Chapter 1

AN INTRODUCTION TO DYNAMICAL SEARCH

Luc Pronzato
Laboratoire I3S, CNRS/Université de Nice-Sophia Antipolis, France
pronzato@i3s.unice.fr

Henry P. Wynn
Dept. of Statistics, University of Warwick, UK
hpw@stats.warwick.ac.uk

Anatoly A. Zhigljavsky
School of Mathematics, Cardiff University, UK
ZhigljavskyAA@Cardiff.ac.uk

1. INTRODUCTION

The name dynamical search was introduced by the authors in the monograph Pronzato *et al* (2000) following a series of papers on special aspects of the area. Although this monograph is a reasonably comprehensive summary of the area, it was considered worthwhile giving a more leisurely introduction showing together something approaching a coherent definition of the field and giving some alternative constructions.

The starting point for the theory is a particular type of optimisation or search algorithm which is *set based*. That is to say, instead of a discrete set of points $\{x_i\}$ in some set which converges to a solution x^*, we considered a collection of sets $\{S_i\}$ typically containing x^* (though not always) and such that the size of S_i (diameter, volume, etc.) converges to zero. Thus at iteration i, the set S_i, or its boundary, provides some kind of approximant for x^*.

The second half of the theory arises from creating a dynamical system from this basic setup.

Let us proceed with a canonical example, which was very much the starting point for the theory and holds some of its salient features. This example belongs to the rich class of examples arising from minimisation of a uniextremal function

$f(\cdot)$ on some interval X using a 'second-order' line-search algorithm. The general 'second-order' line-search algorithm, in its renormalised form, compares function values at two points, first, e_n, from the previous iteration and second, e'_n, selected by the algorithm at the current iteration. Let $X_0 = [0,1]$. Any choice for $e_1 \in [0,1]$ and any function $\psi(.) : [0,1] \longrightarrow [0,1]$, with $e'_n = \psi(e_n)$ then defines a second-order algorithm. Define

$$u_n = \min\{e_n, e'_n\}, \quad v_n = \max\{e_n, e'_n\},$$

then the deletion rule is:

$$\begin{cases} \text{(R)}: & \text{if } f_n(u_n) < f_n(v_n) \text{ delete } (v_n, 1] \\ \text{(L)}: & \text{if } f_n(u_n) \geq f_n(v_n) \text{ delete } [0, u_n), \end{cases}$$

where $f_n(.)$ is the renormalised function; (R) and (L) stand for right and left deletion. The remaining interval is then renormalised to $[0,1]$.

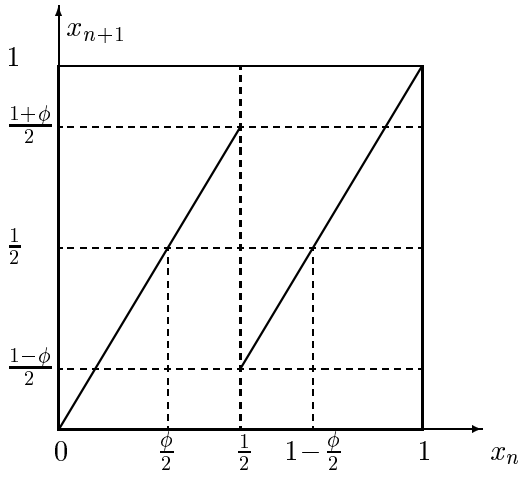

Figure 1.1 The Golden Section iteration.

The *Golden Section algorithm* corresponds to

$$v_n = 1 - u_n = \phi = \frac{\sqrt{5}-1}{2} \simeq 0.61804,$$

i.e. to $e'_n = 1 - e_n$, with $e_1 = \phi$. In the special case when $f(x)$ is symmetric around x^* the algorithm yields the time homogeneous dynamic process $x_{n+1} = h(x_n)$, where $x_1 = x^*$ and

$$h(x_n) = \begin{cases} x_n(1+\phi) & \text{if} \quad x_n < \frac{1}{2} \\ x_n(1+\phi) - \phi & \text{if} \quad x_n \geq \frac{1}{2} \end{cases}. \tag{1.1}$$

Figure 1.1 shows this transformation.

The invariant density $p(x)$ for this dynamical system can be easily computed and is shown in Figure 1.2. Some other second-order line search algorithms are presented in Section 6.

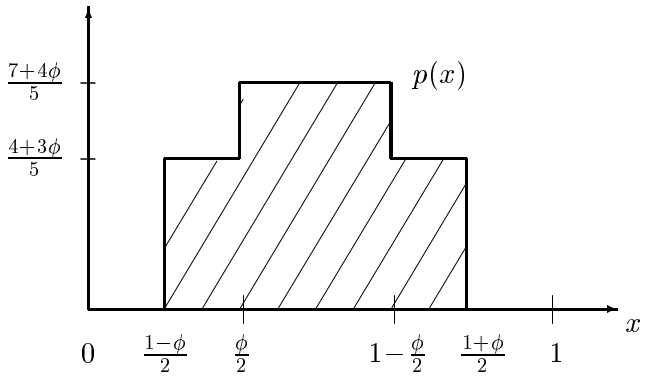

Figure 1.2 The graph of the invariant density for the Golden Section algorithm.

Let us now give a tighter, though not exhaustive, description of the field. Following the above discussion we divide the description of a dynamical search problem into two halves.

2. SEARCH PROBLEM

A search problem can be characterized with the following objects.

X : a space in which the algorithm operates;

x^* : a distinguished point in X, called the 'solution.' We may sometimes consider, rather, a distinguished set $X^* \subseteq X$.

$\{S_i\}_0^\infty$: A set of subsets of X. Very often but not always these are nested so that

$$S_i \supseteq S_{i+1} \quad (i = 0, 1, \ldots)$$

F: a class of functions on X;

$f(\cdot)$: a function in F, usually with some connection to x^*. For example,

$$x^* = \text{argmin}_{x \in X} f(x) \quad \text{or} \quad x^* = \{x | f(x) = 0\}$$

$\{x_i\}$: a set of observation points (often approaching x^*);

y_i : an observation. This may be simply

$$y_i = f(x_i)$$

but it may alternatively be an indicator function describing some feature of $f(x_i)$, or comparisons between a 'window' of $f(x_i)$ values and so on.

Given this main notation we define an algorithm as an iterative scheme which given the current data $\{x_i, y_i\}_{i=0}^n$ computes the next approximating set S_{n+1}.

In reality this will be a construction, sometimes quite complex, which uses all the current information

$$\{\{x_i, y_i\}_{i=0}^n, \{S_i\}_{i=0}^n\}$$

to produce x_{n+1} first and then an observation is made at the specified x_{n+1} giving

$$\{x_{n+1}, y_{n+1}, S_{n+1}\} \ .$$

Let us link this definition to the Golden Section algorithm.

X : $[0, 1]$;

x^* : unique minimizing point of $f(\cdot)$

$\{S_i\}$: current uncertainty interval;

F: a class of uniextremal functions on X;

$f(\cdot)$: a function in F;

$\{x_i\}$: a set of observation points;

y_i : indicator

$$y_i = \begin{cases} 1 & \text{if } f(x_i) \leq f(x_{i-1}) \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm is clear: the new observation point x_{i+1} is the new left or right Golden Section point and the transition $S_i \to S_{i+1}$ is obtained by the deletion.

We now discuss the construction of the dynamical system. At this stage we describe the most basic version when the S_i are nested. There is only one basic idea which is of renormalisation. We need two new objects.

X_0 : A canonical region. This may be the same as the starting region S_0;

g_i : a renormalisation taking the current S_i to X_0:

$$g_i(S_i) = X_0$$

The dynamical system is then derived automatically as

$$x_i^* = g_i(x^*)$$

The system then has trajectory

$$x_0^* \to x_1^* \to \ldots \to x_n^* \to$$

and lives entirely in X_0.

The subject of dynamical search consists of relating convergence of the original algorithm to properties of the dynamical system. Very roughly, the construction of the S_i links to the rate of expansion of the map

$$x_i^* \to x_{i+1}^* \, .$$

In making the connection perhaps the most important object to define is the rate of convergence. The set-based nature of the algorithms has led to the extension of the usual definition to a range of types of convergence which link, then, to classical rates of expansion of Lyapunov type. We shall introduce, in particular, rates based on Renyi entropy of certain partitions generated by the dynamical systems.

3. CONSISTENCY AND UPDATING

Some of the most natural aspects of the set-based algorithms arise from the geometry of updating, that is, the transition $S_i \to S_{i+1}$. Within this the concept of *consistency* is key.

A consistent set $X_i \subseteq X$ is defined as a set of all $x \in X$ consistent with the current data. Thus, let $\{x_i, y_i\}_{i=0}^n$ be the data in the algorithm up to n with the true $\{f(\cdot), x^*\}$ and let $\{\tilde{x}_i, \tilde{y}_i\}_{i=0}^n$ be the data which would have been produced by an alternative pair $\{\tilde{f}(\cdot), \tilde{x}^*\}$, $\tilde{f}(\cdot) \in F$. Then we can define

$$X_n = \{\tilde{x}^* |\ (\tilde{x}_i, \tilde{y}_i) = (x_i, y_i), \quad i = 0, 1, \ldots, n\}$$

We see in the Golden Section algorithm that the new region obtained is precisely the set of x^* consistent with the assumption of unimodality and the current data.

Consistency is often used in global optimization algorithms based on set covering. The idea is as follows. Assume that $f(.)$ satisfies a Lipschitz condition of the type

$$\forall (x, x') \in X \times X \, , \ |f(x) - f(x')| \leq M \|x - x'\| \, .$$

Then, having evaluated $f(.)$ at $x^{(1)}, \ldots, x^{(k)}$, with

$$f(\hat{x}^{(k)}) = \min_{n=1,\ldots,k} f(x^{(n)}) \, ,$$

one knows that x^* does not belong to the union $\cup_{n=1}^{k} \mathcal{B}(x^{(n)}, \rho^{(n)})$, where

$$\mathcal{B}(x, \varepsilon) = \{z \in X : ||x - z|| \leq \varepsilon\}$$

for any $x \in X$ and $\varepsilon > 0$, and

$$\rho^{(n)} = [f(x^{(n)}) - f(\hat{x}^{(k)})]/M \, .$$

The consistency set for x^* is thus

$$S_k = \overline{\cup_{n=1}^{k} \mathcal{B}(x^{(n)}, \rho^{(n)})} \, .$$

The points $x^{(n)}$ can be generated sequentially or not. A possible sequential approach (active covering) is to select $x^{(k+1)}$ as the minimizer of the Lipschitzian minorant $f^{(k)}(.)$ of $f(.)$,

$$f^{(k)}(x) = \max_{n=1,\ldots,k} (f(x^{(n)}) - M\|x - x^{(n)}\|) \, ,$$

as shown in Figure 1.3.

