WORKING PAPER

INTERACTIVE PROGRAM SQG-PC FOR SOLVING STOCHASTIC PROGRAMMING PROBLEMS ON IBM PC/XT/AT COMPATIBLES
– User Guide –

Alexei Gaivoronski

February 1988
WP-88-11

IIASA
International Institute
for Applied Systems Analysis

INTERACTIVE PROGRAM SQG-PC FOR SOLVING STOCHASTIC PROGRAMMING PROBLEMS ON IBM PC/XT/AT COMPATIBLES
– User Guide –

Alexei Gaivoronski

February 1988
WP-88-11

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

This paper contains a detailed description of the SQG-PC program (Stochastic Quasi-Gradients for Personal Computers), which is one of the results of the Optimization Project in the System and Decision Sciences Program.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

CONTENTS

INTERACTIVE PROGRAM SQG-PC FOR SOLVING STOCHASTIC PROGRAMMING PROBLEMS ON IBM PC/XT/AT COMPATIBLES
– User Guide –

Alexei Gaivoronski

1. INTRODUCTION

SQG-PC was developed in The Optimization Project, Systems and Decision Sciences Program of IIASA by Alexei Gaivoronski*. It can be used for optimization of systems which functioning depends on random parameters and/or essential systems characteristics are measured with error. The program is intended for IBM PC/XT/AT compatibles and runs under DOS 2.1 and higher. Some of its essential features are the following:

- Both automatic and interactive modes of problem solving. In automatic mode the programs runs free from the user intervention using default or previously selected values of algorithm parameters. In interactive mode user has considerable measure of control on optimization process, including possibility to change algorithm, tune algorithm parameters, restart from an arbitrary point, etc.
- A considerable selection of algorithms based on stochastic quasi-gradient techniques, involving different rules for choosing stepsize and step direction (averaging, smoothing, random search, finite differences etc.).
- Possibility to monitor process evolution both numerically and graphically.
- The program can solve also deterministic nonlinear programming problems, although it would be less efficient than specifically designed methods like Quasi-Newton.

*Currently working at the V. Glushkov Institute of Cybernetics, Kiev, USSR.

2. THEORETICAL BACKGROUND

SQG-PC solves the following problem:

$$\text{minimize } Ef(x, \omega) = F(x) \tag{1}$$

$$\text{subject to } x \in X$$

Here the set X belongs to Euclidean space R^n, x are decision variables and ω random parameters defined on the appropriate probability space. The main difficulty in solving this problem is that taking mathematical expectation in (1) involves multidimensional integration which can not be afforded for any reasonable number of random parameters. Therefore numerical methods for solving (1) are centered either on approximation of the objective function or the utilization of the observations of the random function $f(x, \omega)$.

Optimization models of the type (1) can be used to formalize many real life situations in industrial and economical modeling and were first put forward in [1] in the form of stochastic programs with recourse. Good overview of the state of the art including algorithms, implementations and applications is given in [2], various approaches to the problem are described in [3]–[6].

The approach utilized in the SQG-PC involves iterative process which starts from the initial point x^0. On each particular iteration no attempt is made to compute exact values of the objective function $F(x)$ or its derivatives, instead only a limited number of observations of the random parameters ω is made, the values of the function $f(x, \omega)$ or its gradient are computed and the step direction ξ^s is obtained. The algorithm looks as follows:

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

where ρ_s is the stepsize, ξ^s is the step direction and π_X stands for projection on the set X. The vector ξ^s should possess the following property:

$$E(\xi^s / x^0, \ldots, x^s) = F_x(x^s) + a_s \tag{3}$$

where a_s is a small vanishing term. The vector ξ^s is called stochastic quasi-gradient and the method (2) is called the method of projection of stochastic quasi-gradients [7]. For unconstrained differentiable problems the techniques is known as stochastic approximation [8]. Some of the relevant publications on the method (2) are [9]–[17]. Different approaches for solving the problem (1), and stochastic problem with recourse can be found in [18]–[21].

Here is the simplest result concerning the convergence of technique (2):

THEOREM *Suppose that*

1 X *is compact convex subset of* R^n

2 $F(x)$ *is finite convex function in some vicinity of* X

3 $\rho_s > 0, \rho_s \to 0, \sum_{s=0}^{\infty} \rho_s = \infty, \sum_{s=0}^{\infty} \rho_s^2 < \infty$

$$E(\|\xi^s - F_x(x^s)\|^2 / x^0, \ldots, x^s) < C < \infty, \|a_s\| \to 0$$

Then $x^s \to X^*$ *with probability 1 where* $X^* = \{x^* : x^* \in X, F(x^*) = \min_{x \in X} F(x)\}$

As can be seen from condition 3 stepsize rules can be quite varied and what is more important the theorem only assures asymptotic convergence and no indication is given as to how fast this would occur.

The important thing in implementation is not asymptotic convergence, but rather convergence to some vicinity of optimum in reasonable number of iterations. To achieve this a lot of work is needed to define practical stepsize rules and step direction rules.

This is the main emphasis in SQG-PC implementation of the stochastic quasi-gradient methods together with elaborate interactive mode.

3. PROGRAM SETUP

3.1. Description of diskettes

SQG-PC comes on a Distribution diskette, which contains SQG-PC library and auxillary files. The Example diskette contains water resource problem discussed in detail in [17]. This problem is used in this paper to explain main features of SQG-PC, for the short description see Appendix.

Contents of the Distribution diskette:

- SQG.LIB — optimization library with compiled SQG-PC subroutines
- MAIN.OBJ — main object file, which should be linked with SQG.LIB and user defined function and random number generator to obtain the executable program

- PARAM	file with default values of the algorithm parameters, should always be present in the same directory as the executable program, which reads and modifies it.
- RUNPAR.MNU	file, which contains the information about screens, which appear in the interactive option. Should be present in the same directory as executable program.
- RUNTABLE.MNU	- * -
- STEP.MNU	- * -
- GRAPHPAR.MNU	- * -
- CONST.MNU	- * -
- QUIT.MNU	- * -
- SETUP.MNU	- * -
- RUN1.MNU	- * -
- INFO.MNU	- * -
- SILENT.MNU	- * -
- DIR.MNU	- * -
- RUN2.MNU	- * -
- BEGIN.MNU	- * -
- AXIS.MNU	- * -
- MAIN.MNU	- * -
- SQG.LNK	example of the file with linking information, the user should create similar file with information on object files and libraries.
- LSQG.BAT	example of the batch link file, it refers to the file SQG.LNK. The executable program on the Example diskette was created by executing this file.

The Example diskette contains the executable program, which solves water resources example. All files on this diskette except JD.FOR and JD.OBJ are necessary for successful solution. To execute the example create subdirectory on the hard disk and copy all files from the Example diskette to this directory, then make this directory default and execute command JD. It is possible also to execute the example by simply inserting the copy of the Example diskette without write protection in the floppy disk drive and execute JD from this drive, although in this case there would not be enough space to keep record of the solution process.

Contents of the Example diskette:

- JD.EXE	executable program, which was created by the batch file LSQG.BAT from the Distribution diskette
- JD.OBJ	compiled object file with the minimized function and the random number generator
- JD.FOR	Fortran text of the example function, subgradient and random number generator
- JD.CON	information about the problem constraints
- JD.DAT	information about the objective function
- NOR.DAT	parameters of random parameters distribution
- PARAM	auxillary file, necessary for the execution of the program. Its composition is similar to the file PARAM from the distribution diskette, but version, present on the Example diskette contains algorithm parameters specifically tuned for the water resources example.
- JDBA.INI	initial point
- RUNPAR.MNU	auxillary file identical to one on the Distribution diskette
- RUNTABLE.MNU	- * -
- STEP.MNU	- * -
- GRAPHPAR.MNU	- * -
- CONST.MNU	- * -
- QUIT.MNU	- * -
- SETUP.MNU	- * -
- RUN1.MNU	- * -
- INFO.MNU	- * -
- SILENT.MNU	- * -
- DIR.MNU	- * -
- RUN2.MNU	- * -
- BEGIN.MNU	- * -
- AXIS.MNU	- * -
- MAIN.MNU	- * -

3.2. System requirements

SQG-PC runs on IBM PC/XT/AT and compatibles under DOS operating system, version 2.1 and higher. The computer should be equipped with hard disk, have 300 KB memory free for the program and CGA or EGA card, however in the latter case screen appearance would be the same as with CGA card. The source of SQG-PC is written in FORTRAN-77, however library file SQG.LIB is compiled by IBM Professional FORTRAN compiler, version 1.22 and therefore needs 8087 or 80287 mathematical coprocessor. The user function should be compiled by the same compiler. The SQG-PC uses NOLIMITS library from M|E|F Environmental for screen and keyboard control, and in order to solve new problems user should have this library too. Excluding the compiler and graphical library memory the SQG-PC needs approximately 240 KB hard disk memory for optimization library and auxillary files and 360 KB additional hard disk memory for each problem.

