International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

Stochastic Quasigradient Methods and their Implications

Ermoliev, Y.M. and Gaivoronski, A.A.

IIASA Working Paper

July 1984

STOCHASTIC QUASIGRADIENT METHODS AND THEIR IMPLEMENTATION

Yuri Ermoliev
Alexei Gaivoronski

July 1984
WP-84-55

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
2361 Laxenburg, Austria

PREFACE

This paper discusses various stochastic quasigradient methods and considers their computer implementation. It is based on experience gained both at the V. Glushkov Institute of Cybernetics in Kiev and at IIASA.

The paper falls naturally into three parts. The first is concerned with problem definition and various ways of choosing the step size and step direction. A detailed description of an interactive stochastic optimization package (STO) available at IIASA forms the second part. Finally, the use of this package is demonstrated by application to three test problems that have arisen during the course of IIASA work: they include a facility location problem and a water management problem.

This work was carried out within the Adaptation and Optimization Project of the System and Decision Sciences Program.

ANDRZEJ WIERZBICKI
Chairman
System and Decision Sciences

ABSTRACT

A number of stochastic quasigradient methods are discussed from the point of view of implementation. The discussion revolves around the interactive package of stochastic optimization routines (STO) recently developed by the Adaptation and Optimization group at IIASA. (This package is based on the stochastic and nondifferentiable optimization package (NDO) developed at the V. Glushkov Institute of Cybernetics in Kiev.) The IIASA implementation is described and its use illustrated by application to three problems which have arisen in various IIASA projects.

STOCHASTIC QUASIGRADIENT METHODS AND THEIR IMPLEMENTATION

Yuri Ermoliev and Alexei Gaivoronski

1. INTRODUCTION

This paper discusses various stochastic quasigradient methods (see [1,2]) and considers their computer implementation. It is based on experience gained both at the V. Glushkov Institute of Cybernetics in Kiev and at IIASA.

We are concerned here mainly with questions of implementation, such as the best way to choose step directions and step sizes, and therefore little attention will be paid to theoretical aspects such as convergence theorems and their proofs. Readers interested in the theoretical side are referred to [1,2].

The paper is divided into five sections. After introducing the main problem in Section 1, we discuss the various ways of choosing the step size and step direction in Sections 2 and 3. A detailed description of an interactive stochastic optimization package (STO) currently available at IIASA is given in Section 4. This package represents one possible implementation of the methods described in the previous sections. Finally, Section 5 deals with the solution of some test problems using this package. These problems were brought to our attention by other IIASA projects and collaborating institutions and include a facility location problem, a water resources management problem, and the problem of choosing the parameters in a closed loop control law for a stochastic dynamical system with delay.

We are mainly concerned with the problem

$$\min \{F(x) : x \in X\} \ , \quad F(x) = \mathrm{E}_{\omega} f(x,\omega) \ , \tag{1}$$

where x represents the variables to be chosen optimally, X is a set of constraints, and ω is a random variable belonging to some probabilistic space (Ω,B,P). Here B is a Borel field and P is a probabilistic measure.

There are currently two main approaches to this problem. In the first, we take the mathematical expectation in (1), which leads to multidimensional

integration and involves the use of various approximation schemes [3–6]. This reduces problem (1) to a special kind of nonlinear programming problem which allows the application of deterministic optimization techniques. In this paper we concentrate on the second approach, in which we consider a very limited number of observations of random function $f(x,\omega)$ at each iteration in order to determine the direction of the next step. The resulting errors are smoothed out until the optimization process terminates (which happens when the step size becomes sufficiently small). This approach was pioneered in [7,8].

We assume that set X is defined in such a way that the projection operation $x \to \pi_X(x)$ is comparatively inexpensive from a computational point of view, where $\pi_X(x) = \arg\min_{z \in X} \|x - z\|$. For instance, if X is defined by linear constraints, then projection is reduced to a quadratic programming problem which, although challenging if large scale, can nevertheless be solved in a finite number of iterations. In this case it is possible to implement a stochastic quasigradient algorithm of the following type:

$$x^{s+1} = \pi_X(x^s - \rho_s v^s) \quad . \tag{2}$$

Here x^s is the current approximation of the optimal solution, ρ_s is the step size, and v^s is a random step direction. This step direction may, for instance, be a statistical estimate of the gradient (or subgradient in the nondifferentiable case) of function $F(x)$: then $v^s \equiv \xi^s$ such that

$$\mathrm{E}(\xi^s \mid x^1, x^2, \ldots, x^s) = F_x(x^s) + \alpha^s \quad , \tag{3}$$

where α^s decreases as the number of iterations increases, and the vector v^s is called a *stochastic quasigradient* of function $F(x)$. Usually $\rho_s \to 0$ as $s \to \infty$ and therefore $\|x^{s+1} - x^s\| \to 0$ from (2). This suggests that we should take x^s as the initial point for the solution of the projection problem at iteration number $s+1$, thus reducing considerably the computational effort needed to solve the quadratic programming problem at each step $s = 1,2,\ldots$. Algorithm (2)–(3) can also cope with problems with more general constraints formulated in terms of mathematical expectations

$$\mathrm{E}_\omega f^i(x,\omega) \geq 0 \quad , \quad i = \overline{1,m}$$

by making use of penalty functions or the Lagrangian (for details see [1,2]).

The principal peculiarity of such methods is their nonmonotonicity, which may sometimes show itself in highly oscillatory behavior. In this case it is difficult to judge whether the algorithm has already approached a neighborhood of the optimal point or not, since exact values of the objective function are not available. The best way of dealing with such difficulties seems to be to use an interactive procedure to choose the step sizes and step directions, especially if it does not take much time to make one observation. More reasons for adopting an interactive approach and details of the implementation are given in the following sections.

Another characteristic of the algorithms described here is their pattern of convergence. Because of the probabilistic nature of the problem, their asymptotic rate of convergence is extremely slow and may be represented by

$$\|x^* - x^s\| \sim \frac{C}{\sqrt{k}} \tag{4}$$

Here x^* is the optimal point to which sequence x^s converges and k is the number of observations of random parameters ω, which in many cases is proportional to the number of iterations. In deterministic optimization a superlinear asymptotic convergence rate is generally expected; a rate such as (4) would be considered as nonconvergence. But no algorithm can do asymptotically any better than this for stochastic problem (1) in the presence of nondegenerate random disturbances, and therefore the aim is to reach some ***neighborhood*** of the solution rather than to find the precise value of the solution itself. Algorithm (2)–(3) is quite good enough for this purpose.

2. CHOICE OF STEP DIRECTION

In this section we shall discuss different ways of choosing the step direction in algorithm (2) and some closely related algorithms. We shall first discuss methods which are based on observations made at the current point x^s or in its immediate vicinity. More general ways are then presented which take into account observations made at previous points.

2.1. Gradients of random function $f(x,\omega)$

The simplest case arises when it is possible to obtain gradients (or subgradients in the nondifferentiable case) of function $f(x,\omega)$ at fixed values of x and ω. In this case we can simply take

$$\xi^s = f_x(x^s, \omega^s) \quad , \tag{5}$$

where ω^s is an observation of random parameter ω made at step number s. If both the observation of random parameters and the evaluation of gradients are computationally inexpensive then it is possible to take the average of some specified number N of gradient observations:

$$\xi^s = \frac{1}{N}\sum_{i=1}^{N} f_x(x^s, \omega^{i,s}) \tag{6}$$

These observations can be selected in two ways. The first is to choose the $\omega^{i,s}$ according to their probability distribution. If we do not know the form of the distribution function (as, for example, in Monte-Carlo simulation models) this is the only option. However, in this case the influence of low-probability high-cost events may not be properly taken into account. In addition, the asymptotic error of the gradient estimate ξ^s is approximately proportional to $1/\sqrt{N}$. The second approach may be used when we know the distribution of the random parameters ω. In this case many other estimates can be derived; the use of pseudo-random numbers* in particular may lead to an asymptotic error approximately proportional to $\log(N)/N$, which is considerably less than in the purely random case. However, more theoretical research and more computational experience are necessary before we can assess the true value of this approach. The main question here is whether the increase in the speed of convergence is sufficient to compensate for the additional computational effort required for more exact estimations of the $F_x(x^s)$.

Unfortunately, our theoretical knowledge concerning the asymptotic behavior of processes of type (2) tells us little about the optimal number of samples, even for relatively well-studied cases. For instance, what would be the optimal number N of observations for the case in which function $F(x)$ is differentiable and there are no constraints? In this case we can establish both asymptotic normality and the value of the asymptotic variance. If, additionally, $\rho_s \sim C/s$ then the total number of observations required to obtain a given asymptotic variance is the same for all $N \ll s$. If $s\rho_s \to \infty$ then the wait-and-see approach is asymptotically superior as long as $N \ll s$.

*A concept which arose from the use of quasi-Monte-Carlo techniques in multidimensional integration [9].

However, there is strong evidence that in constrained and/or nondifferentiable cases the value of N should be chosen adaptively. A very simple example provides some insight into the problem. Suppose that $x \in R^1$, $X = [a,\infty)$, $F(x) = x$, $f_x(x^s,\omega^s) = 1 + \omega^s$, where the ω^s, $s = 1,2,\ldots$, are independent random variables with zero mean. The obvious solution of this problem is $x = a$. Suppose for simplicity that $\rho_s \equiv \rho$. This will not alter our argument greatly because ρ_s usually changes very slowly for large s. In this case method (2),(5) will be of the form:

$$x^{s+1} = x^s - \rho(1 + \omega^s) + \tau_s \quad ,$$

$$\tau_s = \max\{0\,,\ a - x^s + \rho(1 + \omega^s)\}$$

Method (2),(6) requires us to choose a step size N times greater than ρ; otherwise its performance would be inferior to that of method (2),(5) (unless the initial point is in the immediate vicinity of the minimum). Method (2),(6) then becomes

$$z^{s+1} = z^s - N\rho(1 + \frac{1}{N}\sum_{i=1}^{N} \omega^{i,s}) + \vartheta_s \quad ,$$

$$\vartheta_s = \max\{0\,,\ a - z^s + N\rho(1 + \frac{1}{N}\sum_{i=1}^{N} \omega^{i,s})\}$$

In order to compare the two methods we shall let s in the last equation denote the number of observations rather than the number of iterations and renumber the observations $\omega^{i,s}$. The process

$$y^k = y^0 - \rho\sum_{i=0}^{k-1}(1 + \omega^i) + \sum_{i=0}^{k-1}\chi_i \quad , \tag{7}$$

$$\chi_i = \begin{cases} 0 & \text{if } i \neq lN \text{ for } l = 1,2,\ldots \text{ or } a < y^i - \rho(1 + \omega^i) \\ a - y^i + \rho(1 + \omega^i) & \text{otherwise} \end{cases}$$

has the property that $y^{lN} = z^s$ and therefore it is sufficient to compare y^k with x^k for $k = lN$, where

$$x^k = x^0 - \rho\sum_{i=0}^{k-1}(1 + \omega^i) + \sum_{i=0}^{k-1}\tau_i \tag{8}$$

Suppose that $x^0 = y^0 \neq a$. Then if $t_x^1 = \min\{k : x^k = a\}$ represents the time at which process x^k first encounters the optimal point and $t_z^1 = \min\{l : y^{lN} = a\}$

represents the time of the corresponding encounter of process z^k with the optimal point, it is clear that $t_x^1 \le t_z^1$ because from (7) and (8) we have that $y^k = x^k$ for $k \le t_x^1$. This means that algorithm (2),(5) will get from some remote initial point to the vicinity of the optimal point faster then algorithm (2),(6) with $N > 1$. Now let us take $x^0 = y^0 = \alpha$. Then (7) and (8) imply that $\chi_k = 0$ for $k < N$ while τ_k may differ from zero. Therefore in this case $x^N \ge y^N = z^1$ and the performance of algorithm (2),(6) with $N > 1$ becomes superior to that of algorithm (2),(5) after reaching the vicinity of the optimal point. This simple example demonstrates several important properties of constrained stochastic optimization problems, although more work is necessary before we can make any firm theoretical recommendations concerning the choice of the number of samples N. Above all, an appropriate definition of the rate of convergence is needed: recent results by Kushner [10] may be useful in this regard.