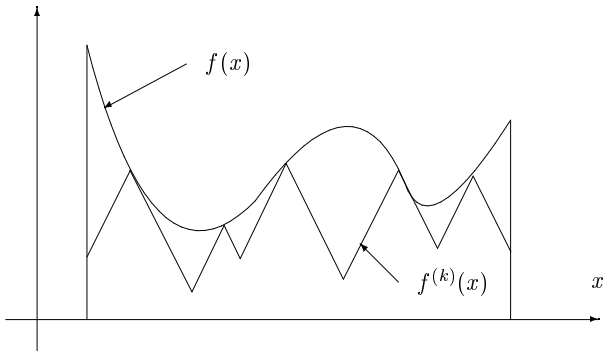

Figure 1.3 $f(\cdot)$ and the lower bounding function $f^{(k)}(\cdot)$.

However, this point $x^{(k+1)}$ is difficult to determine when the dimension of the space $d > 1$, since the graph of $f^{(k)}(\cdot)$ is obtained by intersecting cones in the $(d+1)$-dimensional space, see e.g. the survey paper Hansen and Jaumard (1995). In nonsequential methods (passive covering) the sequence $\{x^{(n)}\}$ does not depend on the observed values of $f(.)$ and can be generated beforehand. Random or quasi–random (based on space–filling curves) sequences can be used, see e.g. Boender and Romeijn (1995). Note that $f(x^{(k)})$ need not be evaluated when the k-th point in the sequence does not belong to the current consistency set. One may refer to Zhigljavsky and Chekmasov (1996) for a comparison between different covering schemes. In practice the Lipschitz constant can be estimated in the course of optimisation, see e.g. Hansen and

Jaumard (1995) or Zhigljavsky (1991). In this case, the consistency sets are not necessarily imbedded.

In many algorithms we simply set

$$S_i = X_i \ni x^* .$$

In such cases we have

$$g_i(X_i) = X_0$$

so that we transform the consistent region to the canonical region X_0. This means that the updating

$$x_i^* \to x_{i+1}^*$$

is intimately related to the updating of the consistent region

$$X_i \to X_{i+1} .$$

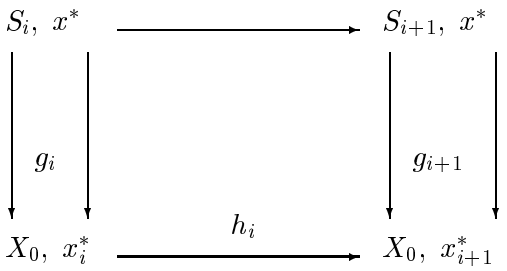

Figure 1.4 Renormalisation.

Consider the following diagram (Figure 1.4). Assume that all the g_i's are invertible. Then

$$x^* = g_i^{-1}(x_i^*) = g_{i+1}^{-1}(x_{i+1}^*)$$

and, if this holds for all x^* in the original region S_0, then in X_0

$$x_{i+1}^* = g_{i+1}\, g_i^{-1}(x_i^*) .$$

This gives the basic updating for the system: writing $h_i = g_{i+1} g_i^{-1}$ we have

$$x_{i+1}^* = h_i(x_i^*)$$

Now suppose we require that the diagram is valid on X_0, then

$$X_0 = g_i(S_i) = g_{i+1}(S_{i+1})$$

and

$$S_{i+1} = g_{i+1}^{-1} g_i(S_i) = \tilde{h}_i(S_i)\,, \quad \text{say.}$$

This gives the updating formula for the S_i, the core of the algorithm. If moreover S_i are the consistent sets then

$$X_{i+1} = g_{i+1}^{-1} g_i(X_i)\,.$$

The nicest cases are where $h_i = g_{i+1} g_i^{-1}$ is time homogeneous that is not dependent in i. In this case we write simply

$$h = g_{i+1} g_i^{-1}. \tag{1.2}$$

It is important to interpret the inverses g_i^{-1} as acting locally. This is because the g_i's used in any realization of the algorithms will be dependent on x^*. If this is understood, we can give a simple proof of the time homogeneuity of that $\tilde{h}_i$ and h_i are equivalent under the condition that g_i and g_{i+1} commute. This follows simply from the fact that in this case $\tilde{h}_i = h_i^{-1}$ locally.

4. PARTITIONS AND RATES

4.1 PARTITIONS

For fixed x^* the algorithm generates nested subsets which themselves depend on x^*. Different x^* may yield the same sequence up to iteration n but in the limit different x^* will normally generate different sequences.

At fixed iteration n let $S_n = S_n(x^*)$ be the nth approximant set given x^*. Then

$$S_0 = \bigcup_{x^* \in S_0} S_n(x^*)\,.$$

If $S_n(x^*)$ is the consistent set $X_n(x^*)$ for x^* (given the background conditions) then

$$S_0 = \bigcup_{x^* \in S_0} X_n(x^*)\,. \tag{1.3}$$

As x^* ranges over S_0 we have disjoint sets corresponding to all x^* with some $X_n(x^*)$ and (1.3) represents a partition of S_0. In general the partition (1.3) into the consistency sets always gives a disjoint partition whether or not the $S_n(x^*)$ are disjoint.

More importantly, for the analysis we can work, rather, with the dynamical system. Furthermore, the simple analysis given in terms of $h_i : x_i \to x_{i+1}$ with $x_1 = x^*$ may be insufficient to make the dynamical system. We typically extend the 'state-space' of the system with additional state variables say e and define the system as

$$\{x_i, e_i\} \to \{x_{i+1}, e_{i+1}\}\,.$$

It is then possible to relate the simple partition in terms of consistent sets for the original algorithm to the partition of the trajectories of the dynamical system in the formal sense. This allows definition of rates of expansion which can be related to the rates of contraction of the S_n (or X_n) in the original algorithm.

4.2 ERGODIC CONVERGENCE RATE

These rates of contraction are measured in terms of some type of volume and we have two competing measures vol(S_n) and vol(X_n). In all the algorithms devised $X_n \subseteq S_n$ and therefore vol$(X_n) \leq$ vol(S_n).

Define $L_n =$ vol(S_n). Then the local rate is defined as

$$r_n = r_n(x^*) = \frac{L_{n+1}}{L_n} .$$

The asymptotic rate is then defined as

$$R = R(x^*) = \lim_{N\to\infty} [L_N(x^*)]^{\frac{1}{N}} = \lim_{N\to\infty} [L_0 \prod_{n=0}^{N-1} r_n(x^*)]^{\frac{1}{N}} .$$

If this limit exists and is the same for almost all x^* with respect to the Lebesgue measure, then R will be called the *ergodic convergence rate*. Since $L_0 < \infty$ this becomes

$$R = \lim_{N\to\infty} (\prod_{n=1}^{N} r_n)^{\frac{1}{N}} , \tag{1.4}$$

and the logarithmic form (called *log–rate*) is

$$\varrho = -\log R = -\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \log r_n .$$

An important question concerns the relation between the log–rate ϱ and the *Lyapunov exponents* Λ_i of the corresponding dynamical system. The renormalisation is typically *affine* with respect to x^*, that is the renormalised location of x^* in S_n satisfies

$$x_n = g_n(x^*) = \Omega_n x^* + \omega_n ,$$

where the full–rank matrix Ω_n and the vector ω_n may depend on some other state variables θ_n of the dynamical system. The variables θ_n do not depend explicitely on x^* since they are related to known characteristics of the objective function or to parameters of the search algorithm. Then the dynamic process is $z_{n+1} = (x_{n+1}, \theta_{n+1}) = T(x_n, \theta_n)$, with, from (1.2),

$$x_{n+1} = \Omega_{n+1}\Omega_n^{-1} x_n - \Omega_{n+1}\Omega_n^{-1}\omega_n + \omega_{n+1} ,$$

and θ_{n+1} not depending explicitly on x_n. Define $\Sigma_n = d\theta_{n+1}/d\theta_n^T$. Then, as shown in Pronzato *et al* (2000), Theorem 4.1, if the matrices $\Omega_k\Omega_1^{-1}$ and $\prod_{i=1}^{k}\Sigma_i$ have real eigenvalues for all k and the Lyapunov exponents of the dynamic system $T(\cdot)$ are well defined, then the log–rate ϱ is equal to the sum of d Lyapunov exponents of $T(\cdot)$, with $d = \dim x$.

4.3 CHARACTERISTICS OF AVERAGE PERFORMANCES

As already noticed, $L_n = \mathrm{vol}(S_n)$ depends on the value of x^*. Adopting a Bayesian point of view, we can assume that x^* has a prior distribution $\mu_0(\cdot)$ on S_0, and $g_1(\cdot)$ thus induces some distribution $\mu(\cdot)$ for x_1.

Since the dynamical process $\{z_n\}$ may depend on some other characteristics, i.e. $z_n = (x_n, \theta_n)$, with θ_n corresponding to characteristics of the objective function $f(\cdot)$ or to parameters of the algorithm; θ_1 is thus known. The expectations considered below are then with respect to x_1 conditional on θ_1.

In the case of optimization of a locally symmetric function, the dynamical process depends on the (non–parametric) function itself. The objective function can then be written as

$$f(x) = h(x - x^*)\,,$$

with $h(0) = \min h(\cdot)$. When taking expectations, we then consider $h(\cdot)$ as fixed and perform expectation with respect to x^* only.

We can define the following performance characteristics:

$$E\log L_n = E_x\{\log L_n(x)\}\,,$$

$$\log EL_n = \log E_x\{L_n(x)\}\,,$$

and more generally

$$\log EL_n^\alpha = \log E_x\{L_n^\alpha(x)\} \quad \text{for } \alpha > -1\,.$$

We shall also consider the asymptotic versions of these characteristics:

$$W_1 = -\lim_{n\to\infty}\frac{1}{n}E\log L_n\,,$$

$$W_2 = -\lim_{n\to\infty}\frac{1}{n}\log EL_n\,,$$

and more generally

$$W_\gamma = \frac{1}{1-\gamma}\lim_{n\to\infty}\frac{1}{n}\log EL_n^{\gamma-1}\,,\ \gamma \neq 1\,,\ \gamma \geq 0\,,$$

if the limits exist. From the authors' point of view, the behaviour of W_γ as a function of γ reflects the most essential asymptotic features of the original algorithm.

