methods (Fletcher 1987). In particular severe numerical difficulties arise when standard line searches are used in solving constrained problems through the minimization of associated penalty functions. However, there is one particular advantage of conjugate gradient methods, namely the particular simple form that requires no matrix operations in determining the successive search directions. Thus, conjugate gradient methods may be the only methods which are applicable to large problems with thousands of variables (Fletcher 1987), and are therefore well worth further investigation.

In this section a new implementation (ETOPC) of the conjugate gradient method (both for the Fletcher-Reeves and Polak-Ribiere versions (see Fletcher 1987) is presented for solving constrained problems. The essential novelty in this implementation is the use of a gradient-only line search technique originally proposed by the author (Snyman 1985), and used in the ETOP algorithm for unconstrained minimization. It will be shown that this implementation of the conjugate gradient method, not only easily overcomes the accuracy problem when applied to the minimization of penalty functions, but also economically handles the problem of severe numerical noise superimposed on an otherwise smooth underlying objective function.

6.5.2 Formulation of optimization problem

Consider again the general constrained optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, x_3, \dots, x_n]^T \in R^n \tag{6.50}$$

subject to the inequality and equality constraints:

$$\begin{aligned} g_j(\mathbf{x}) &\leq 0,\ j = 1, 2, \dots, m \\ h_j(\mathbf{x}) &= 0,\ j = 1, 2, \dots, r \end{aligned} \tag{6.51}$$

where the objective function $f(\mathbf{x})$, and the constraint functions $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$, are scalar functions of the real column vector $\mathbf{x}$. The optimum solution is denoted by $\mathbf{x}^*$, with corresponding optimum function value $f(\mathbf{x}^*)$.

The most straightforward way of handling the constraints is via the unconstrained minimization of the penalty function:

$$P(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^{r} \rho_j h_j^2(\mathbf{x}) + \sum_{j=1}^{m} \beta_j g_j^2(\mathbf{x}) \tag{6.52}$$

where $\rho_j \gg 0,\ \beta_j = 0$ if $g_j(\mathbf{x}) \leq 0$, and $\beta_j = \mu_j \gg 0$ if $g_j(\mathbf{x}) > 0$.

Usually $\rho_j = \mu_j = \mu \gg 0$ for all j, with the corresponding penalty function being denoted by $P(\mathbf{x}, \mu)$.

Central to the application of the conjugate gradient method to penalty function formulated problems presented here, is the use of an unconventional line search method for unconstrained minimization, proposed by the author, in which no function values are explicitly required (Snyman 1985). Originally this gradient-only line search method was applied to the conjugate gradient method in solving a few very simple unconstrained problems. For somewhat obscure reasons, given in the original paper (Snyman 1985) and briefly hinted to in this section, the combined method (novel line search plus conjugate gradient method) was called the ETOP (Euler-Trapezium Optimizer) algorithm. For this historical reason, and to avoid confusion, this acronym will be retained here to denote the combined method for unconstrained minimization. In subsequent unreported numerical experiments, the author was successful in solving a number of more challenging practical constrained optimization problems via penalty function formulations of the constrained problem, with ETOP being used in the unconstrained minimization of the sequence of penalty functions. ETOP, applied in this way to constrained problems, was referred to as the ETOPC algorithm. Accordingly this acronym will also be used here.

6.5.3 Gradient-only line search

The line search method used here, and originally proposed by the author (Snyman 1985) uses no explicit function values. Instead the line search is implicitly done by using only two gradient vector evaluations at two points along the search direction and assuming that the function is near-quadratic along this line. The essentials of the gradient-only line search, for the case where the function $f(\mathbf{x})$ is unconstrained, are as follows. Given the current design point $\mathbf{x}^k$ at iteration k and next search direction $\mathbf{v}^{k+1}$, then compute

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{v}^{k+1}\tau \tag{6.53}$$

where τ is some suitably chosen positive parameter. The step taken in (6.53) may be seen as an "Euler step". With this step given by

$$\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k = \mathbf{v}^{k+1}\tau \tag{6.54}$$

the line search in the direction $\mathbf{v}^{k+1}$ is equivalent to finding $\mathbf{x}^{*k+1}$ defined by

$$f(\mathbf{x}^{*k+1}) = \min_{\lambda} f(\mathbf{x}^k + \lambda\Delta\mathbf{x}^k). \tag{6.55}$$

These steps are depicted in Figure 6.3.

It was indicated in Snyman (1985) that for the step $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{v}^{k+1}\tau$ the change in function value Δf_k, in the unconstrained case, can be approximated without explicitly evaluating the function $f(\mathbf{x})$. Here a more formal argument is presented via the following lemma.

6.5.3.1 Lemma 1

For a general quadratic function, the change in function value, for the step $\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k = \mathbf{v}^{k+1}\tau$ is given by:

$$\Delta f_k = -\langle \mathbf{v}^{k+1}, \frac{1}{2}(\mathbf{a}^k + \mathbf{a}^{k+1})\tau\rangle \tag{6.56}$$

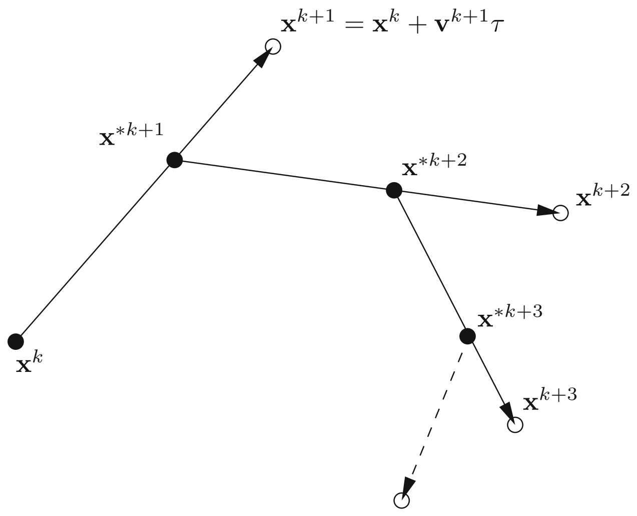

Figure 6.3: Successive steps in line search procedure

where $\mathbf{a}^k = -\boldsymbol{\nabla} f(\mathbf{x}^k)$ and $\langle\ ,\ \rangle$ denotes the scalar product.

Proof:

In general, by Taylor's theorem:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) = \langle \mathbf{x}^{k+1} - \mathbf{x}^k, \boldsymbol{\nabla} f(\mathbf{x}^k)\rangle + \frac{1}{2}\langle \Delta\mathbf{x}^k, \mathbf{H}(\mathbf{x}^a)\Delta\mathbf{x}^k\rangle$$

and

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) = \langle \mathbf{x}^{k+1} - \mathbf{x}^k, \boldsymbol{\nabla} f(\mathbf{x}^{k+1})\rangle - \frac{1}{2}\langle \Delta\mathbf{x}^k, \mathbf{H}(\mathbf{x}^b)\Delta\mathbf{x}^k\rangle$$

where $\mathbf{x}^a = \mathbf{x}^k + \theta_0\Delta\mathbf{x}^k,\ \mathbf{x}^b = \mathbf{x}^k + \theta_1\Delta\mathbf{x}^k$ and both θ_0 and θ_1 in the interval $[0,1]$, and where $\mathbf{H}(\mathbf{x})$ denotes the Hessian matrix of the general function $f(\mathbf{x})$. Adding the above two expressions gives:

$$\begin{aligned} f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) &= \frac{1}{2}\langle \mathbf{x}^{k+1} - \mathbf{x}^k, \boldsymbol{\nabla} f(\mathbf{x}^k) + \boldsymbol{\nabla} f(\mathbf{x}^{k+1})\rangle \\ &\quad + \frac{1}{4}\langle \Delta\mathbf{x}^k, [\mathbf{H}(\mathbf{x}^a) - \mathbf{H}(\mathbf{x}^b)]\Delta\mathbf{x}^k\rangle. \end{aligned}$$