3.3. Program setup

In what follows there are some suggestions of how to organize the hard disk directories for SQG-PC. An experienced user can organize them differently.

- create directories COMPILER, SQG and PROBLEM in the root directory;
- copy contents of the Distribution diskette to the SQG directory;
- copy IBM Professional Fortran compiler, related libraries, NOLIMITS library and linker to the COMPILER directory;
- the directory PROBLEM will be reserved for the solution of particular optimization problem. Create in this directory the file UF.FOR with Fortran text defining the objective function, possibly its gradient and random number generator. Conventions concerning this file will be described in the section 3.4. Instead of UF it is possible to use any other name (JD in case of water resource problem on the Example diskette).
- compile the file UF.FOR, the simplest way to do this is to execute command ..\PROFORT UF.FOR from default directory PROBLEM, for more details see the manual for the IBM Professional FORTRAN Compiler. This will create the file UF.OBJ in the PROBLEM directory

- create executable file UF.EXE in the PROBLEM directory. This can be done by executing batch file LSQG.BAT which is supplied on the Distribution diskette. This file contains one line:

  ..\compiler\link @sqg.lnk

  It refers to the file SQG.LNK, also supplied on the Distribution diskette, which contains the following lines:

  ..\sqg\main.obj+uf.obj

  uf.exe /nodefaultlibrarysearch/X:1024

  ..\sqg\sqg.lib+

  ..\compiler\forlib3.mef+

  ..\compiler\profort.lib

- create the file with information about constraints in the directory PROBLEM. The name of this file is defined by user and supplied to the program as described in the section 4.6, where conventions for defining this file are described also. In the water resources problem this file has the name JD.CON
- create the file which defines the initial point x^0 for the iteration process in the directory PROBLEM. The name of this file is defined by the user and supplied to the program as described in the section 4.3, where conventions for defining this file are described also. In the water resources problem this file has the name JDBA.INI
- create files which contain information about objective function and random parameters. These files are optional and their organization is defined by user (if he/she needs them for defining objective function). In the water resources problem these files have names JD.DAT and NOR.DAT
- copy to the directory PROBLEM file PARAM from the directory SQG and all files with extension MNU from the same directory;
- now you have everything necessary to run the problem. Just make directory PROBLEM default and execute UF (or whatever name you have chosen for the executable file). GOOD LUCK.

3.4. How to define the objective function

The listing of the file JD.FOR with water resources objective function is contained in the Appendix. It is a good idea to have a look at it now. It consists of the following FORTRAN subprograms:

– function uf	defines the objective function $f(x, \omega)$
– subroutine ug	defines the subgradient of the objective function, this subroutine is optional
– subroutine ranv	the header part of the random number generator, it reads parameters of the normal distribution from the file NOR.DAT and computes the value of the normal random variable by summing the specified number of uniformly distributed variables;
– function uran	simple multiplicative-additive random number generator, which generates successive uniformly distributed pseudo-random numbers

Random number generator is part of this file and is totally defined by the user. It is called only from within function UF or subroutine UG. The only requirement is that it supplies successively the independent values of random variables with the distribution of the random parameters of the problem. For the function UF and subroutine UG there are some guidelines which will be described below. In what follows names of identifiers, which can be changed by the user are given in small letters, obligatory parts are given in capital letters and comments within program are given in italics.

3.4.1. Definition of the function UF

```
FUNCTION UF (n, x)
DIMENSION x(n)
COMMON/OMEG/ lomeg,momeg
```

– *reading initial data, in case of water resource problem from file JD.DAT. This reading should occur only during the very first call of the function*

```
IF(lomeg.eq.0) GO TO 301
IF(momeg.ne.1) GO TO 302
```

momeg=0

301 CONTINUE

- *call to random number generator which supplied new random number*

302 CONTINUE

- *computation of the value of the function $f(x, \omega)$, say userval.*

UF=userval

RETURN

END

With each call to UF it should return the value of the function $f(x^s, \omega^s)$. The value of the current point x^s is transferred to UF through array $x(n)$, which has dimension n of the decision variables of the problem. Common block OMEG is needed to arrest sometimes generation of the new random number. This feature is used in one of the finite difference options, where finite differences are calculated for the fixed value of random parameters. If user does not intend to use this feature the function UF can be simplified:

FUNCTION UF (n, x)

DIMENSION x(n)

- *reading initial data, in case of water resource problem from file JD.DAT. This reading should occur only during the very first call of the function*
- *call to random number generator which supplies new random number*
- *computation of the value of the function $f(x,^s, \omega^s)$, say userval.*

UF=userval

RETURN

END

In this version each successive value returned by UF would be computed for the new random number.

3.4.2. Definition of the subroutine UG

SUBROUTINE UG(n, x, g)

DIMENSION x(n), g(n)

- *reading initial data, in case of water resource problem from file JD.DAT. This reading should occur only during the very first call of the subroutine UG*

- *call to random number generator which supplies new random number*
- *computation of the value of the gradient or subgradient of the function $f(x, \omega)$ and storing it in the array $g(n)$*

  RETURN

  END

This subroutine is optional and if not present then finite differences or random search should be used for the computation of the step direction ξ^s (see section 4.5). It has as its input array $x(n)$ with the value of the current point and should return the value of gradient $f_x(x^s, \omega^s)$ in the array $g(n)$. With each call new random number should be generated.

4. HOW TO RUN THE PROGRAM

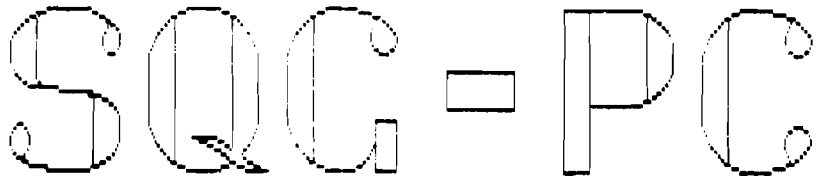
In this section the performance of the program SQG-PC will be described on the example of the water resources problem defined in the Appendix. It will be assumed either a special directory on the hard disk was created and everything from this Example diskette was copied to this directory, which was made default directory, or the copy of the Example diskette without write protection tab was inserted in the floppy disk drive and this drive was made default.

4.1. Starting

Start by executing command JD. The executable program JD.EXE will be loaded in the RAM memory and after a while the screen depicted on the Figure 1 will appear. Press any key and on the next screen will be the short description of the problem (Figure 2). Press once more any key and MAIN MENU screen will appear (Figure 3).

4.2. Main Menu

At this point you define general information about the problem, select between interactive and automatic mode of execution, select the type of algorithm, defined by the type of stepsize and step direction and in the interactive mode decide whether it is time to stop. The screen composition is typical of the other program menus. On the upper part of the screen there is a header with general information about the menu, separated from the rest of the screen by horizontal line. On the lower part of the screen there are tips what to



by Alexei Gaivoronski System and Decision Sciences / IIASA Austria
1987 V.Glushkov Institute of Cybernetics, Kiev USSR

press any key to continue

FIGURE 1

This program solves stochastic programming problem of expectation type

$$\min_{x} \; E_{w} \; f(x,w)$$

by Stochastic QuasiGradient methods, where x belongs to the convex set X defined by linear constraints. User must provide description of the function f(x,w) written in FORTRAN (see manual). For theoretical background on these techniques see Yu.Ermoliev, Stochastics, 9 (1983).

press any key to continue

FIGURE 2

do next, also separated by horizontal line. The menu itself is situated in the central and left part of the screen and is surrounded by double lines. It consists of several entries and cursor, which you can move between items. The cursor would move one item up or down if you press upper or lower arrow key on the keyboard. You can move cursor directly to this item by pressing the key with the first letter of the desired item. At the same time information relevant to the current item is displayed on the central and right part of the screen. The purpose of all this moving is to select items for further execution because each item means some action. In order to select some item move cursor to it and press ENTER key, the selected item will be highlighted. You can select more than one item, in fact an arbitrary number of them. You can deselect item exactly in the same way: move cursor to already selected and therefore highlighted item and press the ENTER key, the highlighted