Note that Jensen's inequality imply

$$\log EL_n \geq E \log L_n \,, \; \forall n \geq 1 \,,$$

and thus $W_1 \geq W_2$ when W_1 and W_2 exist. More generally, Hölder inequality guarantees $W_\gamma \geq W_{\gamma'}$ for $\gamma < \gamma'$ when W_γ and $W_{\gamma'}$ exist.

In many cases W_γ is just the entropy of the associated dynamical system defined through the sequence of the Renyi entropies of order γ of the corresponding partitions, see Pronzato *et al* (2000) and (1997b). Examples of optimisation algorithms where this characteristic is evaluated analytically can be found in Pronzato *et al* (1997a, 1998, 2000).

4.4 CHARACTERISTICS OF THE WORST–CASE PERFORMANCE AND COUNTING CHARACTERISTICS

Considering the worst–case performances with respect to $x^* \in S_0$, one can define

$$\mathbf{M}L_n = \sup_{x^*} L_n(x^*) \,,$$

and, provided that the limit exists,

$$W_\infty = - \lim_{n \to \infty} \frac{1}{n} \log \mathbf{M}L_n \,.$$

The examples of the optimisation algorithms when this characteristic is evaluated analytically, can be found in Pronzato *et al* (1997a), Pronzato *et al* (1998), Pronzato *et al* (2000), and Wynn and Zhigljavsky (1993).

The uncertainty region S_n obtained at iteration n depends on x^*. However, certain values of x^* lead to the same region $S_n(x^*)$. One can thus count the number of different regions that can be obtained when $x^* \in S_0$. Denote this number by N_n. The asymptotic version of this characteristic is:

$$\varrho_0 = \lim_{n \to \infty} \frac{1}{n} \log N_n \,,$$

provided the limit exists. For a wide class of line–search algorithms $\varrho_0 = W_0$, which coincides with the topological entropy of the dynamical system associated with the algorithm.

5. DYNAMICAL SYSTEM REPRESENTATION

5.1 SETTING UP THE DYNAMICAL SYSTEM

As mentioned in Section 3. it is often the case that the simple iteration $h_i : x_i \to x_{i+1}$ is not enough to obtain a first order time-invariant dynamical system. Other information needs to be carried forward.

A simple rule for capturing the additional information is to carry forward in addition to x_{i+1} and S_i the additional restriction on $f(\cdot)$ obtained from the requirement of consistency with the data. Recall that we work with a class of function F. Define

$$F_n = \{f' \in F |\ (x'_i, y'_i) = (x_i, y_i),\ \ i = 0, 1, \ldots, n\},$$

where as before x'_i, y'_i is the data which would have been obtained using f'.

A useful approach is not to work with the class F_n but with the class induced on X_0 by the function g_n, namely F_n^* defined by

$$F_n^* = \{f^* \in F |\ f^*(g_n(x)) = f(x),\ f \in F_n\}.$$

This is, in terms of the dynamical system, a minimal representation of F_n^*, that is a minimal set of 'independent' quantities allowing the updating $F_n^* \to F_{n+1}^*$ on X_0. Thus, the state space of the dynamical system is some kind of minimal representation of the triple

$$\{x_n^*, S_n, F_n^*\}$$

sufficient for the reconstruction of this triple.

Then we have a clearer idea of a time homogeneous system, namely

$$(x_{n+1}^*, S_{n+1}, F_{n+1}^*) = H(x_n^*, S_n, F_n^*)$$

for some function H which does not depend on n. This approach is well known in system theory under the name of state-space methods. These processes which perhaps appear not to have a simple one-step time independent shift are converted to a state-space representation with a one-step transformation. In the stochastic case this typically gives a Markov chain representation. In fact, in the current case the Markovian representation sometimes applies, where the transition probabilities are represented by the invariant measure of the process.

The benefits of the conversion of search and optimisation algorithms to dynamical systems arise from the access it gives to a machinery for the computation of rates not normally employed in optimisation. The methods range from purely analytic exact computation to cruder simulation methods with certain semi-analytic methods in between. In some cases we can take a standard algorithm, do the calculations of rate, try to speed up the algorithm by some relaxation method, recompute the rate and show some improvement.

In Sections 6,7, and 8 we outline three classes of examples to illustrate these ideas. Note that some of the material of these sections can be found in the monograph Pronzato *et al* (2000) and one can also find many other related articles by the authors provided in the list of references; one can find many other examples in these references.

5.2 GUARANTEEING THE SUPPORT BY INITIAL EXPANSION

In both the optimal line-search methods and the adaptation of the line-search to the ellipsoidal algorithm a judicious improvement can be made to guarantee that the algorithms do not stop by loosing consistency, as mentioned in the last section. This is related to the support of the dynamical system x_n achieved by renormalisation. The heuristic description of the method has three steps.

First, one notes that some original algorithm has an x_n process whose support is smaller than the full standard region. Based on this observation one seeks to accelerate the algorithm with some optimistic version. Taking a deep cut in the ellipsoidal algorithm is an example.

Second, one observes that over-optimism can lead to the algorithm ceases to converge because the consistent region looses x^*. This loss of x^* corresponds to the support for the x_n being larger than the standard region.

Third, if $\tilde{X}$ is the support of the $\{x_n\}$ process, one can seek to extend the initial region S_1 so that $X \subset S_1$ and x_1 is automatically placed within $\tilde{X}$. It happens that this particular initialization procedure improves the finite sample performance of the algorithm, but also some asymptotic performance characteristics, see Sections 6.4 and 6.5.

6. SECOND-ORDER LINE SEARCH

6.1 GENERAL SCHEME AND RENORMALISATION

A practically tractable class of algorithms is line search algorithms. At each iteration we assume that x^*, the optimising point, lies in an interval $[A_n, B_n]$. At the next iteration a deletion

$$x^* \in [A_{n+1}, B_{n+1}] \subset [A_n, B_n]$$

is made. Depending on the nature of the problem $[A_n, B_n]$ may be a consistency region calculated from the previous information about an objective function, or the starting interval, often $[0, 1]$.

The nature of the information is critical. For the interval based algorithms of the above kind rather than observing $f(x_n)$ directly we may observe the indicator function of some event. For example, for finding the zero of some monotonic function $f(x^*) = 0$ one may simply receive whether $f(x_n) \leq 0$ or > 0. Another important class of algorithms are the so-called second-order line-search algorithms for minimising a uniextremal function. Let us describe the problem in a formal way.

Let the objective function $f(.)$ be given on an interval $[A_0, B_0]$ and x^* be the unknown point at which $f(.)$ is minimum; let $f(.)$ be decreasing for $x \leq x^*$

and non–decreasing for $x > x^*$ (or non–increasing for $x \leq x^*$ and increasing for $x > x^*$).

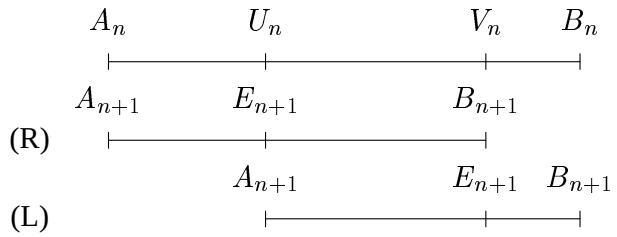

Figure 1.5 One iteration in a second–order line search algorithm

At iteration n we compare the values of $f(.)$ at two points U_n and V_n in the current uncertainty interval $S_n = [A_n, B_n)$, with $U_n < V_n$. The points are $U_n = \min\{E_n, E'_n\}$ and $V_n = \max\{E_n, E'_n\}$, where E_n is the point carried from the previous iteration and E'_n is being selected at the current iteration. Then, if $f(U_n) \geq f(V_n)$ we delete the segment $[A_n, U_n)$, otherwise we delete $[V_n, B_n)$. The remaining part of the interval defines the uncertainty interval $[A_{n+1}, B_{n+1})$ for the next iteration. Either U_n or V_n belongs to $[A_{n+1}, B_{n+1})$; this point is E_{n+1}. Figure 1.5; illustrates this; in this figure, (R) and (L) stand respectively for Right and Left deletion.

A second–order line–search algorithm is therefore defined by:

(i) the initial uncertainty interval $[A_1, B_1)$ that contains $[A_0, B_0)$,

(ii) the initial point $E_1 \in [A_1, B_1)$,

(iii) the selection rule for E'_{n+1}, $n \geq 0$.

An important aspect of these algorithms is that the choice of $[A_1, B_1) \supseteq [A_0, B_0)$, which corresponds to an expansion of the initial uncertainty interval, affects the initialization of the corresponding dynamic system and has strong influence on some of the performance characteristics W_γ.