If $f(\mathbf{x})$ is quadratic then $\mathbf{H}(\mathbf{x})$ is constant and it follows that

$$\Delta f_k = f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) = -\langle \mathbf{v}^{k+1}, \frac{1}{2}(\mathbf{a}^k + \mathbf{a}^{k+1})\tau\rangle$$

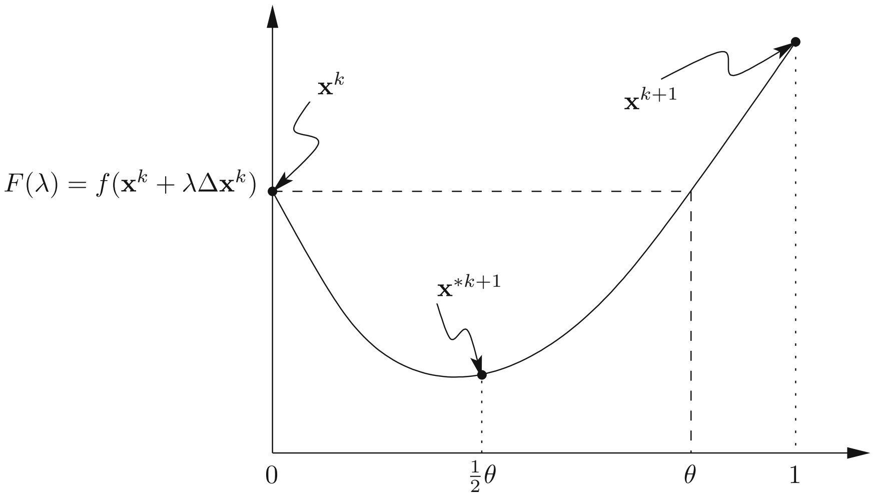

Figure 6.4: Approximation of minimizer $\mathbf{x}^{*k+1}$ in the direction $\mathbf{v}^{k+1}$

where $\mathbf{a}^k = \boldsymbol{\nabla} f(\mathbf{x}^k)$, which completes the proof. □

By using expression (6.56) the position of the minimizer $\mathbf{x}^{*k+1}$ (see Figure 6.4), in the direction $\mathbf{v}^{k+1}$, can also be approximated without any explicit function evaluation. This conclusion follows formally from the second lemma given below. Note that in (6.56) the second quantity in the scalar product corresponds to an average vector given by the "trapezium rule". This observation together with the remark following equation (6.53), gave rise to the name "Euler-trapezium optimizer (ETOP)" when applying this line search technique in the conjugate gradient method.

6.5.3.2 Lemma 2

For $f(\mathbf{x})$ a positive-definite quadratic function the point $\mathbf{x}^{*k+1}$ defined by $f(\mathbf{x}^{*k+1}) = \min_\lambda f(\mathbf{x}^k + \lambda\Delta\mathbf{x}^k)$ is given by

$$\mathbf{x}^{*k+1} = \mathbf{x}^k + \frac{1}{2}\theta\Delta\mathbf{x}^k \tag{6.57}$$

where

$$\theta = \rho/(\langle \mathbf{v}^{k+1}, \frac{1}{2}(\mathbf{a}^k + \mathbf{a}^{k+1})\tau\rangle + \rho) \text{ and } \rho = -\langle \Delta\mathbf{x}^k, \mathbf{a}^k\rangle. \tag{6.58}$$

Proof:

First determine θ such that

$$f(\mathbf{x}^k + \theta\Delta\mathbf{x}^k) = f(\mathbf{x}^k).$$

By Taylor's expansion:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) = \rho + \frac{1}{2}\langle\Delta\mathbf{x}^k, \mathbf{H}\Delta\mathbf{x}^k\rangle, \text{ i.e., } \frac{1}{2}\langle\Delta\mathbf{x}^k, \mathbf{H}\Delta\mathbf{x}^k\rangle = \Delta f_k - \rho$$

which gives for the step $\theta\Delta\mathbf{x}$:

$$f(\mathbf{x}^k + \theta\Delta\mathbf{x}^k) - f(\mathbf{x}^k) = \theta\rho + \frac{1}{2}\theta^2\langle\Delta\mathbf{x}^k, \mathbf{H}\Delta\mathbf{x}^k\rangle = \theta\rho + \theta^2(\Delta f_k - \rho).$$

For both function values to be the same, θ must therefore satisfy:

$$0 = \theta(\rho + \theta(\Delta f_k - \rho))$$

which has the non-trivial solution:

$$\theta = -\rho/(\Delta f_k - \rho).$$

Using the expression for Δf_k given by Lemma 1, it follows that:

$$\theta = \rho/(\langle\mathbf{v}^{k+1}, \frac{1}{2}(\mathbf{a}^k + \mathbf{a}^{k+1})\tau\rangle + \rho)$$

and by the symmetry of quadratic functions that

$$\mathbf{x}^{*k+1} = \mathbf{x}^k + \frac{1}{2}\theta\Delta\mathbf{x}^k.$$

$\square$

Expressions (6.57) and (6.58) may of course also be used in the general non-quadratic case, to determine an approximation to the minimizer $\mathbf{x}^{*k+1}$ in the direction $\mathbf{v}^{k+1}$, when performing successive line searches using the sequence of descent directions, $\mathbf{v}^{k+1}$, $k = 1, 2, \ldots$ Thus in practice, for the next $(k+1)$-th iteration, set $\mathbf{x}^{k+1} := \mathbf{x}^{*k+1}$, and with the next selected search direction $\mathbf{v}^{k+2}$ proceed as above, using expressions (6.57) and (6.58) to find $\mathbf{x}^{*k+2}$ and then set $\mathbf{x}^{k+2} := \mathbf{x}^{*k+2}$. Continue iterations in this way, with only two gradient vector evaluations done per line search, until convergence is obtained.

In summary, explicit function evaluations are unnecessary in the above line search procedure, since the two computed gradients along the search direction allow for the computation of an approximation (6.56) to the change in objective function, which in turn allows for the estimation of the position of the minimum along the search line via expressions (6.57) and (6.58), based on the assumption that the function is near quadratic in the region of the search.

6.5.3.3 Heuristics

Of course in general the objective function may not be quadratic and positive-definite. Additional heuristics are therefore required to ensure descent, and to see to it that the step size (corresponding to the parameter τ between successive gradient evaluations, is neither too small nor too large. The details of these heuristics are as set out below.

(i) In the case of a successful step having been taken, with Δf_k computed via (6.56) negative, i.e. descent, and θ computed via (6.58) positive, i.e. the function is locally strictly convex, as shown in Figure 6.4, τ is increased by a factor of 1.5 for the next search direction.

(ii) It may turn out that although Δf_k computed via (6.56) is negative, that θ computed via (6.58) is also negative. The latter implies that the function along the search direction is locally concave. In this case set $\theta := -\theta$ in computing $\mathbf{x}^{*k+1}$ by (6.57), so as to ensure a step in the descent direction, and also increase τ by the factor 1.5 before computing the step for the next search direction using (6.53).