A rather general adaptive way of changing the number N would be to begin with a small value of N for the first few iterations ($N = 1$, for example), and increase N if additional tests show that the current point is in the vicinity of the optimum. The following averaging procedure has been shown to be useful in tests of this type:

$$v^{s+1} = (1 - \alpha_s)v^s + \alpha_s \xi^s , \quad 0 \le \alpha_s \le 1 \quad , \tag{9}$$

where ξ^s is defined by (5) or (6). It can be shown (see [1,2]) that $\|v^s - F_x(x^s)\| \to 0$ under rather general conditions, which include $\rho_s / \alpha_s \to 0$. The decision as to whether to change N may then be based on the value of $r_s = \|x^s - \pi_X(x^s - v^s)\|$. One possibility is to estimate ξ^s and its empirical variance at the same time:

$$\sigma_N^s = \frac{1}{N} \sum_{i=1}^{N} [f_x(x^s, \omega^{i,s}) - \xi^s]^2$$

and choose N such that $\sigma_N^s \le \beta r_s$, where the value of β is set before beginning the iterations. In practice it is sufficient to consider a constant $\alpha_s \equiv \alpha \sim 0.01$–$0.05$, where the greater the randomness, the smaller the value of α. Our empirical recommendation for the initial value of N is $\sigma_N^0 \sim 0.1 \max_{x_1, x_2 \in X} \|x_1 - x_2\|$.

This method can be used to increase the number of samples per iteration automatically. Another possibility is to alter the value of N interactively; this is one of the options implemented in the interactive package STO, which has recently been developed at IIASA. Numerical experiments conducted with this package show that in problems where $f_x(x,\omega)$ has a high variance, choosing a value of N greater than one can bring about considerable improvements in performance.

The method described above uses increasingly precise estimates of the gradient, and therefore shares some of the features of the approximation techniques developed in [3–6] for solving stochastic programming problems. All of the remarks made here concerning sampling are also valid for the other methods of choosing ξ^s described below.

However, it is not always possible to use observations of the gradient $f_x(x,\omega)$ of the random function to compute a stochastic quasigradient. In many cases the analytic expression of $f_x(x,\omega)$ is not known, and even if it is, it may be difficult to create a subroutine to evaluate it, especially for large-scale problems. In this case it is necessary to use a method which relies only on observations of $f(x,\omega)$.

2.2. Finite-difference approximations

If function $F(x)$ is differentiable, one possibility is to use forward finite differences:

$$\xi^s = \sum_{i=1}^{n} \frac{f(x^s + \delta_s e_i, \omega_{i,1}^s) - f(x^s, \omega_{i,2}^s)}{\delta_s} e_i \quad , \tag{10}$$

or central finite differences:

$$\xi^s = \sum_{i=1}^{n} \frac{f(x^s + \delta_s e_i, \omega_{i,1}^s) - f(x^s - \delta_s e_i, \omega_{i,2}^s)}{2\delta_s} e_i \quad , \tag{11}$$

where the e_i are unit basis vectors from R^n. The most important question here is the value of δ_s. In order to ensure convergence with probability one it is sufficient to take any sequence δ_s such that $\sum_{i=1}^{\infty} \rho_i^2 / \delta_i^2 < \infty$. If it is possible to take $\omega_{i,2}^s = \omega_{i,1}^s$ then any $\delta_s \to 0$ will do. However, the method may reach the vicinity of the optimal point much faster if δ_s is chosen adaptively. On the first few iterations δ_s should be large, decreasing as the current point approaches

the optimal point. The main reason for this is that taking a large step δ_s when the current point is far from the solution may smooth out the randomness to some extent, and may also overcome some of the problems (such as curved valleys) caused by the erratic behavior of the deterministic function $F(x)$. One possible way of implementing such a strategy in an unconstrained case is given below.

(i) Take a large initial value of δ_s, such as $\delta_s \sim 0.1 \max_{x_1, x_2 \in X} \|x_1 - x_2\|$.

(ii) Proceed with iterations (2), where ξ^s is determined using (10) or (11). While doing this, compute an estimate of the gradient v^s from (9).

(iii) Take

$$\delta_{s+1} = \begin{cases} \delta_s & \text{if } v^s \geq \beta_1 \delta_s \\ \beta_2 \delta_s & \text{otherwise} \end{cases},$$

where the values of β_1 and β_2 should be chosen before beginning the iterative process.

It can be shown that this process converges when $\omega_{i,1}^s \equiv \omega_{i,2}^s$, although it will also produce a good approximation to the solution even if this requirement is not met. Estimate (9) is not the only possibility – in fact, any of the estimates of algorithm performance given in Section 3 would do.

Another strategy is to relate changes in the finite-difference approximation step to changes in the step size. This is especially advisable if the step size is also chosen adaptively (see Section 3). In the simplest case one may fix $\beta_1 > 0$ before starting and choose $\delta_s = \beta_1 \rho_s$, which, although contrary to theoretical recommendations, will nevertheless bring the current point reasonably close to the optimal point. To obtain a more precise solution it is necessary to reduce β_1 during the course of the iterations. This may be done either automatically or interactively; both of these options are currently available in the stochastic optimization package STO.

Finite-difference algorithms (10) and (11) have one major disadvantage, and this is that the stochastic quasigradient variance increases as δ_s decreases. This means that finite-difference algorithms converge more slowly than algorithms which use gradients (5). There are two ways of overcoming this problem. Firstly, if it is possible to make observations of function $f(x,\omega)$ for

various values of x and fixed ω, it is a good idea to take the same values of ω for the differences (i.e., $\omega_{i,1}^s = \omega_{i,2}^s$) when δ_s is small because this reduces the variance of the estimates quite considerably. Another way of avoiding this increase in the variance is to increase the number of samples used to obtain ξ^s when approaching the optimal point, i.e., to use finite-difference analogues of (6). If there exists a $\gamma > 0$ such that $N_s \delta_s^2 > \gamma$, where N_s is the number of samples taken at step number s, then the variance of ξ^s remains bounded.

It is sometimes useful to normalize the ξ^s, especially when the variance is large.

Another disadvantage of the finite-difference approach is that it requires $n + 1$ evaluations of the objective function for forward differences and $2n$ for central differences, where n is the dimension of vector x. This may not be acceptable in large-scale problems and in cases where function evaluation is computationally expensive. In this situation a stochastic quasigradient can be computed using some analogue of random search techniques.

2.3. Analogues of random search methods

When it is not feasible to compute $n + 1$ values of the objective function at each iteration, the following approach (which has some things in common with the random search techniques developed for deterministic optimization problems) may be used:

$$\xi^s = \sum_{i=1}^{M_s} \frac{f(x^s + \delta_s h_i, \omega_{i,1}^s) - f(x^s, \omega_{i,2}^s)}{\delta_s} h_i \qquad (12)$$

Here the h_i are vectors distributed uniformly on the unit sphere, M_s is the number of random points and δ_s is the step taken in the random search. The choice of M_s is determined by the computational facilities available, although it is advisable to increase M_s as δ_s decreases. This method of choosing ξ^s has much in common with finite-difference schemes, and the statements made above about the choice of δ_s in the finite-difference case also hold for (12).

2.4. Smoothing the objective function

Methods of choosing ξ^s which rely on finite-difference or random search techniques are only appropriate when the objective function $F(x)$ is differentiable. The use of similar procedures in the nondifferentiable case

would require some smoothing of the objective function. Suppose that the function $F(x)$ is not differentiable but satisfies the Lipschitz condition, and consider the function

$$F(x,r) = \int F(x + y)\mathrm{d}H(y,r) \quad , \tag{13}$$

where $H(y,r)$ is a probability measure with support in a ball of radius r centered at zero. We shall assume for simplicity that $H(y,r)$ has nonzero density inside this ball. The function $F(x,r)$ is differentiable and $F(x,r) \to F(x)$ uniformly over every compact set as $r \to 0$. It is now possible to minimize the nonsmooth function $F(x)$ by computing stochastic quasigradients for smooth functions $F(x,r)$ and find the optimal solution of the initial problem by letting $r \to 0$. This idea was proposed in [11] and studied further in [12]. It is not actually necessary to calculate the integral in (13) – it is sufficient to compute ξ^s using equations (10)–(12), but at point $x^s + y^s$ rather than point x^s, where y^s is a random variable distributed according to $H(y,r_s)$. In this case (10) becomes:

$$\xi^s = \sum_{i=1}^{n} \frac{f(x^s + y^s + \delta_s e_i, \omega_{i,1}^s) - f(x^s + y^s, \omega_{i,2}^s)}{\delta_s} e_i \tag{14}$$

The most commonly used distribution $H(y,r)$ is uniform distribution on an n-dimensional cube of side r. If we want to have convergence with probability one we should choose r_s such that $\delta_s / r_s \to 0$ and $(r_s - r_{s+1})/\rho_s \to 0$. In practical computations it is also advisable to choose the smoothing parameter r_s in a similar way to δ_s, using one of the adaptive procedures discussed above. Smoothing also has beneficial side effects in that it improves the behavior of the deterministic function $F(x)$. In the case where $F(x)$ may be written as the sum of two functions, one with a distinct global minimum and the other with highly oscillatory behavior, smoothing may help to overcome the influence of the oscillations, which may otherwise lead the process to local minima far from the global one. Thus it can sometimes be useful to smooth the objective function even if we can obtain a gradient $f_x(x,\omega)$. In this case we should take a large value for the smoothing parameter r_s on the first few iterations, decreasing it as we approach the optimal point. The points at which r_s should be decreased may be determined using the values of additional estimates, such as those described below in Section 3 or given by (9). Everything said about the choice of the finite-difference parameter δ_s is also valid for the choice of the

smoothing parameter, including the connection between the step size and the smoothing parameter and the possibility of interactive control of r_s. The only difference is that a decrease in r_s does not lead to an increase in the variance of ξ^s and that it is preferable to have $\delta_s < r_s$. This is also reflected in the stochastic optimization software developed at IIASA.