After either left or right deletion, we renormalise each uncertainty interval $[A_n, B_n)$ to $[0, 1)$. Thus introduce the normalised variables x^*, U_n, V_n, E_n and E'_n in $[0, 1)$ by

$$x_n = g_n(x^*) = \frac{x^* - A_n}{L_n}, \quad u_n = \frac{U_n - A_n}{L_n}, \quad v_n = \frac{V_n - A_n}{L_n},$$

$$e_n = \frac{E_n - A_n}{L_n}, \quad e'_n = \frac{E'_n - A_n}{L_n},$$

where $L_n = B_n - A_n$ is the length of the interval $[A_n, B_n]$. Note that

$$u_n = \min(e_n, e'_n)\,, \; v_n = \max(e_n, e'_n)$$

and the deletion rule is

$$\left\{ \begin{array}{ll} \text{(R)}: & \text{if } f_n(u_n) < f_n(v_n) \text{ delete } [v_n, 1) \\ \text{(L)}: & \text{if } f_n(u_n) \geq f_n(v_n) \text{ delete } [0, u_n) \end{array} \right. \tag{1.5}$$

with $f_n(.)$ the renormalised function on $[0, 1)$ defined by

$$f_n(x) = f[g_n^{-1}(x)]\,,$$

where $g_n(\cdot)$ maps the interval $[A_n, B_n]$ back to the base region $[0, 1]$. The remaining interval is then renormalised to $[0, 1)$. Straightforward calculation then shows that right and left deletions respectively give

$$x_{n+1} = \left\{ \begin{array}{ll} \frac{x_n}{v_n} & \text{for (R)} \\ \frac{x_n - u_n}{1 - u_n} & \text{for (L).} \end{array} \right. \tag{1.6}$$

Moreover, from the definition of E_{n+1}, we obtain

$$e_{n+1} = \left\{ \begin{array}{ll} \frac{u_n}{v_n} & \text{for (R)} \\ \frac{v_n - u_n}{1 - u_n} & \text{for (L).} \end{array} \right. \tag{1.7}$$

Assume that the function $f(.)$ is symmetric with respect to x^*. (Some asymptotic results are also valid when $f(\cdot)$ is only locally symmetric with respect to x^*.) Then the decision concerning left or right deletion at iteration n only depends on the position of x^* with respect to $(E_n + E'_n)/2$. In the renormalised form we thus obtain

$$\left\{ \begin{array}{ll} (R) & \text{if } x_n < \frac{e_n + e'_n}{2} \\ (L) & \text{if } x_n \geq \frac{e_n + e'_n}{2}\,. \end{array} \right. \tag{1.8}$$

6.2 SYMMETRIC ALGORITHMS AND THE GOLDEN SECTION

For general *symmetric algorithms* $E'_n = A_n + B_n - E_n$ which is equivalent to $e'_n = 1 - e_n$. In that case, the length L_n does not depend on the sequence of (R) and (L) deletions and is thus independent of the objective function $f(\cdot)$.

The most famous second–order line–search algorithms are the Fibonacci algorithm (defined only when the number of observations is fixed) and the Golden Section algorithm. They both are symmetric. The Golden Section (GS) algorithm was considered in Introduction; it is defined by

$$[A_1, B_1) = [A_0, B_0)\,, \quad E_1 = A_1 + \varphi L_1\,,$$

$$E'_n = \left\{ \begin{array}{lll} A_n + \varphi L_n & \text{if} & E_n = A_n + (1 - \varphi) L_n\,, \\ A_n + (1 - \varphi) L_n & \text{if} & E_n = A_n + \varphi L_n\,, \end{array} \right.$$

where $L_n = B_n - A_n$ and $\varphi = (\sqrt{5}-1)/2 \simeq 0.61804$. The key property of the algorithm is

$$\frac{E_{n+1} - A_{n+1}}{L_{n+1}} \in \mathcal{U} \quad \forall n \geq 0 \tag{1.9}$$

where $\mathcal{U} = \{1-\varphi, \varphi\}$. This algorithm is known to be asymptotically worst–case optimal in the class of all uniextremal functions, see Kiefer (1957) and Du and Hwang (2000), Theorem 13.2.2. The convergence rate at iteration n satisfies $r_0 = 1$ and $r_n = \varphi$ for all $n \geq 1$, so that $L_n = L_0 \varphi^{n-1}$. For the GS algorithm $v_n = 1 - u_n = \varphi$ and the updating rule (1.6) becomes (1.1).

Unlike GS, the convergence rate of a general symmetric algorithm is not exponential. This follows from the following considerations.

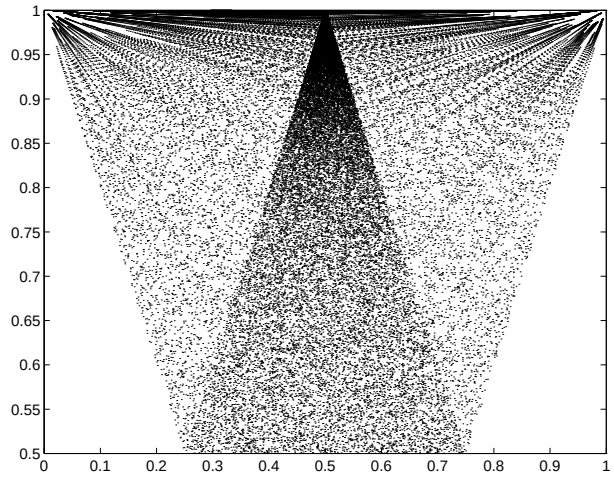

Figure 1.6 Typical sequence of iterates (x_n, r_n) in a symmetric algorithm

For a general symmetric algorithm

$$e_{n+1} \quad = \quad \begin{cases} \frac{u_n}{v_n} & \text{for } (R) \\ \frac{v_n - u_n}{1-u_n} & \text{for } (L) \end{cases} \tag{1.10}$$

and $e'_n = 1 - e_n$ for every n. This last condition implies $u_n = 1 - v_n$, which gives both for the (R) and (L) cases

$$v_{n+1} = \begin{cases} \frac{1}{v_n} - 1 & \text{if } 1/2 \leq v_n < 2/3 \\ 2 - \frac{1}{v_n} & \text{if } 2/3 \leq v_n < 1 \end{cases}$$

with $\{v_n\}$ belonging to the interval $[1/2, 1)$. Note also that for every $n \geq 1$ the rate is $r_n = v_n$. Figure 1.6 presents a plot of a typical sequence of iterates

(x_n, r_n) for a generic symmetric algorithm, with x_n the renormalised location of x^* and r_n the rate.

It is interesting to note that the simple transformation $z_n = 1/v_n - 1$ gives the famous Farey map; that is, $\{z_n\}$ follows the evolution $z_{n+1} = T(z_n)$ with the mapping $T(\cdot)$ defined by

$$T(u) = \begin{cases} u/(1-u) & \text{if } 0 < u < 1/2 \\ (1-u)/u & \text{if } 1/2 \leq u < 1\,. \end{cases}$$

The mapping is presented in Figure 1.7.

The dynamical system $z_{n+1} = T(z_n)$ has a nonexponential divergence of the trajectories. Such a map is called *almost expanding*, which relates to the phenomenon called *chaos with intermittency*. The invariant density for the dynamical system $z_{n+1} = T(z_n)$ is easily computed and equals $p(x) = \text{const}/x$, $0 < x < 1$, where const is any positive number. This density is not integrable, that is

$$\int_0^1 p(x)dx = \infty.$$

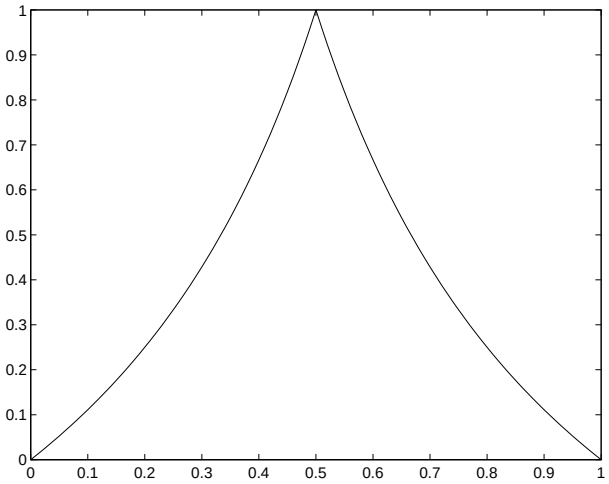

Figure 1.7 The Farey map

Note that any statement on the asymptotic behaviour of $\{z_n\}$ deriving from the Farey map translates into the behaviour of the behaviour of $\{v_n\}$, which is the same for almost all initial values v_1. This implies, for example, that for a generic v_1 (an irrational number that is not a quadratic irrational and, more generally, badly approximable), the invariant density for $\{v_n\}$ is $\phi(v) = 1/[v(1-v)]$. It is also is not integrable. For these generic values of v_1 the asymptotic rate is not exponential, that is $R = 1$ and $\rho = 0$.

However, for some initial points v_1 the rate is exponential. For quadratic irrationals this rate can be easily computed. The best asymptotic convergence rate is for the GS algorithm, that is, when $v_1 = \varphi$, the Golden Section.

6.3 MIDPOINT ALGORITHM

The *midpoint algorithm* was introduced in Wynn and Zhigljavsky (1993). It is defined by $[A_1, B_1) = [A_0, B_0)$, $E_1 = A_1 + e_1 L_1$ with e_1 any irrational number in $[0, 1)$, and

$$E'_n = \frac{A_n + B_n}{2}\,,\ \forall n \geq 1\,.$$

In the renormalised version this is equivalent to simply setting $e'_n = \frac{1}{2}$ for any n.

The dynamical system is now two-dimensional:

$$(x_{n+1}, e_{n+1}) \quad = \quad \begin{cases} (2x_n, 2e_n) & \text{if } x_n < c_n, e_n < \frac{1}{2} \\ (\frac{x_n}{2e_n}, \frac{1}{2e_n}) & \text{if } x_n < c_n, e_n \geq \frac{1}{2} \\ (2x_n - 1, 2e_n - 1) & \text{if } x_n \geq c_n, e_n \geq \frac{1}{2} \\ (\frac{x_n - e_n}{1 - e_n}, \frac{\frac{1}{2} - e_n}{1 - e_n}) & \text{if } x_n \geq c_n, e_n < \frac{1}{2} \end{cases}$$

where $c_n = \frac{e_n + e'_n}{2} = \frac{1}{4} + \frac{e_n}{2}$, $x_1 = x^*$ and e_1 is any irrational number in $[0, 1)$. The first component in this dynamical system is x_n, the renormalised value of x^*. The second component e_{n+1} only depends functionally on x_n through the test for left or right deletion. A typical sequence of iterates $\{x_n, e_n\}$ is shown in Figure 1.8.

One can check that the mapping $T^2(.,.) = T(T(.,.))$ is uniformly expanding, which implies the existence of an invariant measure absolutely continuous with respect to the Lebesgue measure. The Lyapunov exponents for this dynamical system are

$$\begin{aligned} \Lambda_1 &= -\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \log r_n = \varrho \simeq 0.5365\,, \\ \Lambda_2 &= -\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \log(2r_n^2) = 2\Lambda_1 - \log 2 \simeq 0.3799\,. \end{aligned}$$

The numerical values were obtained by simulation as well as by numerical solution of the corresponding Frobenius-Perron equation. In agreement with the result mentioned in Section 4.2, the largest Lyapunov exponent Λ_1 coincides with the log-rate $\varrho = -\log R$. The ergodic convergence rate of the midpoint algorithm is $R \simeq 0.5848$. This is better than the rate $R = \varphi \simeq 0.6180$ of the GS algorithm.