(iii) It may happen that Δf_k computed by (6.56) is negative and exactly equal to ρ, i.e. $\Delta f_k - \rho = 0$. This implies zero curvature with $\theta = \infty$ and the function is therefore locally linear. In this case enforce the value $\theta = 1$. This results in the setting, by (6.57), of $\mathbf{x}^{*k+1}$ equal to a point halfway between $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$. In this case τ is again increased by the factor of 1.5.

(iv) If both Δf_k and θ are positive, which is the situation depicted in Figure 6.4, then τ is halved before the next step.

(v) In the only outstanding and unlikely case, should it occur, where Δf^k is positive and θ negative, τ is unchanged.

(vi) For usual unconstrained minimization the initial step size parameter selection is $\tau = 0.5$.

The new gradient-only line search method may of course be applied to any line search descent method for the unconstrained minimization of a general multi-variable function. Here its application is restricted to the conjugate gradient method.

6.5.4 Conjugate gradient search directions and SUMT

The search vectors used here correspond to the conjugate gradient directions (Bazaraa et al. 1993). In particular for $k = 0, 1, \ldots$, the search vectors are

$$\mathbf{v}^{k+1} = (-\boldsymbol{\nabla} f(\mathbf{x}^k) + \beta_{k+1}\mathbf{v}^k/\tau)\tau = \mathbf{s}^{k+1}\tau \tag{6.59}$$

where $\mathbf{s}^{k+1}$ denote the usual conjugate gradient directions, $\beta_1 = 0$ and for $k > 0$:

$$\beta_{k+1} = \|\boldsymbol{\nabla} f(\mathbf{x}^k)\|^2/\|\boldsymbol{\nabla} f(\mathbf{x}^{k-1})\|^2 \tag{6.60}$$

for the Fletcher-Reeves implementation, and for the Polak-Ribiere version:

$$\beta_{k+1} = \langle \boldsymbol{\nabla} f(\mathbf{x}^k) - \boldsymbol{\nabla} f(\mathbf{x}^{k-1}), \boldsymbol{\nabla} f(\mathbf{x}^k)\rangle/\|\boldsymbol{\nabla} f(\mathbf{x}^{k-1})\|^2. \tag{6.61}$$

As recommended by Fletcher (1987), the conjugate gradient algorithm is restarted in the direction of steepest descent when $k > n$.

For the constrained problem the unconstrained minimization is of course applied to successive penalty function formulations $P(\mathbf{x})$ of the form shown in (6.52), using the well known Sequential Unconstrained Minimization Technique (SUMT) (Fiacco and McCormick 1968). In SUMT, for $j = 1, 2, \ldots$, until convergence, successive unconstrained minimizations are performed on successive penalty functions $P(\mathbf{x}) = P(\mathbf{x}, \mu^{(j)})$ in which the overall penalty parameter $\mu^{(j)}$ is successively increased: $\mu^{(j+1)} := 10\mu^{(j)}$. The corresponding initial step size parameter is set at $\tau = 0.5/\mu^{(j)}$ for each sub problem j. This application of ETOP to the constrained problem, via the unconstrained minimization of successive penalty functions, is referred to as the ETOPC algorithm. In practice, if analytical expressions for the components of the gradient of the objective function are not available, they may be calculated with sufficient accuracy by finite differences. However, when the presence of severe noise is suspected, the application of the gradient-only search method with conjugate gradient search directions, requires that *central finite difference* approximations of the gradients be used in order to effectively smooth out the noise. In this case relatively excessive perturbations δx_i in x_i must be used, which in practice may typically be of the order of 0.1 times the range of interest!

In the application of ETOPC a limit Δ_m, is in practice set to the maximum allowable magnitude Δ^* of the step $\boldsymbol{\Delta}^* = \mathbf{x}^{*k+1} - \mathbf{x}^k$. If Δ^* is greater than Δ_m, then set

$$\mathbf{x}^{k+1} := \mathbf{x}^k + (\mathbf{x}^{*k+1} - \mathbf{x}^k)\Delta_m/\Delta^* \tag{6.62}$$

and restart the conjugate gradient procedure, with $\mathbf{x}^0 := \mathbf{x}^{k+1}$, in the direction of steepest descent. If the maximum allowable step is taken n times in succession, then Δ_m is doubled.

6.5.5 Numerical results

The proposed new implementation of the conjugate gradient method (both the Fletcher- Reeves and Polak-Ribiere versions) is tested here using 40 different problems arbitrarily selected from the famous set of test problems of Hock and Schittkowski (1981). The problem numbers (Pr. #) in the tables, correspond to the numbering used in Hock and Schittkowski. The final test problem, (12-poly), is the 12 polytope problem of Svanberg (1995, 1999). The number of variables (n) of the test problems ranges from 2 to 21 and the number of constraints (m plus r) per problem, from 1 to 59. The termination criteria for the ETOPC algorithm are as follows:

(i) Convergence tolerances for successive approximate sub-problems within SUMT: ε_g for convergence on the norm of the gradient vector, i.e. terminate if $\|\boldsymbol{\nabla} P(\mathbf{x}^{*k+1}, \mu)\| < \varepsilon_g$, and ε_x for convergence on average change of design vector: i.e. terminate if $\frac{1}{2}\|\mathbf{x}^{*k+1} - \mathbf{x}^{*k-1}\| < \varepsilon_x$.

(ii) Termination of the SUMT procedure occurs if the absolute value of the relative difference between the objective function values at the solution points of successive SUMT problems is less than ε_f.

6.5.5.1 Results for smooth functions with no noise

For the initial tests no noise is introduced. For high accuracy requirements (relative error in optimum objective function value to be less than 10^{-8}), it is found that the proposed new conjugate gradient implementation performs as robust as, and more economical than, the traditional penalty function implementation, FMIN, of Kraft and Lootsma reported in Hock and Schittkowski (1981). The detailed results are as tabulated in Table 6.4. Unless otherwise indicated the algorithm settings are: $\varepsilon_x = 10^{-8}$, $\varepsilon_g = 10^{-5}$, $\Delta_m = 1.0$, $\varepsilon_f = 10^{-8}$, $\mu^{(1)} = 1.0$ and $iout = 15$, where $iout$ denotes the maximum number of SUMT iterations allowed. The number of gradient vector evaluations required by ETOPC for the different problems are denoted by nge (note that the number of explicit function evaluations is zero), and the relative error in function value at convergence to the point $\mathbf{x}^c$ is denoted by r_f, which is computed from

$$r_f = |f(\mathbf{x}^*) - f(\mathbf{x}^c)|/(|f(\mathbf{x}^*)| + 1). \tag{6.63}$$

For the FMIN algorithm only the number of explicit objective function evaluations nfe are listed, together with the relative error r_f at convergence. The latter method requires, in addition to the number of function evaluations listed, a comparable number of gradient vector evaluations, which is not given here (see Hock and Schittkowski 1981).