All of the methods discussed so far use only the information available at the current point or in its immediate vicinity. We shall now discuss some more general ways of choosing the step direction which take into account the information obtained at previous points.

2.5. Averaging over preceding iterations

The definition of a stochastic quasigradient given in (3) allows us to use information obtained at previous points as the iterations proceed; this information may sometimes lead to faster convergence to the vicinity of the optimal point. One possible way of using such information is to average the stochastic quasigradients obtained in preceding iterations via a procedure such as (9). The v^s obtained in this way may then be used in method (2). This is another way of smoothing out randomness and neutralizing such characteristics of deterministic behavior as curved valleys and oscillations. Methods of this type may be viewed as stochastic analogues of conjugate gradient methods and were first proposed in [13]. We can choose ξ^s according to any of (5), (6), (10), (11), (12), or (14). Since $v^s \to F_x(x^s)$ under rather general conditions (see [1,2]), method (9) can be considered as an alternative to method (6) for deriving precise estimates of gradient $F_x(x)$. This method has an advantage over (6) in that it provides a natural way of using rough estimates of $F_x(x^s)$ on the first few iterations and then gradually increasing the accuracy as the current point approaches the optimal point. In this case (9) can be incorporated in the adaptive procedures used to choose the smoothing parameter and the step in the finite-difference approximation.

However, it is not necessary to always take $\alpha_s \to 0$, because we have convergence for any $0 \le \alpha_s \le 1$. Sometimes it is even advantageous to take $\alpha_s \equiv \alpha$ = constant, because in this case more emphasis is placed on information obtained in recent iterations. In general, the greater the randomness, the smaller the value of α that should be taken. Another averaging technique is given by

$$v^{s+1} = \frac{1}{M_s} \sum_{i=s-M_s+1}^{s} \xi^s \tag{15}$$

where M_s is the size of the memory, which may be fixed.

2.6. Using second-order information

There is strong evidence that in some cases setting

$$v^s = A_s \xi^s \tag{16}$$

may bring about considerable improvements in performance. Here ξ^s can be chosen in any of the ways discussed above. Matrix A_s should be positive definite and take into account both the second-order behavior of function $F(x)$ and the structure of the random part of the problem. One possible way of obtaining second-order information is to use analogues of quasi-Newton methods to update matrix A_s. To implement this approach, which was proposed by Wets in [3], it is necessary to have $\|\xi^s - F_x(x^s)\| \to 0$.

3. CHOICE OF STEP SIZE

The simplest way of choosing the step-size sequence in (2) is to do it before starting the iterative process. Convergence theory suggests that any series with the properties:

$$\rho_s > 0, \quad \sum_{s=1}^{\infty} \rho_s = \infty, \quad \sum_{s=1}^{\infty} \rho_s^2 < \infty \tag{17}$$

can be used as a sequence of step sizes. In addition, it may be necessary to take into account relations between the step size and such things as the smoothing parameter or the step in a finite-difference approximation. Relations of this type have been briefly described in the preceding sections. In most cases the choice $\rho_s \sim C/s$, which obviously satisfies (17), provides the best possible asymptotic rate of convergence. However, since we are mainly concerned with reaching the vicinity of the solution, rule (17) is of limited use because a wide variety of sequences can be modified to satisfy it. The other disadvantage of choosing the step-size sequence in advance is that this approach does not make any use of the valuable information which accumulates during solution. These "programmed" methods thus perform relatively badly in the majority of cases.

The best strategy therefore seems to be to choose the step size using an interactive method. It is assumed that the user can monitor the progress of the optimization process and can intervene to change the value of the step size or other parameters. This decision should be based on the behavior of the estimates $\hat{F}(x^s)$ of the current value of the objective function. The estimates may be very rough and are generally calculated using only one observation per iteration, as in the following example:

$$\hat{F}^s = \frac{1}{s}\sum_{i=1}^{s} f(x^i,\omega^i) \tag{18}$$

It appears that although the observations $f(x^s,\omega^s)$ may vary greatly, the $\hat{F}^s$ display much more regular behavior. Monitoring the behavior of some components of the vector x^s in addition to the $\hat{F}^s$ also seems to be useful. One possible implementation of the interactive approach may proceed along the following lines:

(i) The user first chooses the value of the step size and keeps it constant for a number of iterations (usually 10–20). During this period the values of the estimate $\hat{F}^s$ and some of the components of the vector x^s are displayed, possibly with some additional information.

(ii) The user decides on a new value for the step size using the available information. Three different cases may occur:

- The current step size is too large. In this case both the values of the estimate $\hat{F}^s$ and the values of the monitored components of x^s exhibit random jumps. It is necessary to decrease the step size.
- The current step size is just right. In this case the estimates decrease steadily and some of the monitored components of the current vector x^s also exhibit regular behavior (steadily decrease or increase). This means that the user may keep the step size constant until oscillations occur in the estimate $\hat{F}^s$ and/or in the components of the current vector x^s.
- The current step size is too small. In this case the estimate $\hat{F}^s$ will begin to change slowly, or simply fluctuate, after the first few iterations, while the change in x^s is negligible. It is necessary to increase the step size.

(iii) Continue with the iterations, periodically performing step (ii), until changes in the step size no longer result in any distinct trend in either the function estimate or the current vector x^s, which will oscillate around some point. This will indicate that the current point is close to the solution.

This method of choosing the step size requires an experienced user, but we have found that the necessary skills are quickly developed by trial and error. The main reasons for adopting an interactive approach may be summarized as follows:

- Interactive methods make the best use of the information which accumulates during the optimization process.
- Because the precise value of the objective function is not available, it is impossible to use the rules for changing the step size developed in deterministic optimization (e.g., line searches).
- Stochastic effects make it extremely difficult to define formally when the step size is "too big" or "too small"; theoretical research has not thrown any light on this problem.

The main disadvantage of the interactive approach is that much of the user's time is wasted if it takes the computer a long time to make one observation $f(x^s,\omega^s)$. For this reason a great effort has been made to develop *automatic adaptive* ways of choosing the step size, in which the value of the step size is chosen on the basis of information obtained at all or some of the previous points x^i, $i = \overline{1,s}$. Methods of this type are considered in [14–20]. The approach described in the following sections involves the estimate of some measures of algorithm performance which we denote by $\Phi^i(\bar{x}^s,u^s)$, where $\bar{x}^s$ represents the whole sequence $\{x^1,x^2,\ldots,x^s\}$ and u^s the set of parameters used in the estimate. In general, algorithm performance measures are attempts to formalize the notions of "oscillatory behavior" and "regular behavior" used in interactive step-size regulation, and possess one or more of the following properties:

- the algorithm performance measure is quite large when the algorithm exhibits distinct regular behavior, i.e., when the estimates of the function value decrease or the components of the current vector x^s show a distinct trend;

- the algorithm performance measure becomes small and even changes its sign if the estimates of the current function value stop improving or if the current point starts to oscillate chaotically;
- the algorithm performance measure is large far from the solution and small in the immediate vicinity of the optimal point.

Automatic adaptive methods for choosing the step size begin with some reasonably large value of the step size, which is kept constant as long as the value of the algorithm performance measure remains high, and then decreases when the performance measure becomes less than some prescribed value. The behavior of the algorithm usually becomes regular again after a decrease in the step size, and the value of the performance measure increases; after a number of iterations oscillations set in and the value of the performance measure once again decreases. This is a sign that it is time to decrease the step size. A rather general convergence result concerning such adaptive piecewise-linear methods of changing the step size is given in [18]. However, in many cases it is difficult to determine how close the current point is to the optimal point using only one such measure – a more reliable decision can be made using several of the measures described below. Unfortunately, it is not possible to come to any general conclusions as to which performance measure is the "best" for all stochastic optimization problems. Moreover, both the values of the parameters used to estimate the performance measure and the value of the performance measure at which the step size should be decreased are different for different problems. Therefore if we fix these parameters once and for all we may achieve the same poor performance as if we had chosen the whole sequence of step sizes prior to the optimization process. Thus, it is necessary to tune the parameters of automatic adaptive methods to different classes of problems, and the interactive approach can be very useful here. An experienced user would have little difficulty in using the values of the performance measures to determine the correct points at which to change the step size, and in learning what type of performance measure behavior requires an increase or a decrease in the step size. The interactive approach is of particular use if one iteration is not very time-consuming and there are a number of similar problems to be solved. In this case the user can identify the most valuable measures of performance in the first few runs, fix their parameters and incorporate this knowledge in automatic adaptive step-size selection methods for the remaining problems.

Although interactive methods usually provide the quickest means of reaching the solution, they cannot always be implemented, and in this case automatic adaptive methods prove to be very useful. The stochastic optimization package STO developed at IIASA and the Kiev stochastic and nondifferentiable optimization package NDO both give the user the choice between automatic adaptive methods and interactive methods of determining the step size. Below we describe some particular measures of algorithm performance and methods of choosing the step size.

The main indicators used to evaluate the performance of an algorithm are estimates of such things as the value of the objective function and its gradient. The averaging procedure (9) may be used to estimate the value of the gradient, as described earlier in this paper. The main advantage of this procedure is that it allows us to obtain estimates of the mean values of the random variables without extensive sampling at each iteration, since a very limited number of observations (usually only one) is made at each iteration. This estimate, although poor at the beginning, becomes more and more accurate as the iterations proceed. One example of such an estimate is (18), which is a special case of the more general formula

$$\hat{F}^{s+1} = (1 - \gamma_s)\hat{F}^s + \gamma_s f(x^s, \omega^s) \quad . \tag{19}$$

Any observation μ^s with the property

$$\mathrm{E}(\mu^s \mid x^1, x^2, \ldots, x^s) = F(x^s) + \alpha_s \tag{20}$$

can be used instead of $f(x^s, \omega^s)$ in (19), where $\alpha_s \to 0$. For example, (6) would do. In order to get $\lim_{s\to\infty} | \hat{F}^s - F(x^s) | = 0$ it is necessary to have $\rho_s / \gamma_s \to 0$. However, estimate (18) assigns all observations of function values the same weight. This sometimes leads to considerable bias in the estimate for all the iterations the user can afford to run. Therefore for practical purposes it is sometimes more useful to adopt procedures of the type described in Section 2 for the estimation of gradients. These include estimate (19) with fixed $\gamma_s \equiv \gamma$, where $\gamma \sim 0.01$–0.05, and the method in which the average is taken over the preceding M_s iterations:

$$\hat{F}^s = \frac{1}{M_s} \sum_{i=s-M_s+1}^{s} f(x^i, \omega^i) \tag{21}$$

Although these estimates do not converge asymptotically to $F(x^s)$, they place more emphasis on observations made at recent points. All of the estimates $\hat{F}^s$ may also be used in an interactive mode to determine the step size, as described above. In addition, the values of the parameters used to determine the step size may also be chosen interactively. For example, the values of parameters b_1 and b_2 in

$$\rho_s = \frac{b_1}{b_2 + s}$$

can be made to depend on the behavior of $\hat{F}^s$.