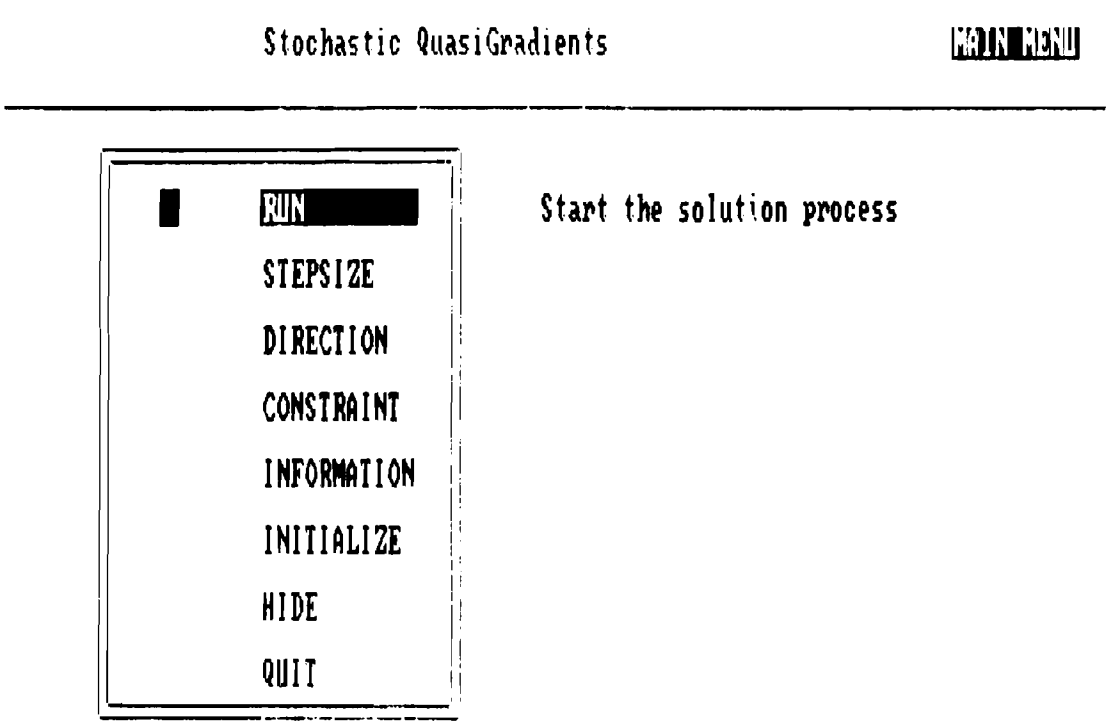


FIGURE 3

item would become normal, which means that it is deselected. After you finish selection process hit Escape key, which would start the process of execution of the highlighted items. If at any time the wrong key is pressed the computer would beep.

Now let us have a look at the MAIN MENU. It appears with the item RUN highlighted. If you will not select anything else the program will start executing in INTERACTIVE mode after you press the *Esc* key. Selection of different items enables you to do the following:

– Choose between AUTOMATIC and INTERACTIVE mode of execution. In AUTOMATIC mode the program would display STOP menu in which you would select stopping criterion options and parameters, clear the screen and proceed silently with optimization process until stopping criterion will be satisfied and terminate. No information on the process evolution will be displayed. The AUTOMATIC mode is described in more detail in the section 4.11. The purpose of AUTOMATIC mode is to relieve user from the process control when the function computation is slow and the whole process can take

many minutes. Another reason to use it arises when the number of similar problems is solved and the best solution algorithm has already been identified during the solution of the typical problem from this set in the INTERACTIVE mode and the algorithm parameters have been tuned already. It is possible to switch from INTERACTIVE to AUTOMATIC mode during solution of the same problem, this is reasonable when the initial behavior of the process promises eventual solution, but the total performance time would be too long.

In the INTERACTIVE mode the program provides user with possibilities to choose algorithm, tune algorithm parameters and monitor the process behavior extensively both numerically and graphically. It is advisable to use the INTERACTIVE mode when solving for the first time the new problem, which differs significantly from the problems solved previously. All the rest of the section 4 except section 4.11 deals with INTERACTIVE mode.

To choose AUTOMATIC mode select option HIDE, if this option is not selected the program would proceed in INTERACTIVE mode.

– Provide initial general information about the problem. This should be done each time the new problem is solved, which differs from the previous in the dimension of decision variables and the structure of constraints.

For providing general information about the problem choose INITIALIZE option, it is described in detail in the section 4.3.

For setting the type of constraints choose CONSTRAINT option, it is described in detail in the section 4.6.

– Choose solution algorithm, that is choose the way of defining stepsize and step direction from (1).

To define the way of stepsize selection choose STEPSIZE option, it is described in detail in the section 4.4.

To define the way of direction selection choose DIRECTION option, it is described in detail in the section 4.5

– Choose the way how certain information about process is processed, more specifically how estimates of the current function value and the gradient norm are defined. In INTERACTIVE mode these estimates could be displayed on the screen and used by user to make decision on the process control, they are also used in some stepsize selection options. In AUTOMATIC mode these estimates are used for choosing stepsize adaptively and for the stopping criteria.

To define the way of estimates selection choose INFORMATION option, it is described in more detail in the section 4.7.

- Terminate the program execution in INTERACTIVE mode. To do this use QUIT option, described in more detail in the section 4.10.

Some general remarks on the MAIN MENU:

- It is possible to select arbitrary subset of options except empty subset.
- RUN option has the lowest priority, it is executed after all other options
- QUIT option has the highest priority.

The example of options selection is shown in Figure 4.

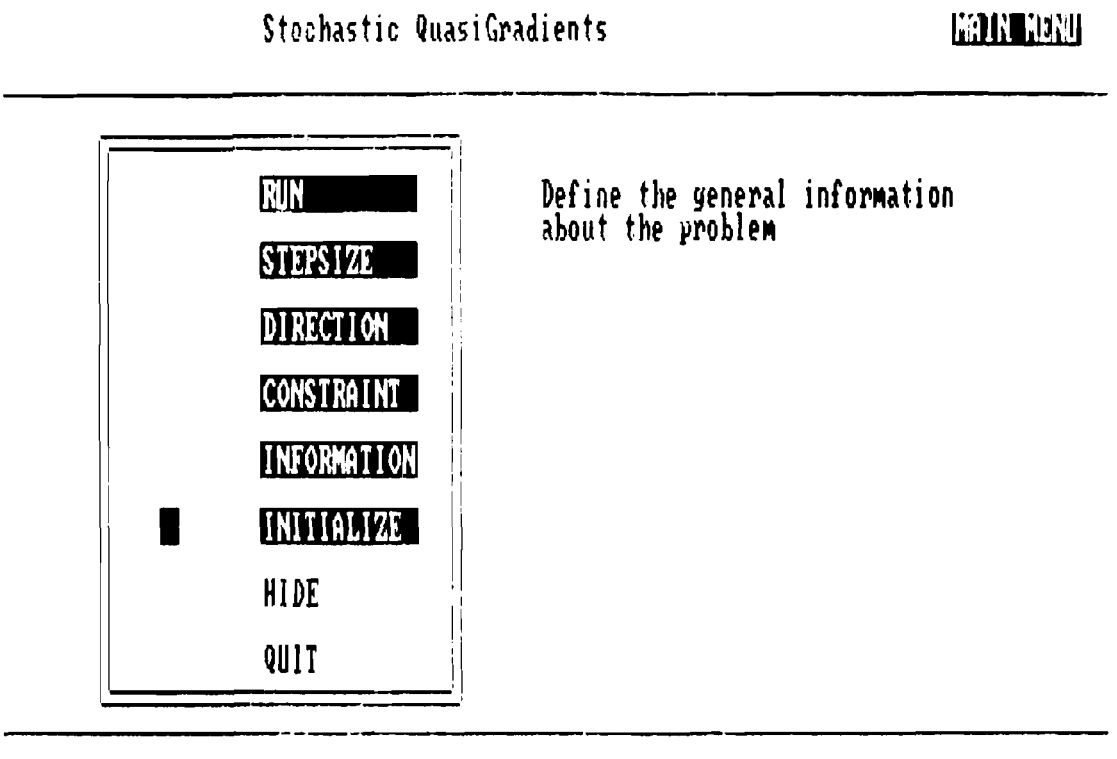


FIGURE 4

4.3. Providing initial general information about the problem

This is done each time the new problem is solved, to do this choose INITIALIZE option from the MAIN MENU. After exiting from the MAIN MENU with the*Escape* key SETUP MENU will appear (Figure 5).