6.5.5.2 Results for severe noise introduced in the objective function

Following the successful implementation for the test problems with no noise, all the tests were rerun, but with severe relative random noise introduced in the objective function $f(\mathbf{x})$ and all gradient components

		Fletcher-Reeves		Polak-Ribiere		FMIN	
Pr. #	$n\ m\ r$	nge	r_f	nge	r_f	nfe	r_f
1	2 1 -	100	$< 10^{-14}$	103	$< 10^{-13}$	549	$< 10^{-8}$
2	2 1 -	290	$< 10^{-8}$	318	$< 10^{-8}$	382	1×10^{-8}
10	2 1 -	231	$< 10^{-9}$	247	$< 10^{-9}$	289	7×10^{-8}
12	2 1 -	163	$< 10^{-10}$	184	$< 10^{-10}$	117	1×10^{-8}
13[1]	2 3 -	4993[2]	0.028	4996[2]	0.034	1522	0.163
14	2 1 1	214	$< 10^{-10}$	200	$< 10^{-10}$	232	2×10^{-7}
15	2 3 -	699	$< 10^{-9}$	632	$< 10^{-9}$	729	4×10^{-7}
16	2 5 -	334	$< 10^{-9}$	284	$< 10^{-7}$	362	1×10^{-8}
17	2 5 -	218	$< 10^{-9}$	209	$< 10^{-9}$	541	1×10^{-8}
20[3]	2 5 -	362	$< 10^{-9}$	375	$< 10^{-9}$	701	4×10^{-6}
22	2 2 -	155	$< 10^{-9}$	202	$< 10^{-9}$	174	1×10^{-7}
23	2 9 -	257	$< 10^{-9}$	244	$< 10^{-9}$	423	6×10^{-6}
24	2 5 -	95	$< 10^{-11}$	163	2×10^{-6}	280	2×10^{-8}
26	3 - 1	78	$< 10^{-8}$	100	2×10^{-8}	182	1×10^{-8}
27	3 - 1	129	$< 10^{-8}$	115	$< 10^{-8}$	173	1×10^{-8}
28	3 - 1	17	$< 10^{-28}$	17	$< 10^{-28}$	23	$< 10^{-8}$
29	3 1 -	254	$< 10^{-10}$	267	$< 10^{-10}$	159	$< 10^{-8}$
30	3 7 -	115	$< 10^{-10}$	124	$< 10^{-10}$	1199	4×10^{-8}
31	3 7 1	309	$< 10^{-9}$	274	$< 10^{-9}$	576	$< 10^{-8}$
32	3 7 1	205	$< 10^{-10}$	207	$< 10^{-10}$	874	$< 10^{-8}$
33[3]	3 6 -	272[3]	$< 10^{-10}$	180	$< 10^{-10}$	672[3]	3×10^{-7}
36	3 7 -	336	$< 10^{-12}$	351	$< 10^{-10}$	263	2×10^{-6}
45	5 10 -	175	$< 10^{-10}$	150	$< 10^{-10}$	369	$< 10^{-8}$
52	5 - 3	403	$< 10^{-9}$	388	$< 10^{-9}$	374	$< 10^{-8}$
55[3]	6 8 6	506	$< 10^{-9}$	488	$< 10^{-9}$	581[3]	3×10^{-8}
56	7 - 4	316	6×10^{-8}	289	7×10^{-8}	446	$< 10 - 8$
60	3 6 1	198	$< 10^{-10}$	189	$< 10^{-10}$	347	$1 \times 10 - 8$
61	3 - 2	205	$< 10^{-10}$	201	$< 10^{-10}$	217	$< 10 - 8$
63	3 3 2	205	$< 10^{-10}$	208	$< 10^{-10}$	298	$< 10 - 8$
65	3 7 -	179	$< 10^{-8}$	198	$< 10^{-10}$	-	fails
71	4 9 1	493	$< 10^{-9}$	536	$< 10^{-9}$	1846	5×10^{-3}
72[4]	4 10 -	317	$< 10^{-10}$	298	$< 10^{-10}$	1606	5×10^{-2}
76	4 7 -	224	$< 10^{-10}$	227	$< 10^{-10}$	424	$< 10^{-8}$
78	5 - 3	261	$< 10^{-10}$	264	$< 10^{-10}$	278	$< 10^{-8}$
80	5 10 3	192	$< 10^{-11}$	194	$< 10^{-11}$	1032	2×10^{-8}
81[5]	5 10 3	138	$< 10^{-11}$	158	$< 10^{-10}$	1662	5×10^{-7}
106[6]	8 22 -	6060	5×10^{-6}	6496	3×10^{-5}	-	fails
108	9 14 -	600	$< 10^{-10}$	519	$< 10^{-10}$	984	7×10^{-5}
118[7]	15 29 -	1233	$< 10^{-8}$	1358	$< 10^{-8}$	-	fails
12-poly[7]	21 22 -	844	$< 10^{-9}$	1478	$< 10^{-9}$	-	-

[1]Constraint qualification not satisfied. [2]Termination on maximum number of steps. [3]Convergence to local minimum. [4]$\mu^{(0)} = 1.0,\ \Delta_m = 1.0$. [5]$\mu^{(0)} = 10^2$. [6]$\Delta_m = 10^2$. [7]Gradients by central finite differences, $\delta x_i = 10^{-6}, \varepsilon_x = 10^{-6}$.

Table 6.4: The respective performances of the new conjugate gradient implementation ETOPC and FMIN for test problems with no noise introduced

computed by central finite differences. The influence of noise is investigated for two cases, namely, for a variation of the superimposed uniformly distributed random noise as large as (i) 5% and (ii) 10% of $(1 + |f(\mathbf{x}^*)|)$, where $\mathbf{x}^*$ is the optimum of the underlying smooth problem. The detailed results are shown in Table 6.5. The results are listed only for the Fletcher-Reeves version. The results for the Polak- Ribiere implementation are almost identical. Unless otherwise indicated the algorithm settings are: $\delta x_i = 1.0$, $\varepsilon_g = 10^{-5}$, $\Delta_m = 1.0$, $\varepsilon_f = 10^{-8}$, $\mu^{(0)} = 1.0$ and $iout = 6$, where $iout$ denotes the maximum number of SUMT iterations allowed. For termination of sub-problem on step size, ε_x was set to $\varepsilon_x := 0.005\sqrt{n}$ for the initial sub-problem. Thereafter it is successively halved for each subsequent sub-problem.

The results obtained are surprisingly good with, in most cases, fast convergence to the neighbourhood of the known optimum of the underlying smooth problem. In 90% of the cases regional convergence was obtained with relative errors $r_x < 0.025$ for 5% noise and $r_x < 0.05$ for 10% noise, where

$$r_x = \|\mathbf{x}^* - \mathbf{x}^c\|/(\|\mathbf{x}^*\| + 1) \tag{6.64}$$

and $\mathbf{x}^c$ denotes the point of convergence. Also in 90% of the test problems the respective relative errors in final objective function values were $r_f < 0.025$ for 5% noise and $r_f < 0.05$ for 10% noise, where r_f is as defined in (6.63).

6.5.6 Conclusion

The ETOPC algorithm performs exceptionally well for a first order method in solving constrained problems where the functions are smooth. For these problems the gradient only penalty function implementation of the conjugate gradient method performs as well, if not better than the best conventional implementations reported in the literature, in producing highly accurate solutions.