We shall now describe some automatic adaptive rules for choosing the step size. The important point as regards implementation is how to choose the initial value of the step size ρ_0. We suggest that the value of a stochastic quasigradient ξ^0 should first be computed at the initial point, and that the initial value of the step size should then be chosen such that

$$\rho_0 l \| \xi^0 \| \sim D \quad ,$$

where $l \sim 10$–20 and D is a rough estimate of the size of the domain in which we believe the optimal solution to be located. This means that it is possible to reach the vicinity of each point in this domain within the first 20 iterations or so.

3.1. Ratio of function estimate to the path length

Before beginning the iterations we choose the initial step size ρ_0, two positive constants α_1 and α_2, a sequence M_s and an integer $\hat{M}$. After every $\hat{M}$ iterations we revise the value of the step size in the following way:

(i) Compute the quantity

$$\Phi^1(\bar{x}^s, u^s) = \frac{\hat{F}^{s-M_s} - \hat{F}^s}{q(s, M_s)} \tag{22}$$

Here the u^s are the averaging parameters used in the estimation of both $\hat{F}^s$ and M_s, while $\bar{x}^s$ is again the whole sequence of points preceding x^s. The quantity

$$q(s, M_s) = \sum_{i=s-M_s}^{s-1} \| x^{i+1} - x^i \| \tag{23}$$

is the length of the path taken by the algorithm during the preceding M_s iterations. The function $\Phi^1(\bar{x}^s, u^s)$ is another example of a measure which can be used to assess algorithm performance.

(ii) Take a new value of the step size:

$$\rho_{s+1} = \begin{cases} \alpha_1 \rho_s & \text{if } \Phi^1(\bar{x}^s, u^s) \le \alpha_2 \\ \rho_s & \text{otherwise} \end{cases} \tag{24}$$

In this method the step size is changed at most once every $\hat{M}$ iterations. This is essential because function Φ^1 changes slowly, and if its value is less than α_2 at iteration number s it is likely that the same will be true at iteration number $s+1$. Therefore $\hat{M}$ should lie in the range 5–20. This procedure can be modified in various ways, such as continuing for $\hat{M}$ iterations with a fixed step size, then starting to compare values until inequality (24) is satisfied whereupon the step size is reduced. We then wait another $\hat{M}$ iterations and repeat the procedure. Recommended values of α_1 and α_2 lie within the ranges 0.5–0.9 and 0.005–0.1, respectively. The number M_s may be chosen to be constant and equal to $\hat{M}$. If we have a number of similar problems it is very useful to make the first run in a semi-automatic mode, i.e., to intervene in the optimization process to improve the values of parameters α_1 , α_2 , $\hat{M}$ – the new values can then be used in a fully automatic mode to solve the remaining problems.

This algorithm is by no means convergent in the traditional sense, but it outperformed traditional choices like C/s in numerical experiments because it normally reaches the vicinity of the optimal point more quickly. However, it is possible to safeguard convergence by considering a second sequence C/s, where C is small, and switching to this sequence if the step size recommended by (24) falls below a certain value. This step size regulation was introduced in [15].

3.2. Use of gradient estimates

Take $\Phi^2 = \hat{G}^s$ instead of $\Phi^1(\bar{x}^s, u^s)$ in (24), where $\hat{G}^s$ is one of the gradient estimates discussed above, and the u^s represent all the parameters used, including averaging parameters and the frequency of changes in the step size.

3.3. Ratio of progress and path

The quantity $\|x^{s-M_s} - x^s\|$ represents the progress made by the algorithm between iteration number $s - M_s$ and iteration number s. If we keep the step size constant, the algorithm begans to oscillate chaotically after reaching some neighborhood of the optimal point. The smaller the value of the step size, the smaller the neighborhood at which this occurs, and thus the total path between iterations s and $s - M_s$ begins to grow compared with the distance between points x^{s-M_s} and x^s. This means that the function

$$\Phi^3(\bar{x}^s, u^s) = \frac{\|x^{s-M_s} - x^s\|}{\sum_{i=s-M_s}^{s-1} \|x^{i+1} - x^i\|} \tag{25}$$

can be used as a performance measure in equation (24).

3.4. Analogues of line search techniques

The decision as to whether (and how) to change the step size may be based on the values of the scalar product of adjacent step directions. If we have $(\xi^{s-1}, \xi^s) > 0$, then this may be a sign that regular behavior prevails over stochastic behavior, the function is decreasing in the step direction and the step size should be increased. Due to stochastic effects the function will very often increase rather than decrease, but in the long run the number of bad choices will be less than the number of correct decisions. Analogously, if this inequality does not hold then the step size should be decreased. The rule for changing the step size is thus basically as follows:

$$\rho_{s+1} = \begin{cases} \rho_s & \text{if } -\alpha_1 \le (\xi^{s-1}, \xi^s) \le \alpha_1 \\ \alpha_2 \rho_s & \text{if } (\xi^{s-1}, \xi^s) > \alpha_1 \\ \alpha_3 \rho_s & \text{if } (\xi^{s-1}, \xi^s) \le -\alpha_1 \end{cases} \tag{26}$$

where the values of α_1, α_2, α_3 (recommended values $\alpha_1 \sim 0.4$–0.8 , $1 < \alpha_2 \le 1.3$ and $0.7 \le \alpha_3 < 1$) should be chosen before starting the iterations. It is also advisable to have upper and lower bounds on the step size to avoid divergence. Sometimes it is convenient to normalize the vectors of step directions, i.e., $\|\xi^s\| = 1$. The lower bound may decrease as the iterations proceed. This method may also be applied to the choice of a vector step size, treating some (or all)

variables or groups of variables separately. A number of different methods based on the use of scalar products of adjacent step directions to control the step size have been developed by Uriasiev [19], Pflug [16], and Ruszczynski and Syski [20].

4. IIASA IMPLEMENTATION

The interactive stochastic optimization package implemented at IIASA (STO) is based on the same ideas as the package for stochastic and nondifferentiable optimization developed in Kiev (NDO). It allows the user to choose between interactive and automatic modes and makes available the stochastic quasigradient methods described in Sections 2 and 3. In the interactive mode the program offers the user the opportunity to change the step parameters and the methods by which the step size and step direction are chosen *during the course of the iterations*. The user can also stop the iterative process and obtain a more precise estimate of the value of the objective function before continuing. The package is written in FORTRAN-77.

Before initiating the optimization process the user has to:

(i) Provide a subroutine UF which calculates the value of function $f(x,\omega)$ for fixed x and ω and, optionally, a subroutine UG which computes the gradient $f_x(x,\omega)$ of this function; the function evaluation subroutine should be of the form:

```
      FUNCTION UF(N,X)
      DIMENSION X(N)
Calculation of f(x,ω)
      RETURN
      END
```

Here N is the dimension of the vector of variables X. (Note that the implementation on the IIASA VAX actually requires the subroutine to be entered in lower-case letters rather than capitals.) A description of the subroutine which calculates a quasigradient is given later in this paper.

(ii) Compile these subroutines with the source code to obtain an executable module.

(iii) Provide at least one of the following additional data files:

- algorithm control file (used only in the non-interactive option)
- parameter file (used only in the interactive option)
- initial data file (should always be present)

All of these files are described in some detail later in the paper.

The optimization process can then begin. The program first asks the user a series of questions regarding the required mode (interactive or automatic), method of step size regulation, choice of step direction, etc. These questions appear on the monitor and should be answered from the keyboard or by reference to a data file. We shall represent the dialogue as follows:

Question? *Answer*

with the user's response given in italics. The first question is

Interactive mode? reply yes or no *yes/no*

To choose the interactive option the user should type in *yes* (or *y*); to select the automatic option he should answer *no* (or *n*). In the latter case the program would ask no further questions, but would read all the necessary information from the algorithm control file (which is usually numbered 2 – under UNIX conventions its name is fort.2). The iterative process would then begin, terminating after 10,000 iterations if no other stopping criterion is fulfilled. The algorithm control file must contain answers to all of the following questions except those concerned either with dialogue during the iterations or with the parameter file (such questions are marked with an asterisk * below). This file is given a name only for ease of reference – the important thing for the user is its number.

Assume now that the user has chosen the interactive option by answering *yes* to the first question. The program then asks

parameter file? *(number)* *

The user should respond either with the number of the file of default parameters or with the number of the file in which the current values of the algorithm parameters are stored. The file of default parameters is provided with the program and has the name fort.12 (under UNIX conventions); thus, to refer the program to the default file the user should answer *12*. The purpose of this file is to help the user to set the values of algorithm parameters in the ensuing dialogue and also to store such improved values as may be discovered by the user

through trial and error. If the user assigns the algorithm parameters any values other than those in the default file, the new values become the default values in subsequent runs of the program. This file is optional.

The program then asks

read parameter file? reply yes or no *yes/no* *

The answer *yes* implies that the file specified in the previous question exists, and that default parameter values are stored in this file. In this case, when asking the user about parameter values, the program will read the default option in the parameter file and reproduce it on the screen together with the question. If the user accepts this default value he should respond with *0* (zero); otherwise he should enter his own value, which will become the new default value.

The answer *no* means that no default values are available at the moment. In this case the program will form a new default file (labeled with the number given as an answer to the previous question); its contents will be based on the user's answers to future questions. This new default file, once formed, can be used in subsequent runs.

The next question is

number of variables? *(number)*

to which the user should respond with the dimension of the vector of variables x. He is then asked

Initial data file? *(number)*

and should reply with the number of the initial data file. This file should contain the following elements (in exactly this order):

- The initial point, which should be a sequence of numbers separated by commas or other delimiters.
- Any additional data required by subroutines UF or UG if such data exists and the user chooses to put it in the initial data file (optional).
- Information about the constraints (described in more detail below)

The program then asks

step size regulation? *is*

Here *is* is a positive integer from the set {1,2,3,4,6,7}, where the different values of *is* correspond to different ways of choosing the step size. (The integer 5 is reserved for an option currently under development.)

is Definition

1 Adaptive automatic step size regulation (24) based on algorithm performance function (22) and function estimate (18).

2 Manual step size regulation based on algorithm performance function (22) and function estimate (18).

3 Adaptive automatic step size regulation (24) using algorithm performance measure (22) and a function estimate based on a finite number of previous observations (21).