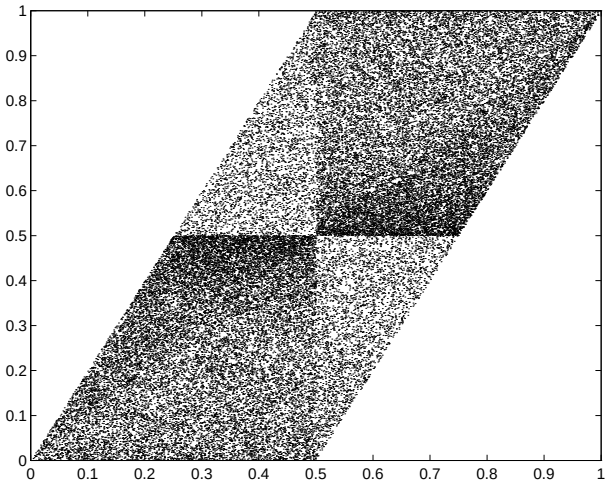

Figure 1.8 Typical sequence of iterates $\{x_n, e_n\}$ for the midpoint algorithm

6.4 WINDOW ALGORITHM

For the so-called *window algorithm*, see Pronzato *et al* (1999) for details,

$$\begin{aligned}[A_1, B_1) &= [A_0 - \epsilon L_0, B_0 + \epsilon L_0)\,, \\ E_1 &= (A_1 + B_1)/2 - wL_1/2\,,\end{aligned}$$

and

$$E'_n = \begin{cases} E_n + wL_n & \text{if } E_n < \frac{1}{2}(A_n + B_n) \\ E_n - wL_n & \text{otherwise,} \end{cases}$$

where $\epsilon \geq 0$ and $w > 0$ are tuning parameters. The ratio $|E'_n - E_n|/L_n$, defining the window width, is thus fixed and equals w.

Renormalisation yields the following two-dimensional dynamical system:

$$(x_{n+1}, e_{n+1}) = \begin{cases} (\frac{x_n}{e_n+w}, \frac{e_n}{e_n+w} - w) & \text{if } x_n < e_n + w/2 \\ (\frac{x_n - e_n}{1-e_n}, \frac{w}{1-e_n}) & \text{if } x_n \geq e_n + w/2\,. \end{cases} \tag{1.11}$$

Figure 1.9 presents a typical plot of the sequence of iterates (x_n, e_n).

The values of ϵ and w could be chosen optimally for each N (the number of iterations) and each criterion of Section 4.3. The ergodic convergence rate R does not depend on ϵ. Good tuning of w for the ergodic criterion R gives $w = 1/8$. This value, however, is far from being optimal for small n for all the criteria W_γ, whatever the choice of ϵ. Also, for a fixed value of w, choosing ϵ large enough ($\epsilon \geq \frac{1-w}{2(1+w)}$), i.e. expanding the initial interval $[A_0\,, B_0)$, guarantees that x_1 belongs to the support of the invariant measure for $\{x_n\}$. This is of crucial importance, since it permits to obtain finite sample characteristics close to their

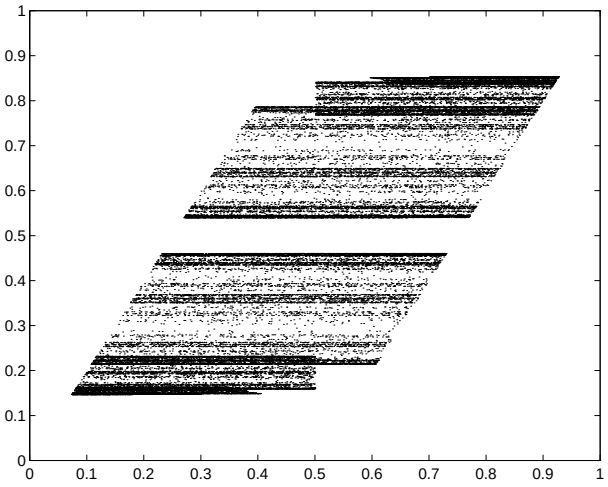

Figure 1.9 Typical sequence of iterates $\{x_n, e_n\}$ for the window algorithm, $w = 1/8$.

asymptotic values. Also, numerical study demonstrates that starting points x_1 outside the support of the invariant measure for x_n give bad convergence rates in the first iterations.

An exhaustive analysis for reasonable values of N ($10 \leq N \leq 30$) has lead to the choice $\epsilon = 0.3772$ and $w = 0.15$, which is close to the best possible for most of the criteria of Section 4.3, both finite-sample and asymptotic ones, see Pronzato *et al* (1999) and (2000) for details.

6.5 GENERALISED GOLDEN SECTION

A natural generalization of the GS algorithm is provided by the use of the so-called *section-invariant numbers*; in this case $\mathcal{U}$ in the right hand side of (1.9) can be chosen as any admissible finite set of points in $(0, 1)$. Section 8.3 in Pronzato *et al* (2000) and Pronzato *et al* (1997a) contain full enumeration of all admissible two– and three–point sets $\mathcal{U}$; there are continuous families of these sets when the number of different points in $\mathcal{U}$ is four or larger. Increasing the number of points in the set $\mathcal{U}$ leads to a bigger flexibility and therefore to a possible increase in efficiency of the corresponding algorithm. In view of the fact that the renormalising transformation is linear, under the condition that the objective function is symmetric around x^*, the dynamical system describing the algorithm can be represented as a piece-wise linear mapping on [0,1].

Consider, for example, the algorithm (we call it GS4 algorithm) defined by $[A_1, B_1) = [A_0 - \epsilon L_0, B0 + \epsilon L_0)$, $\epsilon = (1-a)/2 \simeq 0.43008$, $E_1 = A_1 + bL_1$ and the set $\mathcal{U} = \{a, b, 1-b, 1-a\}$, where $b = 2a^3 - 4a^2 + 3a$, and $a \simeq 0.19412$ is the smallest positive root of the polynomial $2t^4 - 8t^3 + 11t^2 - 7t + 1$. The GS4

algorithm is the best with respect to many ergodic and finite–sample criteria in the class of these algorithms with $|\mathcal{U}| \leq 4$, see Pronzato *et al* (1998) and Pronzato *et al* (2000) for a detailed study.

The updating rule for (x_n, e_n) is

$$(x_{n+1}, e_{n+1}) = \begin{cases} (\frac{x_n}{a'}, c) & \text{if } e_n = a \text{ and } x_n < \frac{a+a'}{2}\,, \\ (\frac{x_n - a}{1-a}, a) & \text{if } e_n = a \text{ and } x_n \geq \frac{a+a'}{2}\,, \\ (\frac{x_n}{c}, d) & \text{if } (e_n = b \text{ or } e_n = c) \text{ and } x_n < \frac{b+c}{2}\,, \\ (\frac{x_n - b}{c}, a) & \text{if } (e_n = b \text{ or } e_n = c) \text{ and } x_n \geq \frac{b+c}{2}\,, \\ (\frac{x_n}{d}, d) & \text{if } e_n = d \text{ and } x_n < 1 - \frac{a+a'}{2}\,, \\ (\frac{x_n - (1-a')}{a'}, b) & \text{if } e_n = d \text{ and } x_n \geq 1 - \frac{a+a'}{2}\,, \end{cases} \tag{1.12}$$

where $c = 1 - b$ and $a' = 2a - a^2$.

Due to the symmetry with respect to $1/2$ of the elements in $\mathcal{U}$, the dynamical system can be simplified and represented as a piece-wise linear mapping of the interval [0,1]; it has the form shown in Figure 1.10. (In this figure we also mark the points defining the partition of the interval [0,1] allowing to represent the transformation as a Markov shift.)

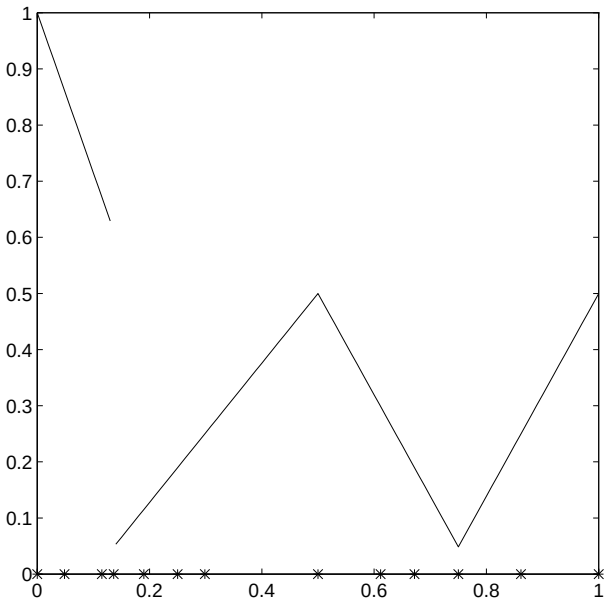

Figure 1.10 Graph of the mapping describing the dynamical system (1.12).

Figure 1.11 presents the values of W_γ for different γ for the GS and GS4 algorithms, and also for the GS4 algorithm with $\epsilon = 0$. The expansion of the search interval at the first iteration of the GS4 algorithm is clearly seen to

improve the performance characteristic W_γ for $\gamma > 1.53$. It also indicates that traditional characteristics like W_1 are not always able to distinguish between algorithms exhibiting much different behaviours.

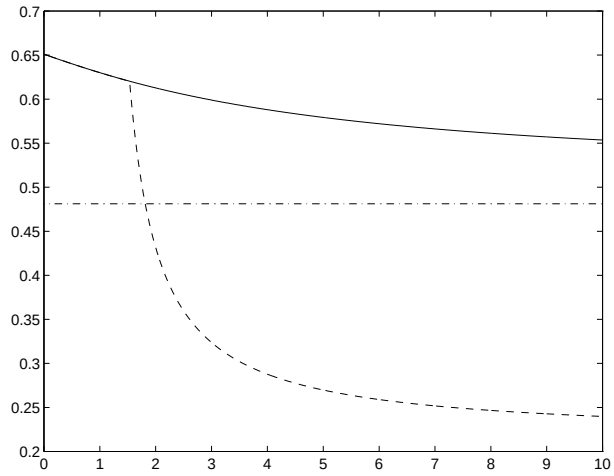

Figure 1.11 Evolution of W_γ as a function of γ for the GS4 (full line), GS4 with $\epsilon = 0$ (dashed line) and GS (dash-dotted line) algorithms.