In the cases where severe noise is introduced in the objective function, relatively fast convergence to the neighborhood of $\mathbf{x}^*$, the solution of the underlying smooth problem, is obtained. Of interest is the fact that with the reduced accuracy requirement associated with the presence of noise, the number of function evaluations required to obtain sufficiently

		5% noise			10% noise		
Pr. #	*nmr*	*nge*	r_f	r_x	*nge*	r_f	r_x
1	2 1 -	54	0.035	5×10^{-3}	54	0.06	5×10^{-3}
2	2 1 -	80	2×10^{-3}	2×10^{-3}	87	9×10^{-3}	2×10^{-3}
10	2 1 -	120	0.02	0.022	160	0.048	0.023
12	2 1 -	99	0.018	8×10^{-2}	232	0.006	0.011
13	2 3 -	394	0.079	0.044	187	0.189	0.095
14	2 1 1	138	0.025	6×10^{-4}	126	0.041	6×10^{-4}
15	2 3 -	152	6×10^{-5}	2×10^{-5}	154	0.006	8×10^{-5}
16[1]	2 5 -	250	0.128	0.13	175	0.135	0.16
17	2 5 -	84	0.012	7×10^{-6}	77	0.041	3×10^{-4}
20	2 5 -	89	0.009	2×10^{-5}	105	0.001	2×10^{-5}
22	2 2 -	75	0.01	4×10^{-5}	86	0.035	9×10^{-5}
23	2 9 -	103	0.008	9×10^{-4}	100	0.005	7×10^{-4}
24	2 5 -	75	0.0095	4×10^{-5}	137	0.014	3×10^{-5}
26	3 - 1	63	0.019	2×10^{-3}	71	0.04	3×10^{-3}
27	3 - 1	159	0.015	0.014	132	0.022	0.036
28	3 - 1	46	0.018	6×10^{-3}	49	0.009	0.025
29	3 1 -	232	0.013	0.01	251	0.046	0.015
30	3 7 -	52	0.025	4×10^{-3}	72	0.043	6×10^{-3}
31	3 7 1	123	0.015	9×10^{-4}	183	0.031	0.013
32	3 7 1	89	0.006	4×10^{-3}	107	0.031	5×10^{-3}
33	3 6 -	183	0.016	0.035	83	0.026	3×10^{-3}
36[2]	3 7 -	177	0.018	6×10^{-5}	179	0.01	8×10^{-5}
45	5 10 -	122	0.0013	9×10^{-4}	92	0.009	4×10^{-5}
52	5 - 3	239	0.019	0.042	318	0.041	0.071
55	6 8 6	137	0.016	5×10^{-3}	188	0.041	4×10^{-3}
56	7 - 4	166	0.012	0.014	144	0.03	0.038
60	3 6 1	95	0.021	0.071	83	0.018	0.033
61[2]	3 - 2	105	0.019	2×10^{-3}	83	0.026	9×10^{-4}
63[2]	3 3 2	198	0.02	8×10^{-3}	652	0.016	0.06
65	3 7 -	94	4×10^{-3}	3×10^{-3}	106	0.012	0.003
71	4 9 1	164	0.021	0.022	143	0.021	0.035
72	4 10 -	454	0.01	0.025	578	0.005	0.094
76	4 7 -	131	0.022	0.002	148	0.012	0.041
78	5 - 3	87	0.011	0.004	88	0.037	0.002
80	5 10 3	92	0.011	0.025	105	0.005	0.02
81[3]	5 10 3	39	0.017	0.032	47	0.012	0.031
106[4]	8 22 -	6016	0.023	0.088	8504	0.038	0.113
108[2]	9 14 -	113	0.017	0.04	140	0.04	0.025
118[2]	15 29 -	395	0.012	0.041	371	0.049	0.1
12-poly[2]	21 22 -	476	0.012	0.065	607	0.047	0.1

[1]$\delta x_i = 10^{-1}$. [2]$\delta x_i = 10$. [3]$\mu^{(0)} = 10^2$ [4]$\delta x_i = 10^3, \Delta_m = 10^2$.

Table 6.5: Performance of ETOPC for test problems with severe noise introduced

accurate solutions in the case of noise, is on the average much less than that necessary for the high accuracy solutions for smooth functions. As already stated, ETOPC yields in 90% of the cases regional convergence with relative errors $r_x < 0.025$ for 5% noise, and $r_x < 0.05$ for 10% noise. Also in 90% of the test problems the respective relative errors in the final objective function values are $r_f < 0.025$ for 5% noise and $r_f < 0.05$ for 10% noise. In the other 10% of the cases the relative errors are also acceptably small. These accuracies are more than sufficient for multidisciplinary design optimization problems where similar noise may be encountered.

6.6 Global optimization using dynamic search trajectories

6.6.1 Introduction

The problem of globally optimizing a real valued function is inherently intractable (unless hard restrictions are imposed on the objective function) in that no practically useful characterization of the global optimum is available. Indeed the problem of determining an accurate estimate of the global optimum is mathematically ill-posed in the sense that very similar objective functions may have global optima very distant from each other (Schoen 1991). Nevertheless, the need in practice to find a relative low local minimum has resulted in considerable research over the last decade to develop algorithms that attempt to find such a low minimum, e.g. see Törn and Zilinskas (1989).

The general global optimization problem may be formulated as follows. Given a real valued objective function $f(\mathbf{x})$ defined on the set $\mathbf{x} \in D$ in $\mathbb{R}^n$, find the point $\mathbf{x}^*$ and the corresponding function value f^* such that

$$f^* = f(\mathbf{x}^*) = \text{ minimum } \{f(\mathbf{x})|\mathbf{x} \in D\} \tag{6.65}$$

if such a point $\mathbf{x}^*$ exists. If the objective function and/or the feasible domain D are non-convex, then there may be many local minima which are not global.

If D corresponds to all $\mathbb{R}^n$ the optimization problem is *unconstrained.*

Alternatively, simple bounds may be imposed, with D now corresponding to the hyper box (or domain or region of interest) defined by

$$D = \{\mathbf{x}|\boldsymbol{\ell} \leq \mathbf{x} \leq \boldsymbol{u}\} \tag{6.66}$$

where $\boldsymbol{\ell}$ and $\boldsymbol{u}$ are n-vectors defining the respective lower and upper bounds on $\mathbf{x}$.

From a *mathematical* point of view, Problem (6.65) is essentially *unsolvable*, due to a lack of mathematical conditions characterizing the global optimum, as opposed to the local optimum of a smooth continuous function, which is characterized by the behavior of the problem function (Hessians and gradients) at the minimum (Arora et al. 1995) (viz. the Karush-Kuhn-Tucker conditions). Therefore, the global optimum f^* can only be obtained by an exhaustive search, except if the objective function satisfies certain subsidiary conditions (Griewank 1981), which mostly are of limited practical use (Snyman and Fatti 1987). Typically, the conditions are that f should satisfy a Lipschitz condition with known constant L and that the search area is bounded, e.g. for all $\mathbf{x}, \bar{\mathbf{x}} \in \mathbf{X}$

$$|f(\mathbf{x}) - f(\bar{\mathbf{x}})| \ \leq \ L\|\mathbf{x} - \bar{\mathbf{x}}\|. \tag{6.67}$$

So called space-covering deterministic techniques have been developed (Dixon et al. 1975) under these special conditions. These techniques are expensive, and due to the need to know L, of limited practical use.

Global optimization algorithms are divided into two major classes (Dixon et al. 1975): deterministic and stochastic (from the Greek word *stokhastikos*, i.e. 'governed by the laws of probability'). Deterministic methods can be used to determine the global optimum through exhaustive search. These methods are typically extremely expensive. With the introduction of a stochastic element into deterministic algorithms, the deterministic *guarantee* that the global optimum can be found is relaxed into a *confidence measure*. Stochastic methods can be used to assess the probability of having obtained the global minimum. Stochastic ideas are mostly used for the development of stopping criteria, or to approximate the regions of attraction as used by some methods (Arora et al. 1995).

The stochastic algorithms presented herein, namely the Snyman-Fatti algorithm and the modified bouncing ball algorithm (Groenwold and Snyman 2002), both depend on dynamic search trajectories to minimize the objective function. The respective trajectories, namely the motion of a particle of unit mass in a n-dimensional conservative force field, and the trajectory of a projectile in a conservative gravitational field, are modified to increase the likelihood of convergence to a low local minimum.