4 Manual step size regulation based on the same estimates of algorithm performance as for *is* = 3.

6 Automatic step size regulation using algorithm performance measure (24) and function estimate (19) with fixed γ_s.

7 Manual step size regulation based on the same estimates of algorithm performance as for *is* = 6.

The difference between adaptive automatic and manual step size regulation (see *is* = 1,2) is that in the first case the step size is chosen automatically, although the user may terminate the iterations at specified points and continue with another step size regulation, while in the second case the user changes the value of the step size himself. Both step size regulations are based on the same estimates of function value and algorithm performance.

The next question is

step direction? (5 figures) *id1 id2 id3 id4 id5*

The user has to respond with five figures which specify various ways of choosing the step direction, e.g., 11111. We shall refer to these figures as *id1*, *id2*, *id3*, *id4* and *id5*. The subroutine which estimates the step direction makes some number of initial observations $\bar{\xi}^{i,s}$ at each step; these are then averaged in some way to obtain the vector ξ^s, and the final step direction v^s is calculated using both ξ^s and values of v^i for $i < s$.

The value of *id1* specifies the nature of the initial observations $\bar{\xi}^{i,s}$.

id1 Definition

1 A direct observation of a stochastic quasigradient is available for $\bar{\xi}^{i,s}$ and the user has to specify a subroutine UG to calculate it:

```
      SUBROUTINE UG(N,X,G)
      DIMENSION X(N),G(N)
Calculation of a stochastic quasigradient
      RETURN
      END
```

where G(N) is an observation of a stochastic quasigradient.

2 Central finite-difference approximation of the gradient as in (11).

3 The $\bar{\xi}^{i,s}$ are calculated using random search techniques (12).

4 Forward finite-difference approximation of the initial observations $\bar{\xi}^{i,s}$ as in (10).

5 Central finite-difference approximation of the gradient as in (11). All observations of the function used in one observation of $\bar{\xi}^{i,s}$ are made with the same values of random parameters ω.

6 The $\bar{\xi}^{i,s}$ are calculated using random search techniques (12). All observations of the function used in one observation of $\bar{\xi}^{i,s}$ are made with the same values of random parameters ω.

7 Forward finite-difference approximation of the initial observations $\bar{\xi}^{i,s}$ as in (10). All observations of the function used in one observation of $\bar{\xi}^{i,s}$ are made with the same values of random parameters ω.

Note that for *id1* = 5,6,7 all observations of the function used in one observation of $\bar{\xi}^{i,s}$ are made with the same values of random parameters ω. In this case the user should write a function UF which supports this feature as follows:

```
FUNCTION UF(N,X)
DIMENSION X(N)
COMMON/OMEG/LO,MO
If LO=1 and MO=1 then obtain new values
of random factors ω and set MO=0.
Make an observation of the function at point x.
RETURN
END
```

The second figure *id2* determines the point at which observations are made:

id2 Definition

1 The initial direction is calculated at the current point x^s

2 The initial direction is calculated at a point chosen randomly from among those in the neighborhood of the current point x^s

The value of *id3* defines the way in which the step in a finite-difference or random search approximation of $\bar{\xi}^{i,s}$ is chosen:

id3 Definition

1 The approximation step is fixed. The observations of the objective function at point x^s originally used to obtain gradient observations $\bar{\xi}^{i,s}$ are not used to update the estimate of the function employed for step size regulation.

2 The ratio δ_s / ρ_s of the step in the finite-difference approximation to the step size of the algorithm is fixed (see (10)–(12)). The observations of the objective function at point x^s originally used to obtain gradient observations $\bar{\xi}^{i,s}$ are not used to update the estimate of the function employed for step size regulation.

3 The approximation step is fixed. The observations described for *id3* = 1,2 above *are* used to update the current estimate of the objective function.

4 The ratio δ_s / ρ_s of the step in the finite difference approximation to the step size of the algorithm is fixed (see (10)–(12)). The observations described for *id3* = 1,2 above *are* used to update the current estimate of the objective function.

The fourth figure *id4* defines the type of averaging used to obtain ξ^s from observations $\bar{\xi}^{i,s}$.

id4 Definition

1 No averaging, $\xi^s = \bar{\xi}^{i,s}$, $i = 1$.

2 Number of samples > 1.

The value of *id5* specifies the way in which the final step direction v^s is obtained from previous values of v^s and from ξ^s.

id5 Definition

1 No previous information is used. The final vector v^s is simply set equal to ξ^s.

2 (9) is used.

3 A positive number n_3 is provided by the user. Set $k(s) = \max\{k : kn_3 + 1 \le s\}$. Then the final direction v^s is computed from (15), where $M_s = s - k(s)n_3 + 1$.

4 No previous information is used. The final vector v^s is set equal to ξ^s and is normalized.

5 (9) is used. The final vector v^s is normalized.

6 A positive number n_3 is provided by the user. Let $k(s) = \max\{k : kn_3 + 1 \le s\}$. Then the final direction v^s is computed from (15), where $M_s = s - k(s)n_3 + 1$. The final vector v^s is normalized.

The program then asks about the type of constraints present in the problem:

constraints? *(number)*

The answer (in the present implementation) must be 1,2,3 or 4. These values define the type of constraints present and correspond to the following options:

1 There are no constraints at all.

2 There are upper and lower bounds on the variables. The values of these bounds should be given at the end of the initial data file in the form of strings of numbers separated by commas or other delimiters. The string containing the upper bounds should come first.

3 There is one constraint $\sum_{i=1}^{n} a_i x_i \leq b$. The coefficients a_i should be given at the end of the initial data file. The string containing the coefficients of linear form comes first and then, on a separate line, the right-hand side.

4 There are general linear constraints $b_l \leq Ax \leq b_u$. In this case the program computes a projection on these constraints at each iteration, using the quadratic programming package SOL/QPSOL [21]. The previous point x^{s-1} is used as the initial approximation to the solution at iteration number s. The precision of projection also varies, being rough during the first few iterations and improving as the process proceeds. All of these facilities are intended to reduce the amount of computation required at each iteration.
The following information should appear at the end of the initial data file (in exactly this order):
   - upper bounds on variables x
   - lower bounds on variables x
   - upper bounds b_u on general linear constraints
   - lower bounds b_l on general linear constraints
   - number of nonzero elements in matrix A
   - numbers of nonzero elements in the columns of matrix A
   - nonzero elements of matrix A in increasing order of column number
   - row numbers of nonzero elements, in the same order as the elements themselves

The next question is

termination condition? *(number)*

There is currently only one possible answer, which is 1. This means that the iterations terminate when the step size becomes smaller than some value specified by the user. Additional options are under development.

The program then asks the user whether the interactive mode is required *during* the iterations:

interactive mode during iterations? reply yes or no *yes/no* *

Note that the answer to this question should not be included in the algorithm control file for the completely non-interactive option (as indicated by the asterisk). If the user replies *yes* (or *y*), the program will allow the user to change the parameters of the algorithm and even the algorithm itself during the course of the iterations. If the answer is *no* (or *n*) the program will not communicate with the user during the iterations but will instead ask the following two questions:

number of iterations? *(number)*

This is the number of iterations that should be performed before the process terminates (if it has not already been terminated by some other condition). It is necessary to put an answer to this question in the algorithm control file for the completely non-interactive option.

extra output? reply yes or no *yes/no*

This is the program's way of asking the user whether information about the iterations should be saved. Note that these two questions do not appear if the user has chosen to run the program in the interactive mode during the iterations.

Now comes a group of questions about step direction parameters. These questions depend on the values of *id1*, *id2*, *id3*, *id4* and *id5* given previously (see the discussion of answers to the question **step direction?**).

If *id1* = 4,5 then the question

number of random directions? *(number)*

appears. The required answer is M_s from (12).

If *id2* = 2 the user is asked

relation between step size and neighborhood? *(number)*

The answer is the ratio of the step size to the size of the neighborhood (of the current point) from which the observation point is chosen (i.e., r_s/ρ_s in the discussion of (13)).

If *id3* = 1,3 and *id1* $\neq$ 1 the program asks

step in finite difference approximation? *(number)*

The required answer is the value of step δ_s in the finite-difference or random search approximation (10)–(12) of the gradient observation. In this case δ_s is fixed. However, if *id3* = 2,4 the question

relation between step in finite difference approximation and step size? *(number)*

appears. The answer is the ratio δ_s/ρ_s of the finite-difference approximation step to the algorithm step size.

If *id4* = 2 the program asks

number of samples? *(number)*

This is the number of samples taken at one point to obtain the averaged estimate (see, for instance, N in (6)).

The question

discount rate? *(number)*

appears if *id5* = 2,5. The required answer is the (fixed) value of α_s from (9). However, if *id5* = 3,6 the program asks

number of averaging steps? *(number)*

The user should respond with the value of n_3 (see earlier discussion of *id5* options).

We now have a group of questions concerning the values of step size parameters. Which questions appear depends on the way in which the step size is being chosen (see earlier discussion of the question **step size regulation?**).

If the user has chosen automatic step size regulation (*is* = 1,3,6) he will be asked the following four questions:

initial step size? *(number)*

This is ρ_0.

multiplier? *(number)*

The required answer is α_1 from (24).

frequency of step size changes? *(number)*

The user should give the value of $\hat{M}$ (see discussion of (24)).

lower bound on function decrease? *(number)*

This is α_2 from (24).

However, if the user has chosen to regulate the step size interactively (*is* = 2,4,7) he will only be asked

value of step size? *(number)*

The following questions appear only if there are general linear constraints, i.e., if the answer to the question **constraints?** is 4:

number of general linear constraints? *(number)*

correspondence between step size and accuracy of projection? *(number)*

The answer to the first question is obvious but the second requires some explanation. In order to keep the amount of computation to a minimum, the accuracy τ_s of projection is linked to the value of the step size: $\tau_s = \varepsilon\rho_s$. This leads to only rough projection during the first few iterations (when the step size is large) and more precise projection as the current point approaches the optimal

point. The required answer to the last question is the value of ε; recommended values lie in the range 0–1.

Another group of questions is concerned with the estimates of the objective function and also affects the choice of step size:

size of memory? *(number)*

The answer is M_s from (22), which in this implementation is fixed. If the step size regulation is defined by *is* = 6,7 the program asks

multiplier for function averaging? *(number)*

The user should give the value of γ_s in (19), which is fixed.

With the answers to these questions the algorithm control file for the non-interactive option is complete. The rest of this section describes the ways in which the algorithm parameters and the algorithm itself may be modified during the course of the iterations. This may be done only if the answer to the question **Interactive mode during iterations? reply yes or no** was *yes*. In this case the program will now perform the first iteration and produce a string of information something like this:

1 0. 7505.826 7505.826 0. 1.000 100.458 109.575

Here the first number is the number of the current iteration, the second is the value of some algorithm performance measure (see (22), (25) for examples of such functions), the third is the estimate of the value of the objective function at the current point (see (18), (19), (21) for examples of such estimates), the fourth is an observation of $f(x^s,\omega^s)$, the fifth currently has no meaning and always contains 0, the sixth is the step size, and the rest are values of variables x_i^s (the default is that only the values of the first two such variables are displayed). After this string the following question will appear:

continue? reply "space",step,dir,var,estim,go,yes or no *

This gives the user the opportunity to continue without any change, to alter the frequency of communication, to change the step size or step direction parameters, to display variables other than the first two, to stop at the current point and obtain a precise estimate of the value of the objective function, to switch from interactive to automatic mode, or to terminate the iterations and continue the solution with another algorithm. We shall now describe all of these options in some detail.