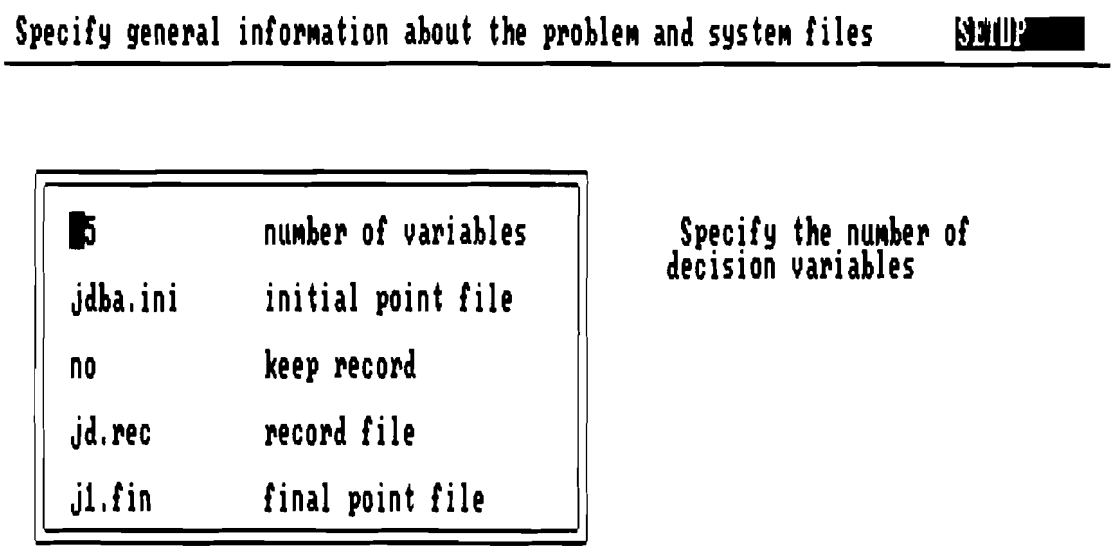


FIGURE 5

This menu gives the first example of the second type of menu, used in SQG-PC. Entries of this menu define not actions, but the problem and algorithm parameters. There are two columns within region, the column to the right shows the names of parameters and the column to the left shows corresponding values. Parameters can have integer, real and string values. Real values can consist of sign, decimal point and digits, integer values consist of sign and digits. There is cursor in the values column, which can be moved with upper, lower, left and right arrow keys. If after pressing left or right arrow key the computer would beep it means that position to the left or right from the current is illegal under present value of parameter.

Purpose of the cursor movement in this menu is to define and change the values of parameters. To facilitate this the simple editor is built in the program. You can change or delete the character on the current cursor position, or put the new character if the current cursor position is blank. In the example the cursor in the SETUP MENU is positioned on the first line in front of the digit 5. Currently this position is blank and reserved for sign. You can leave it blank, put plus or minus sign in it, but if you try to fill digit or letter in it the computer would beep. For the different problem you would probably need to change the problem dimension and to do this you have to move to the right by pressing right arrow key and type desired number. After you type the first digit the cursor would move automatically to the next position to the right. You can delete wrong characters by placing the cursor on it and pressing the *Del* key. The editor distinguishes between integer, real and string parameters. For instance if you try to put decimal point in the integer, letter in numerical parameter or two decimal points in real parameter the computer would beep. Many things in this version of SQG-PC are protected, but still try not to abuse the system like putting negative dimension.

Parameters to be set in SETUP MENU:

- *number of variables* — this should be nonnegative integer which equals the number of decision variables of the problem, not exceeding 1000.
- *initial point file* — the name of the DOS file which contains the starting point for the iteration procedure. Can be anything permitted in DOS. The file should contain the sequence of real numbers separated by blanks. In the example this file has the name JDBA.INI and has the following contents:

  500 40 120 44 25
- *keep record* — two values are permitted: *yes* and *no*. In case of *yes* everything shown on the screen would also be recorded in the specified file, in case of *no* nothing is kept.
- *record file* — This is the DOS name of the file where to keep record. In case of *no* value of *keep record* some dummy name should be provided
- *final point file* — The DOS name of the file where the final point reached by the program will be stored. In case you want to continue execution from the last final point choose this file for the file with initial point the next time you run the program.

4.4. The stepsize selection

This should be done in order to change the algorithm and to do this choose STEPSIZE option from the MAIN MENU. Then STEPSIZE menu would appear on the screen in due course. This menu is of the first type and selection from it is made by means of highlighting as described at the beginning of the section 4.2.

When it appears on the screen some of the options are highlighted already (see Figure 6). These options are either ones selected by user in this or previous run or default options supplied with the system. Some of the options are incompatible. If user still selects them the menu will disappear for a while and then reappear again.

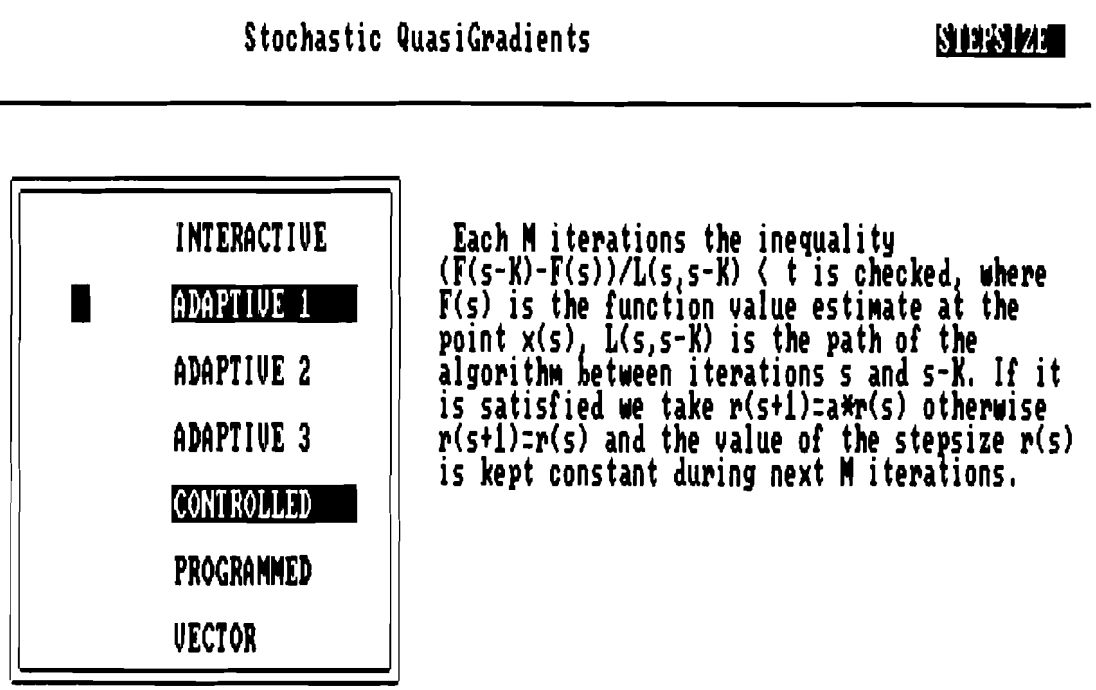


FIGURE 6

MENU OPTIONS:

- INTERACTIVE — This is the only "true" interactive option, which can not be used in AUTOMATIC mode. It keeps stepsize constant until user decides to change it, and to assist user decision various characteristics of the process are available which will be discussed later.

All other options are intended for AUTOMATIC mode use, but can be used also in INTERACTIVE mode, and in this case INTERACTIVE mode is used for the tuning of stepsize selection parameters.

- ADAPTIVE 1

This and subsequent two options are adaptive options. Adaptive means that the program gathers certain information about process behavior, processes it and changes the stepsize ρ_s accordingly. Crucial role in this type of stepsize rules play estimates $F(s)$ of the current value of the objective function $F(x^s)$ or the norm of its gradient. The simplest estimate of the objective function value could be the following:

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

This is very crude estimate, which can not be otherwise due to the fact that it uses only one function observation per iteration. However under fairly general assumptions it converges asymptotically to the true value. What is more important this estimate proved to be quite sufficient for the use in the adaptive stepsize rule. User can choose between this and other kinds of estimates in the INFORMATION MENU (see section 4.7).

Generally adaptive options work as follows. The stepsize ρ_s is kept constant and each iteration the so-called algorithm performance functional $W(s)$ is computed. This functional utilizes estimates mentioned above and is constructed in such a way, roughly speaking, that in the case of regular progress of the process towards minimum it has higher values compared with the case when "chaotic" behavior occurs. For the process with constant stepsize ρ the following pattern characteristic. If we start far from the optimum then in spite of random effects the behavior of the process would be comparatively regular, algorithm progressing more or less systematically towards minimum. Finally the process arrives in the vicinity of the minima and starts to oscillate chaotically, the size of this vicinity depends on the value of the constant stepsize. Adaptive options try to detect this moment with the help of the algorithm performance functional $W(s)$ and then divide stepsize, or make the value of the stepsize proportional to the value of

the performance functional. In the ADAPTIVE 1 option the algorithm performance functional is the difference of the estimate of the objective function in the current iteration and fixed number of iterations before divided by the length of the path traveled by the process during these iterations:

$$W(s) = \frac{F(s-K) - F(s)}{\sum_{i=s-K}^{s-1} \|x^{i+1} - x^i\|}$$

Each M iterations the inequality $W(s) > \alpha$ is checked and if fulfilled then the value of the stepsize ρ_s does not change. Otherwise $\rho_{s+1} = \beta\rho_s$ is taken, where $1 > \beta > 0$ and the process continues with the new value ρ_{s+1} of the stepsize. This rule requires values of the following parameters:

- *memory size* — the number K from the definition of $W(s)$, should not exceed 50, the reasonable choice is 20.
- *change frequency* — number M of iterations to pass before attempting to change the stepsize. Good choice is 20.
- *bound level* — threshold α which triggers stepsize diminishing, advisable to have it zero or small positive
- *initial stepsize* — initial value of stepsize ρ_0, should be of the order of one tenth of the admissible region size
- *change multiplier* — Value β by which the current stepsize is multiplied when necessity to change stepsize is detected. Reasonable values are between 0.5 and 0.9

- ADAPTIVE 2 — In this case algorithm performance functional $W(s)$ equals to the average of the previous stepsize directions. The way to specify the average is defined in the INFORMATION MENU. Each M iterations the new value of the stepsize ρ_s is set: $\rho_s = \beta W(s)$. If this latest value exceeds $\rho_{\max}$ then $\rho_s = \rho_{\max}$ is taken and this

value is kept constant during M subsequent iterations.