6.6.2 The Snyman-Fatti trajectory method

The essentials of the original SF algorithm (Snyman and Fatti 1987) using dynamic search trajectories for unconstrained global minimization will now be discussed. The algorithm is based on the local algorithms presented by Snyman (1982, 1983). For more details concerning the motivation of the method, its detailed construction, convergence theorems, computational aspects and some of the more obscure heuristics employed, the reader is referred to the original paper and also to the more recent review article by Snyman and Kok (2009).

6.6.2.1 Dynamic trajectories

In the SF algorithm successive sample points $\mathbf{x}^j, j = 1, 2, ...$, are selected at random from the box D defined by (6.66). For *each* sample point $\mathbf{x}^j$, a sequence of trajectories T^i, $i = 1, 2, ...$, is computed by numerically solving the successive initial value problems:

$$\begin{gathered}\ddot{\mathbf{x}}(t) = -\boldsymbol{\nabla} f(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0^i \;;\;\; \dot{\mathbf{x}}(0) = \dot{\mathbf{x}}_0^i.\end{gathered} \tag{6.68}$$

This trajectory represents the motion of a particle of unit mass in a n-dimensional conservative force field, where the function to be minimized represents the potential energy.

Trajectory T^i is terminated when $\mathbf{x}(t)$ reaches a point where $f(\mathbf{x}(t))$ is arbitrarily close to the value $f(\mathbf{x}_0^i)$ while moving "uphill", or more precisely, if $\mathbf{x}(t)$ satisfies the conditions

$$\begin{gathered}f(\mathbf{x}(t)) > f(\mathbf{x}_0^i) - \epsilon_u \\ \text{and} \quad \dot{\mathbf{x}}(t)^T \boldsymbol{\nabla} f(\mathbf{x}(t)) > 0\end{gathered} \tag{6.69}$$

where ϵ_u is an arbitrary small prescribed positive value.

An argument is presented in Snyman and Fatti (1987) to show that when the level set $\{\mathbf{x}|f(\mathbf{x}) \leq f(\mathbf{x}_0^i)\}$ is bounded and $\nabla f(\mathbf{x}_0^i) \neq \mathbf{0}$, then conditions (6.69) above will be satisfied at some finite point in time.

Each computed step along trajectory T^i is monitored so that at termination the point $\mathbf{x}_m^i$ at which the minimum value was achieved is recorded together with the associated velocity $\dot{\mathbf{x}}_m^i$ and function value f_m^i. The values of $\mathbf{x}_m^i$ and $\dot{\mathbf{x}}_m^i$ are used to determine the initial values for the next trajectory T^{i+1}. From a comparison of the minimum values the best point $\mathbf{x}_b^i$, for the current j over all trajectories to date is also recorded. In more detail the minimization procedure for *a given sample point* $\mathbf{x}^j$, in computing the sequence $\mathbf{x}_b^i$, $i = 1, 2, ...,$ is as follows.

Algorithm 6.6 Minimization Procedure MP1

```
1. For given sample point $\mathbf{x}^j$, set $\mathbf{x}_0^1 := \mathbf{x}^j$ and compute $T^1$ subject to $\dot{\mathbf{x}}_0^1 := 0$ ; record $\mathbf{x}_m^1, \dot{\mathbf{x}}_m^1$ and $f_m^1$ ; set $\mathbf{x}_b^1 := \mathbf{x}_m^1$ and $i := 2$,
2. compute trajectory $T^i$ with $\mathbf{x}_0^i := \frac{1}{2}\left(\mathbf{x}_0^{i-1} + \mathbf{x}_b^{i-1}\right)$ and $\dot{\mathbf{x}}_0^i := \frac{1}{2}\dot{\mathbf{x}}_m^{i-1}$, record $\mathbf{x}_m^i, \dot{\mathbf{x}}_m^i$ and $f_m^i$,
3. if $f_m^i < f(\mathbf{x}_b^{i-1})$ then $\mathbf{x}_b^i := \mathbf{x}_m^i$ ; else $\mathbf{x}_b^i := \mathbf{x}_b^{i-1}$,
4. set $i := i + 1$ and go to 2.
```

In the original paper (Snyman and Fatti 1987) an argument is presented to indicate that under normal conditions on the continuity of f and its derivatives, $\mathbf{x}_b^i$ will converge to a local minimum. Procedure MP1, for a given j, is accordingly terminated at step Algorithm 6.6 above if $||\nabla f(\mathbf{x}_b^i)|| \leq \epsilon$, for some small prescribed positive value ϵ, and $\mathbf{x}_b^i$ is taken as the local minimizer $\mathbf{x}_f^j$, i.e. set $\mathbf{x}_f^j := \mathbf{x}_b^i$ with corresponding function value $f_f^j := f(\mathbf{x}_f^j)$.

Reflecting on the overall approach outlined above, involving the computation of energy conserving trajectories and the minimization procedure, it should be evident that, in the presence of many local minima, the probability of convergence to a relative low local minimum is increased. This one expects because, with a small value of ϵ_u (see conditions (6.69)), it is likely that the particle will move through a trough associated with a relative high local minimum, and move over a ridge to record a lower function value at a point beyond. Since we assume that the level set associated with the starting point function is bounded, termination of the search trajectory will occur as the particle eventually moves to a region of higher function values.

6.6.3 The modified bouncing ball trajectory method

The essentials of the modified bouncing ball algorithm using dynamic search trajectories for unconstrained global minimization are now presented. The algorithm is in an experimental stage, and details concerning the motivation of the method, its detailed construction, and computational aspects will be presented in future.

6.6.3.1 Dynamic trajectories

In the MBB algorithm successive sample points $\mathbf{x}^j, j = 1, 2, ...,$ are selected at random from the box D defined by (6.66). For *each* sample point $\mathbf{x}^j$, a sequence of *trajectory steps* $\Delta\mathbf{x}^i$ and associated *projection points* $\mathbf{x}^{i+1}$, $i = 1, 2, ...,$ are computed from the successive analytical relationships (with $\mathbf{x}^1 := \mathbf{x}^j$ and prescribed $V_{0_1} > 0$):

$$\Delta\mathbf{x}^i = V_{0_i} t_i \cos\theta_i \boldsymbol{\nabla} f(\mathbf{x}^i)/||\boldsymbol{\nabla} f(\mathbf{x}^i)|| \tag{6.70}$$

where

$$\theta_i = \tan^{-1}(||\boldsymbol{\nabla} f(\mathbf{x}^i)||) + \frac{\pi}{2}, \tag{6.71}$$

$$t_i = \frac{1}{g}\left[V_{0_i}\sin\theta_i + \left\{(V_{0_i}\sin\theta_i)^2 + 2gh(\mathbf{x}^i)\right\}^{1/2}\right], \tag{6.72}$$

$$h(\mathbf{x}^i) = f(\mathbf{x}^i) + k \tag{6.73}$$

with k a constant chosen such that $h(\mathbf{x}) > 0 \ \forall \ \mathbf{x} \in D$, g a positive constant, and

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \Delta\mathbf{x}^i. \tag{6.74}$$

For the next step, select $V_{0_{i+1}} < V_{0_i}$. Each step $\Delta\mathbf{x}^i$ represents the ground or horizontal displacement obtained by projecting a particle in a vertical gravitational field (constant g) at an elevation $h(\mathbf{x}^i)$ and speed V_{0_i} at an inclination θ_i. The angle θ_i represents the angle that the outward normal $\mathbf{n}$ to the hypersurface represented by $y = h(\mathbf{x})$ makes, at $\mathbf{x}^i$ in $n+1$ dimensional space, with the horizontal. The time of flight t_i is the time taken to reach the ground corresponding to $y = 0$.