7. ELLIPSOID ALGORITHMS

Very interesting combinations of geometry and dynamics arise in the ellipsoidal algorithms for linear and convex programming. We describe the basic step in such algorithms.

Assume that (after a renormalisation to be described) we know that x^* lies in a standard unit sphere in R^d

$$S = \{x|\ ||x|| \leq 1\}\,.$$

A 'central cut' algorithm will make a cut through a diametral plane H, which is a plane passing through the centre of S. If H^+ is the half space on one side of this plane and H^- the complement, then the information provided about x^* is

$$x^* \in S \bigcap H^+ \ \text{ or } \ x^* \in S \bigcap H^-$$

Suppose $x^* \in S \bigcap H^+$. Then the minimal volume ellipsoid E is constructed containing $S \bigcap H^+$. The ellipsoid must certainly contain x^*, although it is not the consistency region $S \bigcap H^+$. A renormalisation step is carried out to transform E back to the standard sphere E and the cut step is repeated.

In this primitive form of the algorithm only the direction of the cut needs to be selected at each iteration. The unit direction here is defined by the perpendicular

S_H to H, into H^+, at the origin. A generalised version is to take a so-called 'deep cut', not through the origin but at $H + \alpha S_H$ for some $\alpha \geq 0$. This represents some kind of accelerated algorithm and we gain in the convergence rate while not being penalized because

$$S \cap (H + \alpha S_H)^+ \subset S \cap H^+$$

The choice of direction S_H depends on the nature of the problem. A generic case is that of an unconstrained convex programming where S_H is determined at each iteration as the gradient (or subgradient) direction of the renormalized objective function. The renormalization of the function should be stressed. As the renormalization $h_n : E_n \to S$ is done, one also needs to renormalize f_n to f_{n+1} so that

$$f_n(x) = f_{n+1}(h_n(x))$$

For the deep cut algorithm we find the minimal volume ellipsoid containing

$$S \cap (H + \alpha S_H)^+ .$$

The ellipse is

$$E(S_H, \alpha) = \{z \in R^d \ / \ (z - x)^T (QQ^T)^{-1} (z - x) \leq 1\}$$

with the centre x of the ellipsoid given by

$$x = -\rho \frac{S_H}{\|S_H\|}$$

where $\rho = (1 + d\alpha)/(d + 1)$ and

$$Q = \sqrt{\sigma} \left[I_d - (1 - \sqrt{1 - \tau}) \frac{S_H S_H^T}{\|S_H\|^2} \right]$$

where $\sigma = \frac{d^2(1-\alpha^2)}{d^2-1}$ and $\tau = \frac{2(1+d\alpha)}{(d+1)(\alpha+1)}$, see Figure 1.12

Note that if α is too large then the ellipsoid will not contain the origin. The updating transformation

$$h_n : \ E(S_H, \alpha) \to S$$

is

$$z \mapsto Q^{-1}(z - x) .$$

The measure of the convergence rate is

$$r_n(\alpha) = \frac{\text{vol}(E_{n+1}(S_{H_{n+1}}, \alpha))}{\text{vol}(E_n(S_{H_n}, \alpha))} = \sigma^{d/2} \sqrt{1 - \tau}$$

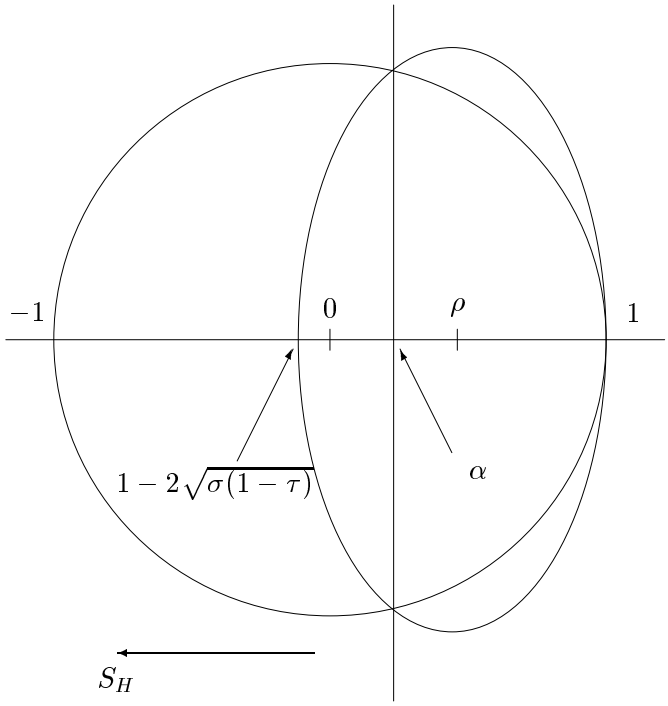

Figure 1.12 A deep cut in the ellipsoid algorithm.

Since $\alpha > 0$ one can see the gain in the rate from taking a deep cut over a central cut when $\alpha = 0$.

Note that

$$(r_n(\alpha))^{1/d} = \sqrt{1-\alpha^2}\left(1 + \frac{1}{d}\log\sqrt{\frac{1-\alpha}{1+\alpha}}\right) + O\left(\frac{1}{d^2}\right)$$

as $d \to \infty$.

In this simple form there are links between the ellipsoidal algorithm and line-search. To see this consider the case when (by good fortune) S_H is a constant direction, say s, taken equal to the first basis vector without any loss of generality. Then the ellipsoid $E(s_1, \alpha)$ will cut the first axis of coordinates at the point $\pm[1 - 2\sqrt{\sigma(1-\tau)}]$. We thus can follow the behaviour of the algorithm as for a line search algorithm on the interval $[-1, 1]$: at each iteration we delete either $(-1 + 2\sqrt{\sigma(1-\tau)}, 1]$ or $[-1, 1 - 2\sqrt{\sigma(1-\tau)})$ and renormalise the remaining interval back to $[-1, 1]$.

Suppose for example that we want this deletion to be precisely as for the Golden Section, then we set $\sqrt{\sigma(1-\tau)} = \phi = \frac{\sqrt{5}-1}{2}$. This is achieved for

$\alpha = \alpha_0 = 1 - (d+1)\phi/d$ and would lead to the asymptotic rate

$$\begin{aligned}(r_n(\alpha))^{1/d} &= \sqrt{1-\alpha_0^2} + O(\frac{1}{d}) = \frac{1}{2}\sqrt{2}\sqrt{3\sqrt{5}-5} + O\left(\frac{1}{d}\right) \\ &\simeq 0.9241763720 + O\left(\frac{1}{d}\right)\end{aligned}$$

as $d \to \infty$. This is an improvement over the asymptotic rate for the central cut algorithm, which is

$$(r_n(0))^{1/d} = \left(\frac{d^2}{d^2-1}\right)^{d/2}\sqrt{\frac{d-1}{d+1}} = 1 - \frac{1}{2d} + O\left(\frac{1}{d^2}\right), \quad d \to \infty.$$

The remarkable aspect of this analysis is that numerical calculations show that this rate is achievable when the direction S_H is changing provided that the algorithm does not break down in the sense that the optimism of taking deep cuts does not loose x^* from the consistent region, altogether. To avoid this, in accordance with Section 5.2, it is necessary to make a sufficient initial scale expansion.

8. STEEPEST DESCENT

Algorithms based on observations of the gradient of the objective function $f(x)$ may also exhibit dynamical system behaviour under suitable renormalisation. We return here to the study of the steepest descent algorithm initiated in Pronzato *et al* (2000, 2001b).

8.1 ALGORITHM AND ITS RENORMALISATION

The basic algorithm takes the form

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)})$$

where

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}\right)^T$$

is the gradient of f and

$$\alpha_k = \arg\min_{\alpha} f(x^{(k)} - \alpha \nabla f(x^{(k)})$$

We can study the steepest descent algorithm in detail for the quadratic function

$$f(x) = \frac{1}{2}(x - x^*)^T A (x - x^*)\,.$$

Numerical simulations show that the same convergence properties hold for convex differentiable functions locally quadratic around their minimum point

x^*. It has been known for many years that the asymptotic convergence rate of the algorithm depends on the ratio $r = \lambda_d/\lambda_1$ (for instance, see Luenberger(1973), p. 152), where λ_1 and λ_d are the minimal and maximal eigenvalues of the matrix A, respectively. Thus for any $x^{(k)} \in I\!R^d$

$$f(x^{(k+1)}) \leq \left(\frac{r-1}{r+1}\right)^2 f(x^{(k)})\,. \tag{1.13}$$

However, it is the worst-case rate, given by (1.13), not the actual rate, which simply depends on λ_1 and λ_d. As we shall see the situation with the actual convergence rate is much more complex.

For the quadratic function

$$g^{(k)} = \nabla f(x^{(k)}) = A(x^{(k)} - x^*)$$

and

$$\alpha_k = \frac{(g^{(k)})^T g^{(k)}}{(g^{(k)})^T A g^{(k)}}$$

and therefore the algorithm can be rewritten as

$$x^{(k+1)} = x^{(k)} - \frac{(g^{(k)})^T g^{(k)}}{(g^{(k)})^T A g^{(k)}} g^{(k)}\,. \tag{1.14}$$

Without loss of generality we can take $A = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$ with $0 < \lambda_1 \leq \ldots \leq \lambda_d$. With this assumption, the iteration of the algorithm (1.14) can be written as

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\sum_{j=1}^d (g_j^{(k)})^2}{\sum_{j=1}^d \lambda_j (g_j^{(k)})^2} g_i^{(k)} \quad \text{for} \;\; i = 1, \ldots, d\,. \tag{1.15}$$

Multiplying both sides of i-th equation of (1.15) by λ_i we obtain

$$g_i^{(k+1)} = g_i^{(k)} - \frac{\sum_{j=1}^d (g_j^{(k)})^2}{\sum_{j=1}^d \lambda_j (g_j^{(k)})^2} \lambda_i g_i^{(k)} \quad \text{for} \;\; i = 1, \ldots, d\,. \tag{1.16}$$

A convenient renormalisation is to set

$$z_i^{(k)} = \frac{y_i^{(k)}}{\sum_{j=1}^d y_j^{(k)}}\,, \quad \text{with } y_i^{(k)} = (g_i^{(k)})^2\,.$$

The equations (1.16) then imply

$$y_i^{(k+1)} = \left(1 - \frac{\lambda_i \sum_{j=1}^d y_j^{(k)}}{\sum_{j=1}^d \lambda_j y_j^{(k)}}\right)^2 y_i^{(k)} \quad \text{for} \;\; i = 1, \ldots, d\,,$$

and

$$z_i^{(k+1)} = z_i^{(k)} \frac{\left(\sum_{j=1}^d \lambda_j z_j^{(k)} - \lambda_i\right)^2}{\sum_{l=1}^d \left(\sum_{j=1}^d \lambda_j z_j^{(k)} - \lambda_l\right)^2 z_l^{(k)}} \quad \text{for } i = 1, \ldots, d\,. \quad (1.17)$$

Note that $z_i^{(k)} \geq 0$ for $i = 1, \ldots, d$ and $\sum_{j=1}^d z_j^{(k)} = 1$. Therefore, we can consider $z_i^{(k)}$ as a weight on the eigenvalue λ_i, so that (1.17) defines a transformation on discrete probability measures supported on $\{\lambda_1, \ldots, \lambda_d\}$.