Required parameters:

- ***initial stepsize*** the same as in ADAPTIVE 1
- ***gradient multiplier*** multiplier β for obtaining the stepsize from the value $W(s)$
- ***maximal stepsize*** upper bound $\rho_{\max}$ on allowed stepsize
- ***change frequency*** number M of iterations to pass before attempting to change stepsize. Good choice is 20. In current implementation you can change this parameter only from ADPTIVE 1 option, this is inconvenient and will be changed.

- ADAPTIVE 3 This is the combination of the ADAPTIVE 1 and ADAPTIVE 2 Each M iterations the new value of the stepsize ρ_s is computed according to ADAPTIVE 1 and simultaneously the value $W(s)$ from ADAPTIVE 2 is computed. If $\rho_s > \beta_1 W(s)$ then $\rho_s = \beta_1 W(s)$, in the case $\rho_s < \beta_2 W(s)$ then $\rho_s = \beta_2 W(s)$ and otherwise ρ_s is taken as in ADAPTIVE 1

Required parameters:

- ***upper bound mult.*** this is β_2 from the definition of the upper bound : $\rho_s = \beta_2 W(s)$
- ***lower bound mult.*** this is β_1 from the definition of the lower bound: $\rho_s = \beta_1 W(s)$.

all the parameters from ADAPTIVE 1 option

- CONTROLLED This option supplements ADAPTIVE options and never used alone. The purpose of it is to assure convergence of ADAPTIVE options. By themselves ADAPTIVE options are not theoretically convergent, although nonconvergence is quite rare in experiments. Nevertheless CONTROLLED option is provided to assure convergence. In this option two additional sequences of positive numbers are provided: $\rho_u = \alpha_2/s$ and $\rho_l = \alpha_1/s$, where $0 < \alpha_1 < \alpha_2$. In case the stepsize is chosen according to one of these sequences the process converges with probability 1 (see section 2). If one of the ADAPTIVE options is selected simultaneously with

CONTROLLED option then preliminary value ρ_{sp} of the stepsize is selected according to the ADAPTIVE option and the final value ρ_s is chosen as follows:

$$\rho_s = \begin{cases} \rho_l & \text{if } \rho_{sp} < \rho_l \\ \rho_u & \text{if } \rho_{sp} > \rho_u \\ \rho_{sp} & \text{otherwise} \end{cases}$$

Thus, ρ_u and ρ_l serve as bounding sequences, which assure convergence of the algorithm.

Required parameters:

- *upper sequence* — the constant α_2 from the definition of the upper bounding sequence α_2/s
- *lower sequence* — the constant α_1 from the definition of the lower bounding sequence α_1/s

plus all the parameters from the selected ADAPTIVE option

- PROGRAMMED — this option features the simplest theoretically convergent sequence (see section 2) and is provided for the reference. In this option the stepsize is selected according to the formula $\rho_s = c_1/(c_2 + s)$ where $c_1 > 0$, $c_2 \geq 0$.

Required parameters:

- *program constant 1* — the constant c_1 from the definition of the stepsize
- *program constant 2* — the constant c_2 from the definition of the stepsize

- VECTOR — so far the stepsize had scalar values. This option provides for the simple vector stepsize, which is the product of the scalar stepsize ρ_s and diagonal matrix $R(s)$. Initially these elements are set to 1 and after M_1 iterations the sums y_i of the quantities $|x_i^j - x_i^{j+1}|$ for $j = 1, \ldots, M_1 - 1$ are computed for all $i = 1, \ldots, n$, where n is the number of decision variables. The values of the diagonal elements of $R(s)$ are taken inversely proportional to the values of y_i and such, that their sum equals n. These values are kept constant for the next M_1 iterations when the new rescaling is performed and so on. This device proved useful in the problems where "fast"

and "slow" variables exist.

Required parameters:

– *scaling frequency* this is the number M_1 iterations after which the scaling is performed

Compatibility considerations:

- INTERACTIVE option is compatible only with VECTOR option
- ADAPTIVE options are compatible with CONTROLLED and VECTOR options which can be chosen simultaneously
- PROGRAMMED option is compatible with VECTOR option
- VECTOR and CONTROLLED options can be chosen only simultaneously with some other option
- If incompatible options are chosen in the most cases STEPSIZE MENU will be repeated

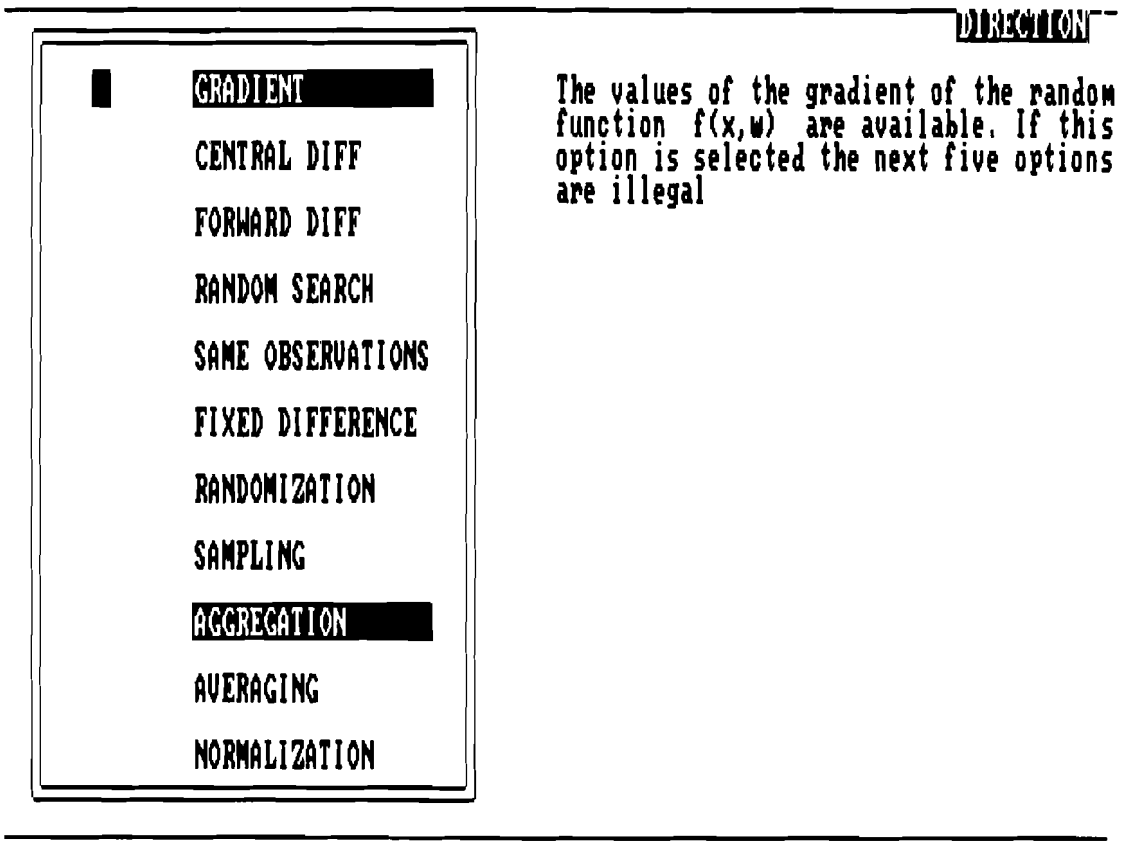


FIGURE 7

4.5. The selection of the step direction

This should be done in order to change algorithm and to do this choose DIRECTION option from the MAIN MENU. Then DIRECTION MENU would appear on the screen in the due course. This menu is of the first type and selection from it is made by means of highlighting as described at the beginning of the section 4.2 (see Figure 7).

When it appears on the screen some of the options are highlighted already. These options are either selected by user in this or previous run or default options supplied with the system. Some of the options are incompatible. If the user still selects them the menu will disappear for a while and then reappear again.

MENU OPTIONS:

First let's describe primary options. They are incompatible with each other and one of them should be selected.