More formally, the minimization trajectory for *a given sample point* $\mathbf{x}^j$ and some initial prescribed speed V_0 is obtained by computing the sequence $\mathbf{x}^i$, $i = 1, 2, ...,$ as follows.

Algorithm 6.7 Minimization Procedure MP2

```
1. For given sample point $\mathbf{x}^j$, set $\mathbf{x}^1 := \mathbf{x}^j$ and compute trajectory
   step $\Delta\mathbf{x}^1$ according to (6.70)–(6.73) and subject to $V_{0_1} := V_0$;
   record $\mathbf{x}^2 := \mathbf{x}^1 + \Delta\mathbf{x}^1$, set $i := 2$ and $V_{0_2} := \alpha V_{0_1}$ $(\alpha < 1)$.
2. Compute $\Delta\mathbf{x}^i$ according to (6.70)–(6.73) to give $\mathbf{x}^{i+1} := \mathbf{x}^i + \Delta\mathbf{x}^i$,
   record $\mathbf{x}^{i+1}$ and set $V_{0_{i+1}} := \alpha V_{0_i}$.
3. Set $i := i + 1$ and go to 2.
```

In the vicinity of a local minimum $\hat{\mathbf{x}}$ the sequence of projection points $\mathbf{x}^i$, $i = 1, 2, ...,$ constituting the search trajectory for starting point $\mathbf{x}^j$ will converge since $\Delta\mathbf{x}^i \to 0$ (see (6.70)). In the presence of many local minima, the probability of convergence to a relative low local minimum is increased, since the kinetic energy can only decrease for $\alpha < 1$.

Procedure MP2, for a given j, is successfully terminated if $||\boldsymbol{\nabla} f(\mathbf{x}^i)|| \leq \epsilon$ for some small prescribed positive value ϵ, or when $\alpha V_0^i < \beta V_0^1$, and $\mathbf{x}^i$ is taken as the local minimizer $\mathbf{x}_f^j$ with corresponding function value $f_f^j := h(\mathbf{x}_f^j) - k$.

Clearly, the condition $\alpha V_0^i < \beta V_0^1$ will always occur for $0 < \beta < \alpha$ and $0 < \alpha < 1$.

MP2 can be viewed as a variant of the steepest descent algorithm. However, as opposed to steepest descent, MP2 has (as has MP1) the ability for 'hill-climbing', as is inherent in the physical model on which MP2 is based (viz., the trajectories of a bouncing ball in a conservative gravitational field.) Hence, the behavior of MP2 is quite different from that of steepest descent and furthermore, because of it's physical basis, it tends to seek local minima with relative low function values and is therefore suitable for implementation in global searches, while steepest descent is not.

For the MBB algorithm, convergence to a local minimum is not proven. Instead, the underlying physics of a bouncing ball is exploited. Unsuccessful trajectories are terminated, and do not contribute to the probabilistic stopping criterion (although these points are included in the number of unsuccessful trajectories $\tilde{n}$). In the validation of the algorithm the philosophy adopted here is that the practical demonstration of convergence of a proposed algorithm on a variety of demanding test problems may be as important and convincing as a rigorous mathematical convergence argument.

Indeed, although for the steepest descent method convergence can be proven, in practice it often fails to converge because effectively an infinite number of steps is required for convergence.

6.6.4 Global stopping criterion

The above methods require a termination rule for deciding when to end the sampling and to take the current overall minimum function value $\tilde{f}$, i.e.

$$\tilde{f} = \text{ minimum } \left\{ f_f^j, \text{ over all } j \text{ to date } \right\} \tag{6.75}$$

as an approximation of the global minimum value f^*.

Define the *region of convergence* of the dynamic methods for a local minimum $\hat{\mathbf{x}}$ as the set of all points $\mathbf{x}$ which, used as starting points for the above procedures, converge to $\hat{\mathbf{x}}$. One may reasonably expect that in the case where the *regions of attraction* (for the usual gradient-descent methods, see Dixon et al. 1976) of the local minima are more or less equal, that the region of convergence of the global minimum will be relatively increased.

Let R_k denote the region of convergence for the above minimization procedures MP1 and MP2 of local minimum $\hat{\mathbf{x}}^k$ and let α_k be the associated probability that a sample point be selected in R_k. The region of convergence and the associated probability for the global minimum $\mathbf{x}^*$

are denoted by R^* and α^* respectively. The following basic assumption, which is probably true for many functions of practical interest, is now made. BASIC ASSUMPTION:

$$\alpha^* \geq \alpha_k \text{ for all local minima } \hat{\mathbf{x}}^k. \tag{6.76}$$

The following theorem may be proved.

6.6.4.1 Theorem (Snyman and Fatti 1987)

Let r be the number of sample points falling within the region of convergence of the current overall minimum $\tilde{f}$ after $\tilde{n}$ points have been sampled. Then under the above assumption and a statistically non-informative prior distribution the probability that $\tilde{f}$ corresponds to f^* may be obtained from

$$Pr\left[\tilde{f} = f^*\right] \geq q(\tilde{n}, r) = 1 - \frac{(\tilde{n}+1)!(2\tilde{n}-r)!}{(2\tilde{n}+1)!(\tilde{n}-r)!}. \tag{6.77}$$

On the basis of this theorem the *stopping rule* becomes: STOP when $Pr\left[\tilde{f} = f^*\right] \geq q^*$, where q^* is some prescribed desired confidence level, typically chosen as 0.99.

Proof:

We present here an outline of the proof of (6.77), and follow closely the presentation in Snyman and Fatti (1987). (We have since learned that the proof can be shown to be a generalization of the procedure proposed by Zielińsky 1981.) Given $\tilde{n}^*$ and α^*, the probability that at least one point, $\tilde{n} \geq 1$, has converged to f^* is

$$\Pr[\tilde{n}^* \geq 1 | \tilde{n}, r] = 1 - (1-\alpha^*)^{\tilde{n}}\ . \tag{6.78}$$

In the Bayesian approach, we characterize our uncertainty about the value of α^* by specifying a prior probability distribution for it. This distribution is modified using the sample information (namely, $\tilde{n}$ and r) to form a posterior probability distribution. Let $p_*(\alpha^*|\tilde{n}, r)$ be the posterior probability distribution of α^*. Then,

$$\begin{aligned}\Pr[\tilde{n}^* \geq 1|\tilde{n}, r] &= \int_0^1 \left[1-(1-\alpha^*)^{\tilde{n}}\right] p_*(\alpha^*|\tilde{n}, r) d\alpha^* \\ &= 1 - \int_0^1 (1-\alpha^*)^{\tilde{n}} p_*(\alpha^*|\tilde{n}, r) d\alpha^*. \qquad (6.79)\end{aligned}$$

Now, although the r sample points converge to the current overall minimum, we do not know whether this minimum corresponds to the global minimum of f^*. Utilizing (6.76), and noting that $(1-\alpha)^{\tilde{n}}$ is a decreasing function of α, the replacement of α^* in the above integral by α yields

$$\Pr[\tilde{n}^* \geq 1|\tilde{n}, r] \geq \int_0^1 \left[1-(1-\alpha)^{\tilde{n}}\right] p(\alpha|\tilde{n}, r) d\alpha \; . \qquad (6.80)$$