"space" If the user hits the space bar nothing will change and the program will perform another 10 iterations. The information about the process is displayed after each iteration; after the 10-th iteration the user is once again given the opportunity to make changes (the question **continue? reply "space",step...** appears).

step This means that the user wants to change the step size parameters (but not the step size regulation itself) and all the related questions will be repeated. Default or previous values of the step parameters will appear on the screen together with the questions.

dir This means that the user wants to change the step direction parameters (but not the way in which the step direction is chosen) and the questions concerned with this will be repeated. Default or previous values of the direction parameters will appear on the screen together with the questions.

var In this case the quantity and/or the selection of variables displayed on the screen may be changed. The following questions will appear:

number of printed variables? *(number)* *

i.e., if the user wants to print out the values of four variables rather than the default two, he answers *4*.

printed variables? *(number, number,....)* *

Here the user specifies which particular variables he wants displayed by giving the numbers of the chosen variables separated by commas.

Questions concerning the frequency of communication will also appear here (see description of response *yes* below).

estim In this case the program will stop at the current point and estimate the value of the objective function. The following questions will appear:

number of observations? *(number)* *

i.e., the number of observations to be made, and

message frequency? *(number)* *

i.e., the number of observations after which the current estimate is displayed. The user is also asked for the point at which the estimate should be made:

what point? reply current, new or exit *current/new/exit* *

If the answer is *new* the program asks the question:

where to find new point? reply screen or file *screen/file* *

If the user wants to enter the new point from the keyboard he should reply *screen* (or *s*). He should then type the desired point on a new line, separating the components by commas. If, however, the new point is stored in some file the response should be *file* (or *f*) and the user is then asked

file number? *(number)* *

The answer is obviously the number of the file containing the new point. This new point is taken as the starting point for future iterations if the user answers *yes* to the following question:

replace current point by new? reply yes or no *yes/no* *

which appears when the estimation of the objective function at the new point has been completed. This facility makes it possible to exchange the current point for an arbitrary point chosen by the user and also to make precise estimations at arbitrary points. Finally, if the answer to the question **what point? reply current, new or exit** is *exit* the estimation procedure will end and the iterations will continue.

go This means that the user does not want to continue in the interactive mode; he wants the process to proceed automatically. This is useful once the algorithm parameters have been established and also in the case when one iteration is very time-consuming. The user is then asked

number of iterations? *(number)* *

i.e., the total number of iterations before termination. After this the program has no more communication with the user and terminates after the specified number of iterations.

yes In this case the frequency of communication can be changed. The following questions appear:

output frequency? *(number)* *

This is the number of iterations after which information about the process is displayed on the screen (the default value is 1, i.e., a string of information is printed after every iteration).

dialogue frequency? *(number)* *

This is the number of process information strings (see above) printed before the user is asked the question **continue? reply space,step,dir,var,estim,yes or no.** The default is 10, i.e., the user is given ten strings of information about the process before he is asked whether he wishes to make any changes.

no This means that the user wishes either to terminate the iterations or change the method. The program asks:

continue? reply "space",yes or no *"space"/yes/no* *

Here hitting the space bar means that the user wishes to proceed with the iterations using the same method, maybe returning to the initial point (see below); *yes* means he wishes to change the way in which the step size and/or step direction are chosen (the program will ask further questions about this – see below); *no* means that he wishes to terminate the iterations completely (some self-explanatory questions will then appear). If the user answers *"space"* or *yes* the program will ask

return to initial values? reply yes or no *yes/no* *

and the user should give the appropriate response.

The very first appearance of the question **continue? reply space,step,dir, var,estim,yes or no** is followed by the question

least value of step size? *(number)* *

The answer is the least permissible value of the step size. If the current step size is less than this value then the iterations will terminate. In other cases the process terminates after 10,000 iterations with a question about whether to continue or not.

Everything that appears on the screen during the interactive dialogue automatically also goes to file number 15 (fort.15 in UNIX). This makes it possible to study the process after it has terminated.

This section provides some idea of the capabilities of the package of stochastic optimization subroutines STO available at IIASA. The implementation described here is the first version, and development of the second continues. This revised version will include methods for solving certain special problems, in particular problems with recourse, and new methods for step size regulation will be introduced.

5. SOME NUMERICAL EXPERIMENTS

5.1. Facility location problem

We first consider a simple model of facility location in a stochastic environment. Suppose that we have to determine the amounts x_i of materials, facilities, etc., required at points $i = \overline{1,n}$ in order to meet a demand ω_i. The demand is random, and all we know is its distribution function $P\{\omega_1 \le \bar{\omega}_1,\dots,\omega_n \le \bar{\omega}_n\} = H(\bar{\omega})$. The actual value $\omega = (\omega_1,\dots,\omega_n)$ of the demand is not known when the decision concerning the $x = (x_1,\dots,x_n)$ has to be made. Assume that we have made a decision x about the distribution of facilities and then found that the actual demand is ω. We have to pay for both oversupply and shortfalls, i.e., the penalty charged at the i-th location is $\psi_1^i(\omega_i - x_i)$ if $\omega_i \ge x_i$ and $\psi_2^i(x_i - \omega_i)$ if $\omega_i < x_i$, where the functions $\psi_1^i(y)$ and $\psi_2^i(y)$ are nondecreasing. In the simplest case these functions are linear and the total penalty for fixed x and ω is $\sum_{i=1}^n \max\{a_i(\omega_i - x_i),\ b_i(x_i - \omega_i)\}$, where $a_i \ge 0,\ b_i \ge 0$, $i = \overline{1,n}$. In most cases it is reasonable to select x in such a way that the average penalty is at a minimum, i.e., to minimize the following function:

$$F(x) = E_\omega f(x,\omega) = E_\omega \sum_{i=1}^{n} \max\{a_i(\omega_i - x_i),\ b_i(x_i - \omega_i)\} =$$

$$\int \sum_{i=1}^{n} \max\{a_i(\omega_i - x_i),\ b_i(x_i - \omega_i)\} dH(\omega) \qquad (27)$$

This approach can easily be generalized to deal with more complex facility location models (see [1,15,22]). The numerical experiment presented here is basically an application of the facility location model described above to the

problem of high school location in Turin, Italy (see [15,22]). In this example n is the number of districts in the city (23 in this case), ω_i is the number of students who want to attend schools in district i, and x_i is the capacity of schools in district i. It is assumed that a student living in district i will choose a school in district j with probability p_{ij}, where

$$p_{ij} = \frac{e^{-\lambda c_{ij}}}{\sum_{j=1}^{n} e^{-\lambda c_{ij}}}$$

and c_{ij} is proportional to the distance between districts i and j. The values of c_{ij} are taken from [15], as are the values of the parameters ($\lambda = 0.15$ and $a_i = b_i = 1.0$ for all i). The demand ω_i is assessed by assigning individual students to a school in a particular district on the basis of probabilities p_{ij}, thus simulating the student's choice of school. In order to reduce the amount of computation the number of students was scaled. Table 1 gives the resulting solution (the number of places that should be provided), together with the total number of students actually attending schools in each district.

TABLE 1 The solution of the problem of high school location in Turin, Italy [15,22]

District	1	2	3	4	5	6	7	8
Number of students	14.0	13.0	15.0	11.0	14.0	14.0	11.0	12.0
Solution	17.9	13.0	18.9	19.0	16.0	13.9	10.8	10.2
District	9	10	11	12	13	14	15	16
Number of students	12.0	23.0	26.0	23.0	22.0	18.0	14.0	15.0
Solution	13.0	19.8	26.0	20.0	16.6	15.7	14.0	13.0
District	17	18	19	20	21	22	23	
Number of students	14.0	14.0	10.0	10.0	5.0	8.0	21.0	
Solution	13.0	15.7	10.0	10.1	5.0	10.3	17.0	

All real data was divided by a scaling factor of 100. We also have the constraint $\sum_{i=1}^{n} x_i = M$, where M is the total number of students in the city divided by 100 (339 in this case). Once ω has been obtained it is quite easy to calculate a stochastic quasigradient. We can use vector $\xi^s = (\xi_1^s, \xi_2^s, \ldots, \xi_n^s)$ in method (2), where

$$\xi_i^s = \begin{cases} -a_i & \text{if } \omega_i^s \geq x_i^s \end{cases}$$

Here ω_i^s is the demand in district i (calculated by simulating the students' behavior) at iteration number s, and x_i^s is the i-th component of the solution at this iteration. The initial point was obtained by assuming that each student goes to school in his native district. After extensive averaging, the value of the objective function at this point was found to be 74.2 – the optimal value is 55.9. We shall first present results obtained using the interactive option for changing the step size, i.e., results obtained by giving the answer *2* to the question **step size regulation?** The step direction was specified as 11111, i.e, a direct observation of a stochastic quasigradient is available, this observation is made at the current point, the approximation step is fixed, there is no averaging, and no previous information is used. The size of the memory available for calculating the performance measure (22) was set at 10. Table 2 reproduces the information displayed on the monitor during the first 30 iterations.

TABLE 2 Information displayed during the first 30 iterations (facility location problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_4	x_{23}
2	−0.335	73.696	75.304	1.000	13.435	19.435
3	−0.172	73.739	73.826	1.000	14.565	18.565
4	−0.029	72.500	68.783	1.000	15.783	17.783
5	0.200	68.243	51.217	1.000	16.826	16.826
6	0.201	67.275	62.435	1.000	17.522	17.522
7	0.196	66.435	61.391	1.000	18.391	16.391
8	0.172	66.326	65.565	1.000	19.435	15.435
9	0.108	67.952	80.957	1.000	18.391	16.391
10	0.082	68.539	73.826	1.000	17.609	17.609
12	0.119	68.609	84.609	1.000	19.522	19.522
14	0.017	67.491	55.304	1.000	19.696	17.696
16	0.010	66.011	59.565	1.000	19.435	19.435
18	0.064	65.174	52.348	1.000	19.348	19.348
20	0.066	64.287	64.435	1.000	19.522	17.522
22	0.097	64.221	56.174	1.000	19.609	15.609
24	0.076	63.181	51.043	1.000	17.609	15.609
26	0.062	63.271	60.870	1.000	19.870	15.870
28	0.025	63.221	64.696	1.000	19.696	17.696
30	0.036	63.032	42.522	1.000	17.696	17.696

The observations of $f(x^s,\omega^s)$ given in Table 1 do not provide any clues as to whether the algorithm is improving the values of the objective function $F(x^s)$ or not. At first sight these observations appear to oscillate randomly between 40 and 80. By contrast, the estimates $\hat{F}^s$ of the function $F(x^s)$ display much more stable behavior, generally decreasing during the first 22 iterations from 73 to 64 and then stabilizing around the values 63–64 with some small

oscillations. Looking at the behavior of the two selected variables, we see that their values show a steady increase or decrease until iteration number 8 for x_4 and iteration number 5 for x_{23}. In later iterations both variables exhibit oscillatory behavior. The value of the performance measure during the first 4 iterations is negative, due to the instability of the initial estimates. It then begins to increase and reaches approximately 0.2, reflecting the regular behavior of the estimate $\hat{F}_s$. After this it decreases in an oscillatory fashion to the range 0.03–0.06. All of this indicates that it is time to decrease the step size.