- GRADIENT — the current direction ξ^s will be equal to the gradient or subgradient of the random function $f(x^s, \omega^s)$. In order to use this option user has to provide subroutine for gradient calculation as described in the section 3.4.2

- CENTRAL DIFF — this and other similar options require only subroutine for the random function $f(x^s, \omega^s)$ values as described in the section 3.4.1. In this case

$$\xi^s = \sum_{i=1}^{n} \frac{f(x^s + \delta e_i, \omega_{i1}^s) - f(x^s - \delta e_i, \omega_{i2}^s)}{2\delta} e_i$$

where ω_{i1}^s and ω_{i2}^s are observations of random parameters which can be different or can be the same, this being specified in the SAME OBSERVATIONS option, e_i are unit vectors of the n-dimensional Euclidean space.

Required parameters:

- *finite difference* — the step δ in the finite difference approximation is either fixed or proportional to the value of the stepsize, exact way is defined in the FIXED DIFFERENCE option

- FORWARD DIFF

requires only subroutine for the random function $f(x^s, \omega^s)$ values as described in the section 3.4.1. In this case

$$\xi^s = \sum_{i=1}^{n} \frac{f(x^s + \delta e_i, \omega_{i1}^s) - f(x^s, \omega_{i2}^s)}{\delta} e_i$$

where ω_{i1}^s and ω_{i2}^s are observations of random parameters which can be different or can be the same, this being specified in the SAME OBSERVATIONS option, e_i are unit vectors of the n-dimensional Eucliden space.

Required parameters:

- *finite difference* — the step δ in the finite difference approximation is either fixed or proportional to the value of the stepsize, exact way is defined in the FIXED DIFFERENCE option

- RANDOM SEARCH

requires only subroutine for the random function $f(x^s, \omega^s)$ values as described in the section 3.4.1. This option is useful in the case when the dimension of the problem is considerable and $n + 1$ or $2n$ function evaluations per iteration, required by the finite differences are impossible to afford. Then the number L is chosen and the number of function evaluations per iteration will be $L + 1$. The vectors t_i, $i = 1, \ldots, L$ are chosen each iteration with components independently uniformly distributed on the interval $[0, \delta]$. Then

$$\xi^s = \sum_{i=1}^{L} \frac{f(x^s + t_i, \omega_{i1}^s) - f(x^s, \omega_{i2}^s)}{\|t_i\|} t_i$$

where ω_{i1}^s and ω_{i2}^s are observations of random parameters which can be different or can be the same, this be-

ing specified in the SAME OBSERVATIONS option.

Required parameters:

- *random directions* — the number L of random vectors t_i used to determine ξ^s.
- *finite difference* — the size δ of the interval from which the components of t_i are chosen is either fixed or proportional to the value of the stepsize, exact way is defined in the FIXED DIFFERENCE option

The following options are secondary and are selected together with one of the primary options

- SAME OBSERVATIONS — If this option is not selected then values of the random parameters ω_{i1}^s and ω_{i2}^s are taken all different and independent for all i, if this option is selected then all these values are taken the same and each new iteration only one new random value is generated.
- FIXED DIFFERENCE — This option defines how the value of the step δ in the CENTRAL DIFF or FINITE DIFF options and the size of the random vicinity in the RANDOM SEARCH option are determined. If this option is not selected then on the step number s we have $\delta = \Delta\rho_s$ where Δ is the value of the *finite difference* parameter. If this option is selected then on the step number s the value of the δ equals the value of the parameter *finite difference* itself.
- RANDOMIZATION — This option shifts the point in which direction ξ^s is computed from the current point x^s to the point which components are random variables uniformly independently distributed in the interval $[x_i^s - \Delta_s,\ x_i^s + \Delta_s]$, where x_i^s is the i-th component of the current point x^s and $\Delta_s = r\rho_s/2$. This randomization is useful if the function $F(x)$ has nonregularities like nondifferentiabilities or multiple close extrema. Then

randomization smoothes function behavior.

Required parameters:

- *point neighborhood* the value r which defines the proportionality between the current value of the stepsize and the size of neighborhood from which the random point is chosen

- SAMPLING

Each iteration the K direction vectors $v^{s,i}$ are computed according to the one of the primary options and the final direction ξ^s is obtained as the average of all these directions:

$$\xi^s = \frac{1}{K} \sum_{i=1}^{K} v^{s,i}$$

Required parameters:

- *samples* number K of the independent direction vectors computed at each iteration

- AGGREGATION

In this option the values of the previous step directions are used to form the current step direction, namely the current step direction ξ^s is computed as linear combination of all previous step directions: $\xi^1 = v^1$, $\xi^s = (1 - \alpha)\xi^{s-1} + \alpha^s$, where v^s is computed according to one of the primary options. This technique can be called analogue of the conjugate gradient method of the nonlinear optimization.

Required parameters

- *aggregation* the coefficient α which is used to for the linear combination with the previous direction

- AVERAGING

This is another way to combine current direction with previous ones. The number L_1 is selected and for the particular iteration s let $L(s) = L_1[s/L_1]$ where $[\cdot]$ denotes the integer part. Then the current direction ξ^s is computed as the average of the directions v^s during

previous $s - L(s)$ iterations, where v^s is obtained according to one of the primary options

$$\xi^s = \frac{1}{s - L(s) + 1} \sum_{i=L(s)}^{s} v^i$$

Required parameters:

- *averaging* defines the value of L_1

- NORMALIZATION — This option is useful if the norm of the quasi-gradient ξ^s can vary considerably. Suppose that v^s was obtained according to one of the primary options possibly combined with one of the secondary options mentioned above. Then $\xi^s = v^s / \|v^s\|$

4.6. The selection of the constraints type

This should be done each time the new problem is being solved and to do this choose CONSTRAINT option from the MAIN MENU. Then CONSTRAINT MENU would appear on the screen in the due course. This menu is of the first type and selection from it is made by means of highlighting as described at the beginning of the section 4.2.

When it appears on the screen some of the options are highlighted already. These options are either selected by user in this or previous run or default options supplied with the system (see Figure 8). This menu only defines the type of constraints and user has to describe actual constraints in the special file according to specified rules. This file has the name, specified by user and this name is supplied as *constraints file* parameter, which appears on PARAMETER DEFINITION MENU (see section 4.8). The composition of this file depends on the type of constraints and is described in this section.

MENU OPTIONS:

- NONE — the unconstrained problem is solved and no additional information is required

- BOUNDS — the feasible region is defined by upper and lower bounds on the individual variables. This option is preferable to the NONE option even if actually there are no bounds, but the region with the optimal solution can be identified, however loosely. In this case artificial introduction of bounds can prevent overflow in the case when initial stepsize is incorrectly chosen. The user should put the values of these

bounds in the file with the name accepted by DOS. The contents of this file should consist of the sequence of real numbers separated by blanks, first come all the upper bounds in order of increasing variable index and then all lower bounds. The algorithm would make projection on the bounds.

- EXAMPLE

suppose that we have two decision variables x_1 and x_2 and the feasible region is defined as follows: $0 \leq x_1 \leq 100$, $10 \leq x_2 \leq 200$. Then the constraints file looks as follows:

100 200

0 10

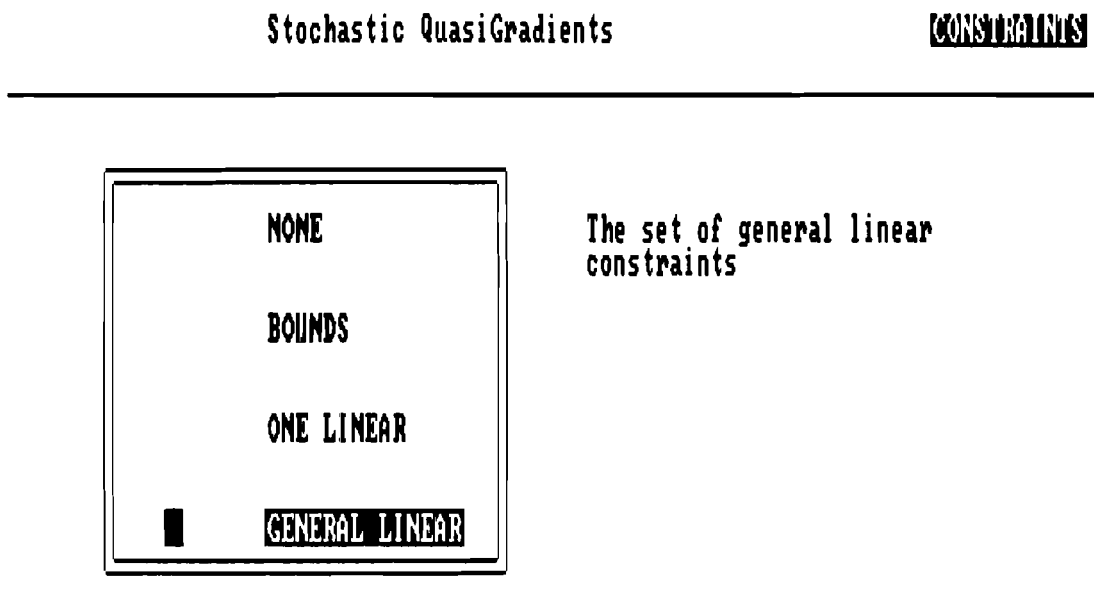


FIGURE 8

Required parameters:

- *constraints file* — any name acceptable by DOS, file with this name should contain constraints information specified above

- ONE LINEAR — the feasible region is defined by one hyperplane and algorithm makes projection on this hyperplane. The constraints file consists of the sequence of real numbers separated by blanks, first come hyperplane coefficients and then the right hand side.