8.2 CONVERGENCE TO A TWO-DIMENSIONAL PLANE

The asymptotic behaviour of the sequence $\{z^{(k)}\}$ generated by (1.17) is studied in Akaike (1959), Forsythe(1968) and Chapter 7 of Pronzato *et al* (2000). The main result is that, assuming $0 < \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_{d-1} < \lambda_d$, the sequence converges to a two-dimensional plane, spanned by the eigenvectors u_1, u_d associated with λ_1 and λ_d. More precisely, the following result holds.

Theorem 1. *Let the objective function f be $f(x) = \frac{1}{2}(x - x^*)^T A(x - x^*)$ with the eigenvalues of A satisfying $0 < \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_{d-1} < \lambda_d$. Let $V = \text{span}(u_1, u_d)$ be the two-dimensional plane generated by the (distinct orthonormal) eigenvectors u_1, u_d corresponding to λ_1 and λ_d, respectively. Then for any starting vector $x^{(1)}$ for which*

$$u_1^T(x^{(1)} - x^*) \neq 0 \;\; and \;\; u_d^T(x^{(1)} - x^*) \neq 0\,, \quad (1.18)$$

the algorithm attracts to the plane V in the following sense:

$$w^T z^{(k)} \to 0\,, \quad k \to \infty\,,$$

for any non-zero vector $w \in V^\perp$. Moreover, the sequence $\{z^{(k)}\}$ of renormalised variables defined by (8.1) *converges to a two-point cycle:*

$$z^{(2k)} \to p\,u_1 + (1-p)\,u_d\,,\; z^{(2k+1)} \to (1-p)\,u_1 + p\,u_d \;\; \textit{when } k \to \infty\,, \quad (1.19)$$

where p is some number in $(0, 1)$ depending on $z^{(1)}$.

Note that the updating formula (1.17) is exactly the same for the weights $z_i^{(k)}$ and $z_j^{(k)}$ corresponding to equal $\lambda_i = \lambda_j$, and therefore these weights can be summed. Hence the assumption that all λ_i are different can easily be extended to the case of ties among $\lambda_2, \ldots, \lambda_{d-1}$. In the case when at least one of the eigenvalues λ_1 or λ_d is not simple, we still have convergence to a cycle (1.19), where u_1 and u_d are certain eigenvectors associated with λ_1 and λ_d. These eigenvectors are determined uniquely only in the case when the eigenvalues

λ_1 and λ_d are simple and (1.18) holds. Also, the result obviously generalizes to the case when (1.18) is not satisfied (at least one of the scalar products is equal to zero). The algorithm then attracts to a two-dimensional plane V' and $\{z^{(k)}\}$ converges to a two-point cycle. V' is defined by the eigenvectors u_i, u_j associated with the smallest and largest eigenvalues such that $u_i^T z^{(1)} \neq 0$, $u_j^T z^{(1)} \neq 0$. For the sake of simplicity of notations, we shall assume that (1.18) is satisfied, and therefore $u_i = u_1$, $u_j = u_d$.

8.3 ASYMPTOTIC RATE

The property stated in Theorem 1 has important consequences on the asymptotic rate of convergence of the steepest descent algorithm. Namely, the actual rate is determined by the limiting value of p in (1.19). Under the assumption that $x^{(1)} - x^*$ does not belong to the two-dimensional plane V, the limiting value of p is not arbitrary; as proved in Pronzato *et al* (2000), it belongs to the interval

$$I = \left[\frac{1}{2} - s(\lambda_{i^*}),\ \frac{1}{2} + s(\lambda_{i^*})\right] ,$$

where

$$s(\lambda) = \frac{\sqrt{(\lambda_d - \lambda)^2 + (\lambda_1 - \lambda)^2}}{2(\lambda_d - \lambda_1)} , \tag{1.20}$$

and i^* is such that $|\lambda_{i^*} - \frac{\lambda_1+\lambda_d}{2}|$ is minimum over all λ_i's, $i = 2, \ldots, d-1$. The smallest possible interval I is

$$I_0 = \left[\frac{1}{2} - \frac{\sqrt{2}}{4},\ \frac{1}{2} + \frac{\sqrt{2}}{4}\right] \simeq [0.14645, 0.85355] .$$

The minimal length interval $I = I_0$ for the limiting value p does appear when $\lambda_{i^*} = \frac{\lambda_1+\lambda_d}{2}$ for some $i^* \in \{2, \ldots, d-1\}$.

Assuming that the initial point $x^{(1)}$ is random and uniformly distributed on a sphere with the centre x^*, the density of the limiting values of p is well-spread over the interval I, see Figure 1.13.

The relation between the limiting value p in (1.19) and the asymptotic convergence rate R defined in (1.4) is simple. Indeed, Theorem 1 implies that for any fixed x^* and almost all $x^{(1)}$ the asymptotic rate depends only on the value of p and is given by $R(p) = [f(x^{(k+2)})/f(x^{(k)})]^{1/2}$, where $x^{(k)}$ is associated with $p\, e_1 + (1-p)\, e_d$ or $(1-p)\, e_1 + p\, e_d$. This gives

$$R(p) = \frac{p(1-p)(\rho-1)^2}{[p + \rho(1-p)][(1-p) + \rho p]} ,$$

with $r = \lambda_d/\lambda_1$, the condition number of the matrix A. The function $R(p)$ is symmetric with respect to $1/2$ and monotonously increasing from 0 to $1/2$.

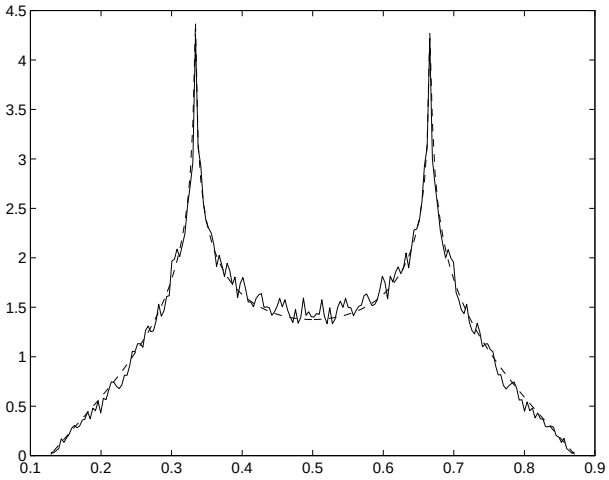

Figure 1.13 The density of the limiting values of p in (1.19); $d = 3$, $\lambda_1 = 1, \lambda_2 4, \lambda_3 = 10$.

The worst asymptotic rate is thus obtained at $p = 1/2$:

$$R_{\max} = \left(\frac{r-1}{r+1}\right)^2 ,$$

see (1.13). In terms of $x^{(1)}$, the worst rate is achieved only when $x_1^{(k)} = \pm r x_d^{(k)}$ and $x_2^{(k)} = \cdots = x_{d-1}^{(k)} = 0$.

Although the limiting behaviour of the algorithm is simple, its behaviour en route to the attractor is fairly complicated and presents a fractal structure. It is very difficult, if not impossible, to relate the initial point $x^{(1)}$ and the limiting value p (and thus the asymptotic rate).

To give a flavour of the behaviour of the algorithm we can take $d = 3$, $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 4$ and consider the region of attraction in the two-dimensional simplex occupied by z_1, z_2, z_3 or a small region around a selected (allowable) point. Figure 1.14 shows the projection onto the plane (z_1, z_3) of a typical region of attraction for $(x^{(k)} - x^*)/||x^{(k)} - x^*||$.

In Figure 1.15 we used the baricentric coordinates for z_1, z_2, z_3 and have coloured points in black or white according to whether their limit points lie in a selected half of the interval I. Since the rate is a function of the limit 'cycle' one sees immediately that

(i) The rate depends on the start;

(ii) As we approach the vertices of the simplex given by $z_i = 1$ for some $i = 2, \ldots, d-1$, the separation between black and white is arbitrary fine. This shows that starts closer and closer to these points lead to instability in the rate.

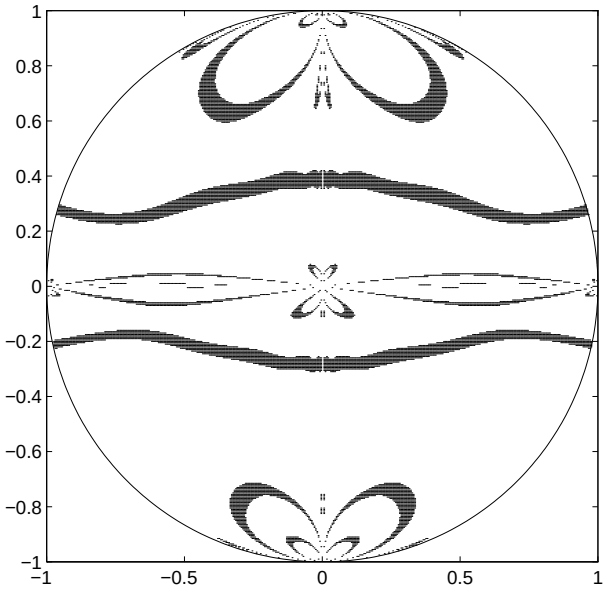

Figure 1.14 Projection of the region of attraction to a small neighbourhood of a limiting point.