Now, using Bayes theorem we obtain

$$p(\alpha|\tilde{n}, r) = \frac{p(r|\alpha, \tilde{n})p(\alpha)}{\int_0^1 p(r|\alpha, \tilde{n})p(\alpha)d\alpha} \; . \qquad (6.81)$$

Since the $\tilde{n}$ points are sampled at random and each point has a probability α of converging to the current overall minimum, r has a binomial distribution with parameters α and $\tilde{n}$. Therefore

$$p(r|\alpha, \tilde{n}) = \binom{\tilde{n}}{r} \alpha^r (1-\alpha)^{\tilde{n}-r} \; . \qquad (6.82)$$

Substituting (6.82) and (6.81) into (6.80) gives:

$$\Pr[\tilde{n}^* \geq 1|\tilde{n}, r] \geq 1 - \frac{\int_0^1 \alpha^r (1-\alpha)^{2\tilde{n}-r} p(\alpha) d\alpha}{\int_0^1 \alpha^r (1-\alpha)^{\tilde{n}-r} p(\alpha) d\alpha} \; . \qquad (6.83)$$

A suitable flexible prior distribution $p(\alpha)$ for α is the beta distribution with parameters a and b. Hence,

$$p(\alpha) = [1/\boldsymbol{\beta}(a,b)]\, \alpha^{a-1}(1-\alpha)^{b-1}, \qquad 0 \leq \alpha \leq 1. \qquad (6.84)$$

Using this prior distribution gives:

$$\begin{aligned}\Pr[\tilde{n}^* \geq 1|\tilde{n}, r] &\geq 1 - \frac{\Gamma(\tilde{n}+a+b)\;\Gamma(2\tilde{n}-r+b)}{\Gamma(2\tilde{n}+a+b)\;\Gamma(\tilde{n}-r+b)} \\ &= 1 - \frac{(\tilde{n}+a+b-1)!\;(2\tilde{n}-r+b-1)!}{(2\tilde{n}+a+b-1)!\;(\tilde{n}-r+b-1)!}.\end{aligned}$$

Assuming a prior expectation of 1, (viz. $a = b = 1$), we obtain

$$\Pr[\tilde{n}^* \geq 1 | \tilde{n}, r] = 1 - \frac{(\tilde{n}+1)!\,(2\tilde{n}-r)!}{(2\tilde{n}+1)!\,(\tilde{n}-r)!},$$

which is the required result. □

6.6.5 Numerical results

No.	Name	ID	n	Ref.
1	Griewank G1	G1	2	Törn and Zilinskas; Griewank
2	Griewank G2	G2	10	Törn and Zilinskas; Griewank
3	Goldstein-Price	GP	2	Törn and Zilinskas; Dixon and Szegö
4	Six-hump Camelback	C6	2	Törn and Zilinskas; Branin
5	Shubert, Levi No. 4	SH	2	Lucidl and Piccioni
6	Branin	BR	2	Törn and Zilinskas; Branin and Hoo
7	Rastrigin	RA	2	Törn and Zilinskas
8	Hartman 3	H3	3	Törn and Zilinskas; Dixon and Szegö
9	Hartman 6	H6	6	Törn and Zilinskas; Dixon and Szegö
10	Shekel 5	S5	4	Törn and Zilinskas; Dixon and Szegö
11	Shekel 7	S7	4	Törn and Zilinskas; Dixon and Szegö
12	Shekel 10	S10	4	Törn and Zilinskas; Dixon and Szegö

Table 6.6: The test functions

		SF - This Study			SF - Previous		MBB		
No.	ID	N_f	$(r/\tilde{n})_b$	$(r/\tilde{n})_w$	N_f	$r/\tilde{n}$	N_f	$(r/\tilde{n})_b$	$(r/\tilde{n})_w$
1	G1	4199	6/40	6/75	1606	6/20	2629	5/8	6/23
2	G2	25969	6/84	6/312	26076	6/60	19817	6/24	6/69
3	GP	2092	4/4	5/12	668	4/4	592	4/4	5/10
4	C6	426	4/4	5/9	263	4/4	213	4/4	5/10
5	SH	8491	6/29	6/104	—	—	1057	5/7	6/26
6	BR	3922	4/4	5/12	—	—	286	4/4	5/6
7	RA	4799	6/67	6/117	—	—	1873	4/4	6/42
8	H3	933	4/4	5/8	563	5/6	973	5/9	6/29
9	H6	1025	4/4	5/10	871	5/8	499	4/4	5/9
10	S5	1009	4/4	6/24	1236	6/17	2114	5/8	6/39
11	S7	1057	5/8	6/37	1210	6/17	2129	6/16	6/47
12	S10	845	4/4	6/31	1365	6/20	1623	5/7	6/39

Table 6.7: Numerical results

	Test Function					
Method	BR	C6	GP	RA	SH	H3
TRUST	55	31	103	59	72	58
MBB	25	29	74	168	171	24

Table 6.8: Cost (N_f) using *a priori* stopping condition

The test functions used are tabulated in Table 6.6, and tabulated numerical results are presented in Tables 6.7 and 6.8. In the tables, the reported number of function values N_f are the average of 10 independent (random) starts of each algorithm.

Unless otherwise stated, the following settings were used in the SF algorithm (see Snyman and Fatti 1987): $\gamma = 2.0$, $\alpha = 0.95$, $\epsilon = 10^{-2}$, $\omega = 10^{-2}$, $\delta = 0.0$, $q^* = 0.99$, and $\Delta t = 1.0$. For the MBB algorithm, $\alpha = 0.99$, $\epsilon = 10^{-4}$, and $q^* = 0.99$ were used. For each problem, the initial velocity V_0 was chosen such that $\Delta \mathbf{x}^1$ was equal to half the 'radius' of the domain D. A local search strategy was implemented with varying α in the vicinity of local minima.

In Table 6.7, $(r/\tilde{n})_b$ and $(r/\tilde{n})_w$ respectively indicate the best and worst $r/\tilde{n}$ ratios (see equation (6.77)), observed during 10 independent optimization runs of both algorithms. The SF results compare well with the previously published results by Snyman and Fatti, who reported values for a single run only. For the Shubert, Branin and Rastrigin functions, the MBB algorithm is superior to the SF algorithm. For the Shekel functions (S5, S7 and S10), the SF algorithm is superior. As a result of the stopping criterion (6.77), the SF and MBB algorithms found the global optimum between 4 and 6 times for each problem.

The results for the trying Griewank functions (Table 6.7) are encouraging. G1 has some 500 local minima in the region of interest, and G2 several thousand. The values used for the parameters are as specified, with $\Delta t = 5.0$ for G1 and G2 in the SF-algorithm. It appears that both the SF and MBB algorithms are highly effective for problems with a large number of local minima in D, and problems with a large number of design variables.

In Table 6.8 the MBB algorithm is compared with the deterministic

TRUST algorithm (Barhen et al. 1997). Since the TRUST algorithm was terminated when the global approximation was within a specified tolerance of the (known) global optimum, a similar criterion was used for the MBB algorithm. The table reveals that the two algorithms compare well. Note however that the highest dimension of the test problems used in Barhen et al. (1997) is 3. It is unclear if the deterministic TRUST algorithm will perform well for problems of large dimension, or problems with a large number of local minima in D.

In conclusion, the numerical results indicate that both the Snyman-Fatti trajectory method and the modified bouncing ball trajectory method are effective in finding the global optimum efficiently. In particular, the results for the trying Griewank functions are encouraging. Both algorithms appear effective for problems with a large number of local minima in the domain, and problems with a large number of design variables. A salient feature of the algorithms is the availability of an apparently effective global stopping criterion.