TABLE 3 Information displayed during iterations 31–59 (facility location problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_4	x_{23}
31	0.045	62.379	42.783	0.500	18.087	17.087
33	0.025	62.295	62.783	0.500	18.261	16.261
35	0.052	61.652	52.609	0.500	19.391	16.391
37	0.063	61.565	46.957	0.500	19.348	16.348
39	0.079	61.318	52.261	0.500	19.261	17.261
41	0.050	61.211	68.174	0.500	19.174	16.174
43	0.051	60.815	51.304	0.500	18.261	16.261
45	0.070	60.452	57.913	0.500	17.304	16.304
47	0.059	60.279	45.652	0.500	17.348	15.348
49	0.035	60.277	64.957	0.500	18.391	15.391
51	0.043	60.104	61.739	0.500	18.652	14.652
53	0.017	60.133	64.696	0.500	18.565	14.565
55	0.017	60.240	67.043	0.500	18.652	14.652
57	−0.030	60.819	65.565	0.500	18.565	15.565
59	−0.052	61.189	85.391	0.500	18.609	16.609

After changing the step size, the estimates of $F(x^s)$ decreased steadily during iterations 31–51, and then started to increase during iterations 52–59 (see Table 3). The performance measure first increased, reaching a level of 0.05–0.07 between iterations 35 and 47 before dropping back to negative values. It is necessary to decrease the step size once again.

We decided to stop after iteration number 80 (see Table 4) and estimate the value of the objective function at the current point. The average after the first 500 observations was 56.53, which shows that we are fairly close to the optimal solution. Note that this estimate is considerably lower than the value of $\hat{F}^s$ (61.0) given in the table. This is due to the fact that the estimate $\hat{F}^s$ is calculated from (18) including only one additional observation $f(x^s,\omega^s)$ per iteration, and it therefore includes observations made at early points which are clearly far from the optimum. Nevertheless, this estimate is still useful in determining the value of the step size because it reflects the general behavior

TABLE 4 Information displayed during iterations 62–80 (facility location problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_4	x_{23}
62	−0.098	61.971	92.557	0.200	17.652	17.052
66	−0.067	61.684	46.713	0.200	18.104	17.504
70	0.013	61.353	61.026	0.200	18.226	16.826
74	0.087	61.167	55.739	0.200	17.861	16.461
78	0.061	60.832	58.104	0.200	18.296	16.896
80	0.020	61.001	78.557	0.200	18.348	17.348

of the algorithm. Subsequent iterations improved the value of the objective function only marginally (see Table 5).

TABLE 5 Information displayed during iterations 90–3070 (facility location problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_4	x_{23}
90	0.063	60.601	54.087	0.200	17.930	17.730
100	0.143	59.876	45.739	0.100	18.287	17.687
120	0.022	59.579	57.670	0.100	18.330	17.530
140	0.061	58.890	45.374	0.100	18.626	17.826
160	−0.011	59.161	56.278	0.100	19.226	17.626
180	0.319	58.761	44.744	0.020	19.379	17.299
200	0.008	58.608	49.144	0.020	19.237	17.277
300	0.317	57.847	43.322	0.020	18.946	17.146
400	−0.368	57.627	81.986	0.005	18.909	17.129
500	0.270	57.584	63.554	0.005	18.869	17.099
800	−0.830	57.012	58.455	0.001	18.967	17.017
1100	3.773	57.071	66.512	0.0003	18.980	17.000
1570	1.521	56.858	79.613	0.0001	18.983	16.998
2070	0.916	56.629	46.567	0.0001	18.975	16.998
2570	−0.874	56.603	71.741	0.0001	18.978	17.001
3070	0.118	56.425	55.729	0.0001	18.982	17.000

Our final estimate of the objective function was 56.0, which is close to the optimal solution.

The same results can be obtained by automatic regulation of the step size. In this case we give the answer *1* to the question **step size regulation?**, i.e., adaptive automatic step size regulation (24) using function estimate (18). We also set

initial step size	1.0
multiplier	0.7
frequency of step size change	15
lower bound on function decrease	0.02
size of memory	15

(see the description of the step size parameters in Section 4). The results are presented in Table 6.

TABLE 6 Information displayed during iterations 2–1200 (facility location problem, adaptive automatic step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_4	x_{23}
2	3.663	77.826	60.261	1.000	10.739	20.739
4	1.590	72.522	57.739	1.000	12.739	18.739
6	1.091	69.232	54.522	1.000	14.826	20.826
8	0.892	65.457	48.174	1.000	14.826	18.826
10	0.736	63.609	56.087	1.000	16.913	18.913
15	0.453	64.980	65.652	1.000	18.130	18.130
20	0.071	64.435	58.522	1.000	17.522	19.522
30	0.023	64.304	49.652	1.000	19.783	15.783
50	0.007	61.951	49.391	1.000	17.609	15.609
70	0.017	61.563	68.696	0.700	15.104	15.104
100	0.017	60.593	90.195	0.490	18.665	18.245
150	0.017	60.246	65.349	0.240	20.166	16.855
200	0.054	59.526	48.282	0.082	19.657	17.223
300	0.036	59.277	50.012	0.028	19.131	17.248
400	−0.035	58.495	58.695	0.020	19.074	16.999
500	−0.100	58.440	63.486	0.010	18.903	16.986
600	0.143	57.936	36.450	0.007	18.913	16.984
700	0.446	57.683	47.760	0.003	18.955	16.998
800	−0.024	57.387	43.263	0.003	18.945	16.995
900	0.412	57.116	50.086	0.002	18.975	16.958
1000	0.430	57.006	43.503	0.001	18.947	16.969
1100	−0.063	56.726	76.801	0.001	18.969	16.997
1200	0.165	56.623	65.457	0.001	18.989	16.994

The value of the objective function at the final point (average of 4000 observations) is 56.2, which is close to the optimal value. The behavior of the algorithm was virtually the same as in the interactive case: quite a reasonable approximation of the optimal solution was obtained after 100–150 iterations, with little improvement being observed thereafter.

5.2. Control of water resources

This example is taken from work by A. Prekopa and T. Szantai. An extended description of the problem together with a solution obtained by reduction to a special type of nonlinear programming problem is given in [23]. Here we shall show how the problem can be solved using stochastic quasigradient methods. The basic aim is to control the level of water in Lake Balaton (a large, shallow lake in western Hungary). A certain volume of water ω_i flows into the lake from rivers, rainfall, etc., in time period i. This inflow varies randomly from one period to another, but it is possible to derive its probabilistic distribution from

previous observations. The control parameter is the amount x_i of water released from the lake into the River Danube in each time period; the objective is to maximize the probability of the water level lying within specified bounds. It turns out that a reasonable control policy can be determined by considering only two consecutive periods of time, which in this example are measured in months. After appropriate transformations we arrive at the following problem (for details see [23]):

$$\max_{x_1,x_2} P\{Z(x_1,x_2)\}$$

$$0 \le x_1 \le R$$

$$0 \le x_2 \le R \quad ,$$

where the set $Z(x_1,x_2)$ is defined as follows:

$$Z(x_1,x_2) = \{(\omega_1,\omega_2): a_1 \le \omega_1 - x_1 \le b_1 \,,\ a_2 \le \omega_2 - x_1 - x_2 \le b_2\}$$

Here a_i, b_i are respectively the lower and upper bounds on the "generalized" water level: in this particular example we took $a_1 = a_2 = -205$, $b_1 = b_2 = 95$, $R = 200$. The random water inputs ω_1 and ω_2 have a joint normal distribution $H(\omega_1,\omega_2)$ with expectations $E(\omega_1) = -28.07$, $E(\omega_2) = -59.43$ and covariance matrix

$$C = \begin{bmatrix} 3636.12 & 4660.51 \\ 4660.51 & 10121.36 \end{bmatrix}$$

Let $\chi(x_1,x_2,\omega_1,\omega_2)$ denote the indicator function of the set $Z(x_1,x_2)$, i.e.,

$$\chi(x_1,x_2,\omega_1,\omega_2) = \begin{cases} 1 & \text{if } (\omega_1,\omega_2) \in Z(x_1,x_2) \\ 0 & \text{otherwise} \end{cases}$$

The problem then becomes

$$\max_{x \in X} \int \chi(x_1,x_2,\omega_1,\omega_2)\, dH(\omega_1,\omega_2)$$

and can be solved using stochastic quasigradient methods. We took (95,95) as the initial point; the value of the objective function at this point was 0.32. According to [23], the optimal solution is (2,0), with an objective function value of 0.857. We decided to solve the problem using a finite-difference

approximation of a stochastic quasigradient. Below we demonstrate how our interactive software package STO may be used to solve this problem, specifying interactive step size regulation (option 2) and step direction 21124, (i.e., taking a central finite-difference approximation of the gradient, calculating the step direction at the current point, with a fixed approximation step, a number of samples greater than 1, no previous information, and such that the step direction vector has unit norm).

The parameters were set at the following values:

step in finite difference approximation	10.0
number of samples	5
value of step size	10.0
size of memory	20

The results are given in Table 7.

TABLE 7 Information displayed during iterations 1–110 (water management problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\tilde{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_1	x_2
1	0.	0.	0.	10.000	102.071	102.071
2	1.000	0.	0.	10.000	102.071	102.071
4	0.025	0.250	1.000	10.000	106.543	93.127
6	0.011	0.333	0.	10.000	113.614	110.198
8	0.007	0.375	0.	10.000	106.543	113.127
10	0.006	0.400	0.	10.000	106.543	93.127
15	0.003	0.333	0.	10.000	83.944	101.254
20	0.002	0.350	0.	10.000	68.397	90.630
30	0.001	0.467	0.	10.000	18.240	93.229
40	0.000	0.475	1.000	10.000	48.678	63.727
50	0.000	0.500	1.000	10.000	41.277	29.097
60	0.000	0.567	1.000	10.000	0.	43.004
70	0.000	0.571	1.000	10.000	1.056	30.405
80	0.000	0.588	1.000	10.000	1.386	14.142
90	0.000	0.600	1.000	10.000	0.	24.142
100	0.000	0.610	1.000	10.000	7.071	20.000
110	0.000	0.609	1.000	10.000	10.000	0.