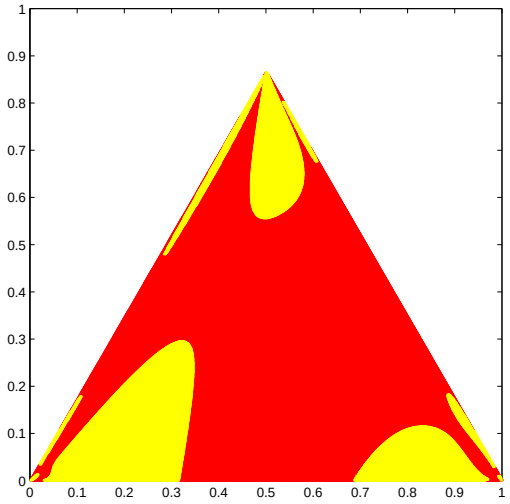

Figure 1.15 The region of attraction for a half of the interval I.

It also holds that until a neighbourhood of the limit cycle is reached there, the $\{z^{(k)}\}$ sequence may at any stage stay close to a corner or switch between different corners. At each vertex the interchanging of the trajectories has a self-similar nature. This kind of behaviour is not unusual in the family of the

rational maps, which the transformation (1.17) belongs to, but it still came as a surprise to find such complexity in a very classical algorithm.

In Figure 1.16, the grey level of starting points on the unit sphere depends on the limiting value of p in (1.19). Again, this figure illustrates the difficulty of predicting the limiting attractor for the sequence $\{z^{(k)}\}$, and thus the asymptotic rate of convergence of the algorithm, as a function of the starting point.

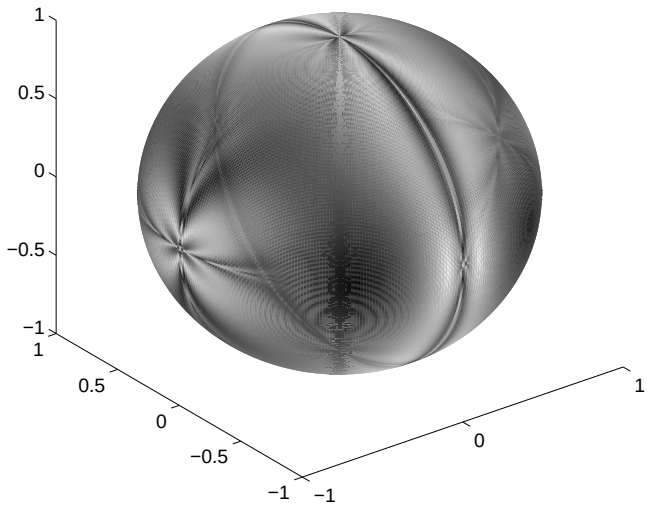

Figure 1.16 Starting points on the unit sphere colored as a function of the limiting rate

8.4 STEEPEST DESCENT WITH RELAXATION

The introduction of a relaxation coefficient γ, with $0 < \gamma < 1$, in the steepest-descent algorithm totally changes its behaviour. The algorithm (1.14) then becomes

$$x^{(k+1)} = x^{(k)} - \gamma \frac{(g^{(k)})^T g^{(k)}}{(g^{(k)})^T A g^{(k)}} g^{(k)} .$$

For fixed A, depending on the value of γ, the renormalized process either attracts to periodic orbits (the same for almost all starting points) or exhibits a chaotic behaviour. Figure 1.17 presents the classical period-doubling phenomenon in the case $d = 2$ when $\lambda_1 = 1$ and $\lambda_2 = 4$. Figure 1.18 gives the asymptotic rate (1.4) as a function of γ in the same situation.

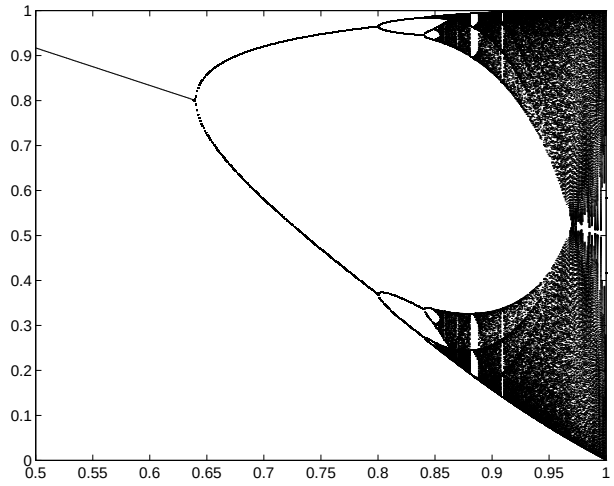

Figure 1.17 Attractors for z_1 as a function of γ ($d = 2, \lambda_1 = 1, \lambda_2 = 4$)

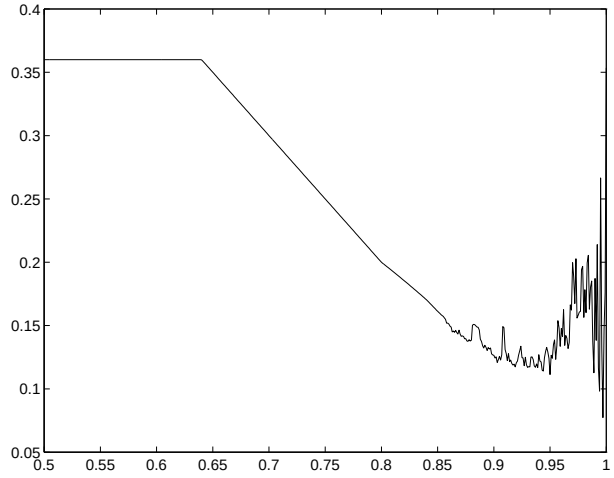

Figure 1.18 Asymptotic rate (1.4) as a function of γ ($d = 2, \lambda_1 = 1, \lambda_2 = 4$)

Numerical results show that for γ large enough (but $\gamma < 1$) the asymptotic rate is significantly better than $R_{\max} = (1 - \frac{2}{r+1})^2$, the worst-case rate for the steepest descent algorithm ($\gamma = 1$). A detailed analysis of the 2-dimensional case gives the following results.

(i) If $0 < \gamma \leq \frac{2}{r+1}$, where $r = \lambda_2/\lambda_1$, the process attracts to the fixed point $p = 1$ and $R = R(\gamma) = 1 - \frac{2r\gamma}{r+1}$.

(ii) If $\frac{2}{r+1} < \gamma \leq \frac{4r}{(r+1)^2}$, the process attracts to the fixed point $p = \frac{2r-\gamma(r+1)}{2(r-1)}$, and $R(\gamma) = R_{\max}$.

(iii) If $\frac{4r}{(r+1)^2} < \gamma \leq \frac{2(\sqrt{2}+1)r}{(r+1)^2}$, the process attracts to the 2-point cycle (p_1, p_2), with

$$p_{1,2} = \frac{2r - \gamma(r+1) \pm \sqrt{\gamma(\gamma r^2 - 4r + 2\gamma r + \gamma)}}{2(r-1)}\,,$$

and $R(\gamma) = 1 - \gamma$.

(iv) For larger values of γ one observes a classical period-doubling phenomenon, see Figure 1.17.

(v) If $r > 3 + 2\sqrt{2} \approx 5.828427$, the process attracts again to a 2-point cycle for values of γ larger than $\gamma_r = \frac{8r}{(r+1)^2}$, see Figure 1.17. For the limiting case $\gamma = \gamma_r$, the cycle is given by (p'_1, p'_2), with

$$p'_{1,2} = \frac{r\left(r^2 - 2r + 5 \pm 2\sqrt{(r^2 - 2r + 5)(5r^2 - 2r + 1)}\right)}{(r-1)(r+1)^3}\,,$$

and the associated asymptotic rate is $R(\gamma_r) = (r^2 - 6r + 1)/(r^2 - 1)$.

In higher dimensions, the process typically no longer attracts to the 2-dimensional plane spanned by (u_1, u_d) and exhibits a chaotic behaviour in the d-dimensional space. For large values of the relaxation coefficient $\gamma < 1$ the asymptotic convergence rate is typically significantly better than $R_{\max} = (1 - \frac{2}{r+1})^2$, the worst-case rate for the steepest descent algorithm.

References

Akaike H. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Statist. Math. Tokyo*, 11:1–16, 1959.

Boender C.G.E. and Romeijn H.E. Stochastic methods. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization I*, pages 829–869. Kluwer, Dordrecht, 1995.

Du D.Z. and Hwang F.K. *Combinatorial Group Testing and its Applications*. World Scientific, Singapore, 2000.

Forsythe G. On the asymptotic directions of the s-dimensional optimum gradient method. *Numerische Mathematik*, 11:57–76, 1968.

Hansen P. and Jaumard B. Lipschitz optimization. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization I*, pages 407–493. Kluwer, Dordrecht, 1995.

Kiefer J. Optimum sequential search and approximation methods under minimum regularity assumptions. *J. Soc. Indust. Appl. Math.*, 5:105–136, 1957.

Luenberger D. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachusetts, 1973.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. Stochastic analysis of convergence via dynamic representation for a class of line-search algorithms. *Combinatorics, Probability and Computing*, 6:205–229, 1997.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. Using Renyi entropies in search problems. *Lectures in Applied Mathematics*, 33:253–268, 1997.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. A generalised Golden-Section algorithm for line–search. *IMA Journal on Math. Control and Information*, 15:185–214, 1998.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. Finite sample behaviour of an ergodically fast line-search algorithm. *Computational Optimisation and Applications*, 14:75–86, 1999.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. *Dynamical Search*. Chapman & Hall/CRC, Boca Raton, 2000.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. Analysis of performance of symmetric second–order line search algorithms through continued fractions. *IMA Journal on Math. Control and Information*, 18:281–296, 2001.

Pronzato L., Wynn H.P., and Zhigljavsky A.A. Renormalised steepest descent in Hilbert space converges to a two-point attrator. *Acta Applicandae Mathematicae*, 40, 2001. (to appear).

Wynn H.P. and Zhigljavsky A.A. Chaotic behaviour of search algorithms. *Acta Applicandae Mathematicae*, 32:123–156, 1993.

Zhigljavsky A.A. *Theory of Global Random Search*. Kluwer, Dordrecht, 1991.

Zhigljavsky A.A. and Chekmasov M.V. Comparison of independent, stratified and random covering sample schemes in optimization problems. *Math. Comput. Modelling*, 23:97–110, 1996.