- EXAMPLE — suppose that we have two decision variables x_1 and x_2 and the feasible region is defined as follows: $5x_1 + 10.2x_2 = 200$. Then the constraints file looks as follows:

5 10.2

200

Required parameters:

- *constraints file* — any name acceptable by DOS, file with this name should contain constraints information specified above

- GENERAL LINEAR — the feasible region is specified by the set of general linear constraints and bounds. In this case in order to make precise projection it is necessary to solve quadratic programming problem. This is a too costly thing to do each iteration on the computer like AT compatible. Moreover, the direction ξ^s only in average coincides with the gradient of the objective function $F(x)$ and each individual ξ^s could be very far from actual gradient due to random effects, this questions wisdom of precisely projecting imprecise direction. In this particular version of SQG-PC the exact penalty function approach is chosen instead. Each iteration the candidate y^s for the new point is computed $y^s = x^s - \rho_s \xi^s$ and if the point is feasible with respect to general linear constraints then $x^s = y^s$ is taken. Otherwise the most violated constraint is identified, suppose this is constraint with index j and vector

of coefficients b^j, then we take $x^s = y^s - \gamma c \rho_s b^j \|\xi^s\| / \|b^j\|$. Here γ equals either 1 or -1 depending on the type of constraint and c is the penalty coefficient, which can be changed interactively by the user. If the point x^s defined in this way does not fit within bounds then additional projection on bounds is performed. For sufficiently large, but finite penalty coefficient c all the accumulation points of the sequence x^s generated in this way belong to the feasible region if the stepsize ρ_s tends to zero. This method is of course not competitive in deterministic optimization problems, but experience shows that it is quite reasonable in stochastic environment. For example the water resources problem from the Appendix was solved on the XT compatible with exact penalties and on the VAX 780 with projections, in both cases approximately the same amount of CPU time was required.

The constraints file contains information about bounds and general linear constraints and consists of the sequence of real and integer numbers, separated by blanks. This sequence is composed in exactly the following order:

- upper bounds - sequence of n real numbers, where n is the number of decision variables

- lower bounds - sequence of n real numbers

- number n_1 of constraints, excluding bounds - integer in the current implementation $n_1 \leq 200$ constraints

- number n_2 of nonzero coefficients in constraints - integer in the current implementation $n_2 \leq 1\,000$

- types of constraints - sequence of n_1 integer numbers, these should be 0,1 or 2, zero for equality constraint, 1 for less or equal constraint and 2 for greater or equal constraint.

- numbers of nonzero elements in each constraint - sequence of n_1 integer numbers

- column positions of nonzero elements in the corresponding constraints - sequence of n_2 integer numbers

– nonzero elements – sequence of n_2 real numbers, the order should be the same as in the previous array.

– right hand sides of constraints – sequence of n_1 real numbers.

EXAMPLE: For the water resources problem from the Appendix the constraints file is JD.CON on the Example diskette and looks as follows:

```
500.0  102.319  252.0  252.0  252.0
100.0  38.1  0.0  0.0  0.0
7  23
1  1  1  2  2  2  2
2  3  4  2  3  4  5
2  3  2  3  4  2  3  4  5  1  2  1  2  3 1  2  3  4  1  2  3  4  5
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
156.448  201.866  225.297  512.886  592.872  654.152  720.183
```

Required parameters:

– *constraints file* any name acceptable by DOS, file with this name should contain constraints information specified above

4.7. The selection of the information processing options

This should be done each time the new problem is being solved and the type of the stepsize selection is changed. It is especially important for the ADAPTIVE stepsize options, but also useful for providing on-line information to the user about the process behavior. The most important information which is selected are the estimates of the current value of the objective function and its gradient. To make selection choose INFORMATION option from the MAIN MENU (see Figure 9). Then INFORMATION MENU would appear on the screen in due course. This menu is of the first type and selection from it is made by means of highlighting as described at the beginning of the section 4.2.

When it appears on the screen some of the options are highlighted already. These options are either selected by user in this or previous run or default options supplied with the system.

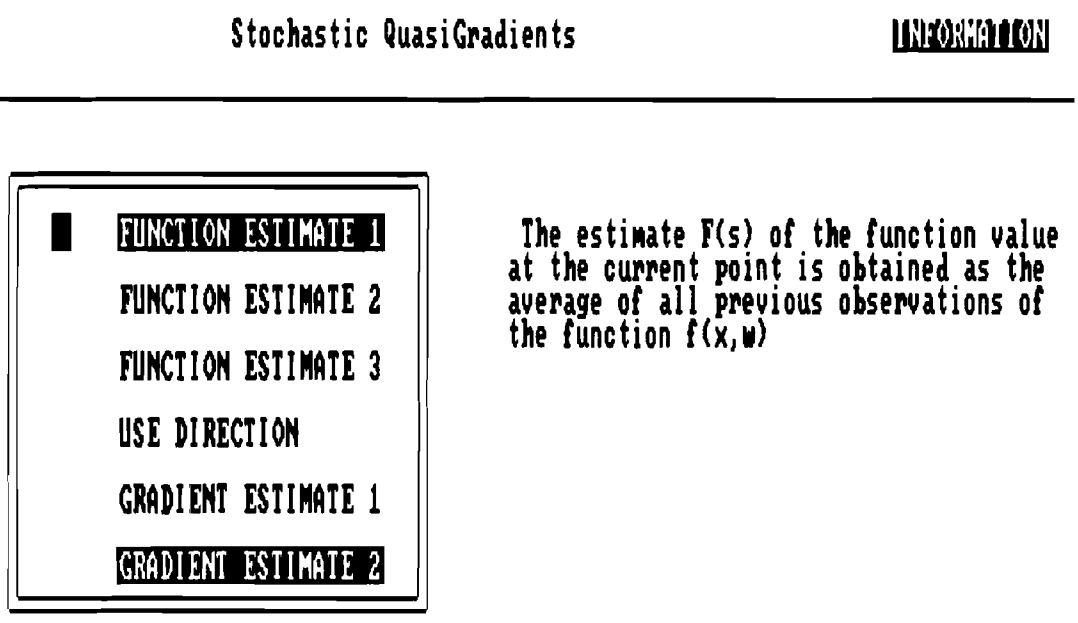


FIGURE 9

MENU OPTIONS:

- FUNCTION ESTIMATE 1

The estimate $F(s)$ of the current value $F(x^s)$ of the objective function is computed as average of the all previous observations of the random function $f(x^i, \omega^i)$:

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

where ω^i are independent observations of the random parameters of the problem. This estimate can use as little as one value of the random function per iteration to form the current estimate of the objective function value. In some stepsize and direction options the program generates more than one observation of the random parameters and compute more than one value of the random function (this is the case for example in DIFF and SAMPLE options of the DIRECTION

MENU). These additional function evaluations can be used in the estimates $F(x)$ too, for more details see USE DIRECTION option of this menu

- FUNCTION ESTIMATE 2

The estimate $F(s)$ of the current value $F(x^s)$ of the objective function is computed as the moving average of all previous observations of the random function $F(x^i, \omega^i)$:

$$F(1) = f(x^1, \omega^1),$$
$$F(s+1) = (1 - \alpha_1)F(s) + \alpha_1 f(x^s, \omega^s)$$

where $0 < \alpha_1 \leq 1$. This estimate less depends on the initial observations, which are made far from solution, but unlike the previous estimate it does not converge asymptotically to the true value of the objective function.

Required parameters:

- *moving average* coefficient α_1

- FUNCTION ESTIMATE 3

This estimate is similar to the FUNCTION ESTIMATE 1 except the average is computed for the last K iterations:

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

Required parameters:

- *memory size* the value K of the "depth" of memory, this is the same parameter as described in ADAPTIVE 1 stepsize selection option

- USE DIRECTION

This is supplementary option to the FUNCTION ESTIMATE options. If this option is not selected then only one observation of the random function per iteration will be used to form the current estimate of the objective function. If this option is selected then additional observations will also be incorporated in the estimate. These additional observations are made for

the estimation of the step direction in the CENTRAL DIFF, FORWARD DIFF, RANDOM SEARCH and SAMPLE options.

- GRADIENT ESTIMATE 1 — The estimate $G(s)$ of the current value $F_x(x^s)$ of the objective function gradient is computed as the average of all previous observations of the step direction ξ^s

$$G(s) = \frac{1}{s} \sum_{i=1}^{s} \xi^i$$

where ω^i are independent observations of the random parameters of the problem.