After iteration 110 we stopped and estimated the value of the function at the current point on the basis of 4000 observations – we obtained a value of 0.843, which is close to the optimal value. Subsequent iterations improved the value of the objective function only marginally (see Table 8).

After iteration 200 we changed the step in the finite-difference approximation to 1.0. The value of the objective function at the final point was 0.85, i.e., we had reached the optimal value. However, the values of the controls were far

TABLE 8 Information displayed during iterations 120–8090 (water management problem, interactive step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_1	x_2
120	0.000	0.625	1.000	10.000	10.000	17.071
150	0.000	0.673	1.000	1.000	0.106	1.707
200	0.001	0.720	0.	1.000	2.707	6.309
390	0.005	0.792	1.000	0.100	3.071	7.835
590	−0.001	0.797	0.	0.100	1.787	8.110
1090	0.000	0.829	1.000	0.100	3.463	6.392
2090	0.000	0.845	1.000	0.100	0.383	5.538
3090	−0.005	0.852	0.	0.010	0.161	4.895
4090	−0.004	0.854	1.000	0.005	0.071	5.049
5090	0.004	0.856	1.000	0.005	0.064	4.955
6090	−0.002	0.855	1.000	0.005	0.106	4.980
7090	0.007	0.856	1.000	0.001	0.016	4.970
8090	0.005	0.855	1.000	0.001	0.020	4.985

from the solution due to the flatness of the function around the optimum.

5.3. Determining the parameters in a closed loop control law for stochastic dynamical systems with delay

We have so far considered only static optimization problems. However, all of the techniques described above can also be applied to many classes of dynamical stochastic optimization problems. The example that we shall consider was suggested by A. Wierzbicki and is the problem of finding the optimal control parameters in a closed loop control law for a linear dynamical system disturbed by random noise. The state equations include response delay and may be written as follows:

$$z_{t+1} = a_t z_t + u_{t-k} + \omega_t \ , \quad t = \overline{0,T} \tag{28}$$

$$z_0 = 1 \ , \quad u_{-i} = 0 \ , \quad i = \overline{0,k} \ ,$$

where t is a discrete time, z_t is the state of the dynamical system at time t, u_t is the value of the control at time t, and ω_t is the random noise at time t. In this particular example the ω_t were taken to be distributed uniformly over the interval $[-b,b]$ and such that ω_i and ω_j are uncorrelated for $i \neq j$. However, neither this particular type of distribution nor these correlation properties are prerequisites for the use of the methods described in the preceding sections. The controls u_t were chosen according to the following closed loop control law:

$$u_t = x_1(-z_t - x_2 \sum_{\tau=0}^{t} z_\tau) \ . \tag{29}$$

where the decision parameters are $x_1 \geq 0$ and $x_2 \geq 0$.

The objective is to minimize the deviation of the state of the system from zero. We may therefore state the problem as follows: minimize the objective function

$$F(x_1,x_2) = E_\omega \sum_{t=1}^{T} z_t^2 \tag{30}$$

with respect to the control law parameters x_1 and x_2, subject to constraints (28) and (29) and non-negativity constraints on x_1, x_2. We solved the problem with the following parameter values: time horizon $T = 100$, delay $k = 5$, state equation coefficient $a = 0.9$, bounds for random noise $b = 0.1$. With these values the optimal control parameters are $x_1 = 0.1$, $x_2 = 0$; the value of the objective function obtained after 10,000 observations was 4.52. It was discovered during preliminary runs that for $x_1 \geq 0.3$, $x_2 \geq 0.1$ the system becomes unstable and therefore these values were taken as upper bounds for the variables.

We set the initial point equal to the upper bounds $x_1^0 = 0.3$, $x_2^0 = 0.1$; the value of the objective function at this point (based on 3000 observations) was 422.56. We chose automatic step size regulation (option 1), i.e., the step size changes are based upon performance function (22). The step direction was specified as 71114, i.e., taking a forward finite-difference approximation of the gradient of the random objective function $f(x,\omega)$ with all observations of the function needed for one gradient evaluation made at the same value of the noise; with a fixed finite difference step and the finite-difference evaluation performed at the current point; without averaging; using no previous information and normalizing the resulting step direction. The parameters of the algorithm were as follows:

step in finite difference approximation	0.0001
initial step size	0.1
multiplier (for diminishing the step size)	0.85
frequency of step size change (actually the frequency with which the step size is reviewed)	15

lower bound on function decrease (the lowest value of performance function (22) which does not lead to a decrease in the step size)	0.09
size of memory (for evaluating (22))	15
least value of step size (stopping criterion)	0.000001

The results of the calculations are given in Table 9.

TABLE 9 Information displayed during iterations 1–120 (control law problem, automatic step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_1	x_2
1	0.	8.141	8.141	0.100	0.232	0.027
2	18.570	6.284	4.427	0.100	0.149	0.
3	12.231	5.695	4.517	0.100	0.054	0.
4	7.093	6.013	6.968	0.100	0.097	0.
5	6.727	5.450	3.199	0.100	0.075	0.
10	3.428	5.056	4.084	0.100	0.073	0.
15	2.416	4.759	4.254	0.100	0.103	0.099
20	0.421	4.733	4.214	0.100	0.029	0.
30	0.119	4.651	5.326	0.100	0.052	0.
40	0.058	4.615	4.896	0.100	0.050	0.
50	−0.012	4.631	5.143	0.085	0.071	0.
70	0.001	4.668	5.131	0.072	0.112	0.
90	0.005	4.665	4.943	0.061	0.076	0.059
100	0.042	4.621	3.481	0.052	0.076	0.
120	0.033	4.601	4.872	0.044	0.094	0.

We stopped after iteration 120 to estimate the value of the objective function, which was calculated to be 4.54 after 3000 observations and is fairly close to the optimal value. Subsequent iterations improved the solution only marginally (see Table 10).

This example once again demonstrates the characteristic behavior of stochastic optimization algorithms: the neighborhood of the optimal solution is reached reasonably rapidly; oscillations then occur in this neighborhood and the current approximation to the optimal solution improves slowly.

The nature of stochastic quasigradient algorithms allows easy extension of model (28)–(30) to multivariable and nonlinear systems.

TABLE 10 Information displayed during iterations 150–1500 (control law problem, automatic step size regulation)

Iter. no.	Performance measure	Estimate $\hat{F}^s$ of $F(x^s)$	Observation of $f(x^s,\omega^s)$	Step size	x_1	x_2
150	0.044	4.517	3.776	0.032	0.102	0.000
170	0.084	4.485	4.234	0.023	0.101	0.
200	−0.015	4.473	5.224	0.017	0.101	0.
240	0.087	4.473	4.413	0.012	0.087	0.009
300	−0.155	4.503	4.478	0.006	0.095	0.
340	0.036	4.491	4.958	0.005	0.090	0.
400	0.089	4.501	4.973	0.002	0.093	0.000
440	−0.299	4.512	4.544	0.001	0.092	0.003
500	−0.131	4.512	3.571	0.001	0.098	0.
540	−0.416	4.502	4.437	0.001	0.101	0.
600	0.225	4.515	4.789	0.001	0.102	0.
640	0.710	4.508	3.704	0.001	0.101	0.
700	0.046	4.501	4.120	0.001	0.101	0.
800	0.079	4.517	4.633	0.000	0.100	0.000
900	−1.183	4.533	5.070	0.000	0.099	0.000
1000	2.700	4.534	4.860	0.000	0.099	0.000
1500	29.344	4.504	4.621	0.000	0.099	0.000

REFERENCES

1. Yu. Ermoliev. *Methods of Stochastic Programming* (in Russian). Nauka, Moscow, 1976.
2. Yu. Ermoliev. Stochastic quasigradient methods and their applications to systems optimization. *Stochastics*, 9 (1983) 1–36.
3. R. J.-B. Wets. Stochastic programming: solution techniques and approximation schemes. In *Mathematical Programming. The State of the Art*. Springer-Verlag, 1983.
4. L. Nazareth and R. J-B. Wets. Algorithms for stochastic programs: the case of nonstochastic tenders. Working Paper WP-83-5, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1983.
5. P. Kall. *Stochastic Linear Programming*. Springer-Verlag, Berlin, 1976.
6. A. Prekopa. On probabilistic constrained programming. In H. Kuhn (Ed.), *Proceedings of the Princeton Symposium on Mathematical Programming*, pp. 113–138. Princeton University Press, Princeton, 1970.
7. H. Robbins and S. Monroe. A stochastic approximation method. *Ann. Math. Stat.*, 22 (1951) 400–407.
8. J. Kiefer and J. Wolfowitz. Stochastic approximation of the maximum of a regression function. *Ann. Math. Stat.*, 23 (1952) 462–466.
9. B. Fox and H. Niedereiter. Lectures on quasi-Monte-Carlo methods given at IIASA, 1983.
10. H. Kushner. Asymptotic behavior of stochastic approximation and large deviations. Lecture given at IIASA, 1983.
11. Yu. Ermoliev and E. Nurminski. Limit extremum problems. *Kibernetika*, 4 (1973) 130–132.
12. A. Gupal. *Stochastic Methods for Solution of Nonsmooth Optimization Problems*. Naukova Dumka, Kiev, 1979.

13. A. Gupal and A. Basenov. Stochastic analogue of conjugate gradient methods. *Kibernetika*, 1 (1972) 79–83.

14. H. Kesten. Accelerated stochastic approximation. *Ann. Math. Statist.*, 29 (1958) 41–59.

15. Yu. Ermoliev, G. Leonardi and J. Vira. The stochastic quasigradient method applied to a facility location problem. Working Paper WP-81-14, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.

16. G. Pflug. On the determination of the step size in stochastic quasigradient methods. Collaborative Paper CP-83-25, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1983.

17. V. Fabian. Stochastic approximation method. *Czechoslovakian Mathematical Journal*, 10 (1960) 191–200.

18. N. Chepurnoi. Dissertation. V. Glushkov Institute of Cybernetics, Kiev, 1982.

19. S. Uriasiev. Dissertation. V. Glushkov Institute of Cybernetics, Kiev, 1982.

20. A. Ruszczynski and W. Syski. A method of aggregate stochastic subgradients with on-line stepsize rules for convex stochastic optimization problems. *Mathematical Programming Study*, forthcoming.

21. P. Gill, W. Murray, M. Saunders, and M. Wright. User's guide for SOL/QPSOL: a FORTRAN package for quadratic programming. Technical Report SOL 83-7, Systems Optimization Laboratory, Stanford University, 1983.

22. Yu. Ermoliev and G. Leonardi. Some proposals for stochastic facility location models. Working Paper WP-80-176, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.

23. A. Prekopa and T. Szantai. On optimal regulation of a storage level with application to the water level regulation of a lake. *European Journal of Operations Research*, 3 (1979) 175–189.