- GRADIENT ESTIMATE 2 — The estimate $G(s)$ of the current value $F_x(x^s)$ of the objective function gradient is computed as the moving average of all previous step directions ξ^s:

$$G(1) = \xi^1,\ G(s+1) = (1 - \alpha_2)G(s) + \alpha_2\xi^s$$

where $0 < \alpha_2 \leq 1$. This estimate less depends on the initial observations, which are made far from solution, but unlike the previous estimate it does not convergence asymptotically to the true value of the objective function gradient.

Required parameters:

- *gradient estimator* coefficient α_2

This ends the description of the preliminary actions, which consist of problem and algorithm definitions. After pressing the *ESC* key from the last selected MAIN MENU option the optimization process begins.

4.8. Defining the values of the algorithm parameters

This is done at the beginning of the solution process from the PARAMETER DEFINITION MENU which appears after option menus invoked from the MAIN MENU (see Figure 10). This menu is of the second type and features two columns. The column of the parameter values comes first and the column of the parameter names comes second. The menu displays either the default values of the parameters or values defined by the user previously. These values can be changed as described in the general information on the menus of the second type in the beginning of the section 4.3. All parameters relevant to

```
Select stepsize, direction and information parameters
```

4.0	initial stepsize
0.5	change multiplier
20	change frequency
0.1	bound level
100.00	upper sequence
50.0	lower sequence
0.02	aggregation
20	memory size
1.0	gradient estimator
jd.con	constraints file
0.0001	feasibility
5.00	penalty coefficient

```
Specify the initial value for stepsize

Press Escape key when finished with changes
```

FIGURE 10

iteration	method performance / 1e0	step size / 1e0	constraints violat. / 1e0	x(1) / 1e0	x(2) / 1e0	x(3) / 1e0	x(4) / 1e0	x(5) / 1e0
1	0.000	4.000	3.703	499.920	38.100	99.999	23.999	12.999
2	17.094	25.000	45.166	500.000	102.319	252.000	230.555	218.555
3	11.514	25.000	578.133	498.530	38.100	80.443	107.999	95.018
4	2.668	25.000	96.263	496.589	38.100	0.000	0.000	0.000
5	3.146	20.000	185.494	500.000	102.319	192.277	252.000	229.172
6	1.739	16.667	550.470	498.097	38.100	16.696	139.831	83.721

```
Press Escape key to access runtime menu
```

FIGURE 11

the options selected previously and described in the sections 4.4–4.7 will appear in the PARAMETER DEFINITION MENU. It is recommended that user first try default values and only then change the parameter values from the RUNTIME menu described in the next section. To start iterations press *Esc* key.

4.9. Iteration loop

Now the iterations have started and the display screen looks like the Figure 11 with new lines of information adding below. The table is appearing on the screen, which contains information about the process. The first column of the table is always iteration number, contents of other columns can be changed by the user. In Figure 11 the second column contains the value of the algorithm performance function $W(s)$, defined in the section 4.4, the third column contains the value of the stepsize and the third the absolute value of the worst violated constraint. All the subsequent columns contain the values of the decision variables. Note the bottom row from the table header. This row contains things like $/\ 1em$ or $*\ 1em$ where m is positive number. These figures represent scales and mean that displayed numbers in these columns were obtained by division or multiplication of the original numbers by 10^m. The scales can be changed from RUNTIME MENU as described in the section 4.9.1.3.

During the iteration loop the user can change the values of the algorithm parameters, penalty coefficient, composition of the screen, can switch from the numerical to the graphical process information representation, estimate more precisely the value of the objective function $F(x)$ at the current or specified point and jump to arbitrary point from the current one. To do this it is necessary to suspend execution of the iteration loop and access RUNTIME MENU. This can be done after any iteration by pressing the *Esc* key.

4.9.1. Interactive capabilities during iteration process

Press *Esc* key and the screen would appear as on the Figure 12. Two horizontal lines appear on the top of the table. The first one is RUNTIME MENU with 8 options and the second is the short explanation of the option where the cursor is currently situated.

The RUNTIME MENU is very similar to the menu of the first type described at the beginning of the section 4.2. The main difference is that for the cursor movement it is necessary to press left and right arrow keys instead of upper and lower arrow keys. The initial position of the cursor is in the very beginning of the menu line. Initial selection is RESUME option and if nothing else would be selected the program would resume execu-

resume stepsize direction monitor estimate graph constrain quit
proceed with iterations

iteration	method performance / 1e0	step size / 1e0	constraints violat. / 1e0	x(1) / 1e0	x(2) / 1e0	x(3) / 1e0	x(4) / 1e0	x(5) / 1e0
1	0.000	4.000	3.703	499.920	38.100	99.999	23.999	12.999
2	17.094	25.000	45.166	500.000	102.319	252.000	230.555	218.555
3	11.514	25.000	578.133	498.530	38.100	80.443	107.999	95.018
4	2.668	25.000	96.263	496.589	38.100	0.000	0.000	0.000
5	3.146	20.000	185.494	500.000	102.319	192.277	252.000	229.172
6	1.739	16.667	550.470	498.097	38.100	16.696	139.831	83.721
7	1.091	14.286	53.051	496.213	38.100	0.000	29.204	0.000
8	1.420	12.500	156.665	500.000	102.319	207.788	233.168	179.991
9	1.598	11.111	497.970	498.153	38.100	96.731	118.779	44.720
10	1.181	10.000	73.033	496.323	38.100	0.000	17.873	0.000
11	1.296	9.091	167.887	500.000	102.319	191.926	188.999	154.717

Select item by arrow and Return keys, proceed further by Escape key

FIGURE 12

Select stepsize, direction and information parameters

4.0	initial stepsize
0.5	change multiplier
20	change frequency
0.1	bound level
100.00	upper sequence
50.0	lower sequence

Specify the initial value for stepsize

Press Escape key when finished with changes

FIGURE 13

tion of the iteration loop after the *Esc* key is pressed. In order to select other options it is necessary to place the cursor in front of the option and press the ENTER key, the selected option will be highlighted. To select option proceed in a similar fashion. Several options can be selected simultaneously and in this menu there are no incompatible options. After selection is finished press the *Esc* key and one or several of the parameter definition menus will appear successively. After the new values of parameters will be entered the execution of the iteration loop will be resumed.

4.9.1.1. Changing stepsize parameters

In order to do this choose STEPSIZE option from the RUNTIME MENU. The ALGORITHM PARAMETER MENU will appear which is menu of the second type and contains all parameters relevant to the stepsize selection (see Figure 13). Any of them can be changed now as described in the beginning of the section 4.3. In case of only one relevant parameter then instead of the whole menu only one line on the place of the RUNTIME MENU will appear and the current table with the process information will be retained. The manipulation with this single line is exactly the same as with the whole menu. When the changes are finished press the *Esc* key to exit this menu and to go to the next selected menu or to resume iterations loop.

4.9.1.2. Changing direction parameters

In order to do this choose DIRECTION option from the RUNTIME MENU. The ALGORITHM PARAMETER MENU will appear which is menu of the second type and contains all parameters relevant to the stepsize selection (see Figure 14). Any of them can be changed now as described in the beginning of the section 4.3. In case of only one relevant parameter, as in this particular example then instead of the whole menu only one line on the place of the RUNTIME MENU will appear and the current table with the process information will be retained. The manipulation with this single line is exactly the same as with the whole menu. When the changes are finished press the *Esc* key to exit this menu and to go to the next selected menu or to resume iterations loop.

If both STEPSIZE and DIRECTION options from the RUNTIME MENU are selected then only one ALGORITHM PARAMETER MENU will appear, which will combine relevant parameters for both stepsize and step direction options.

0.02 aggregation

iteration	method performance / 1e0	step size / 1e0	constraints violat. / 1e0	x(1) / 1e0	x(2) / 1e0	x(3) / 1e0	x(4) / 1e0	x(5) / 1e0
38	0.248	2.632	90.308	500.000	102.319	146.270	107.691	91.573
39	0.201	2.564	222.556	498.602	38.100	86.774	47.368	43.035
40	0.158	2.500	6.304	500.000	102.319	165.118	129.923	131.850
41	0.310	2.439	303.913	498.626	38.100	105.136	73.967	81.879
42	0.160	2.381	73.785	497.265	38.100	47.747	20.430	34.068
43	0.124	2.326	82.574	500.000	102.319	122.522	103.543	118.011
44	0.400	2.273	221.099	498.662	38.100	63.296	52.302	67.565
45	0.363	2.222	1.793	500.000	102.319	146.574	147.677	85.409
46	0.432	2.174	256.683	498.684	38.100	84.814	97.513	31.714
47	0.274	2.128	26.844	497.380	38.100	25.571	49.393	0.000
48	0.242	2.083	109.740	500.000	102.319	98.244	132.739	84.263
49	0.371	2.041	192.269	498.718	38.100	37.159	81.901	34.305
50	0.341	2.000	30.001	500.000	100.614	111.346	161.928	115.178

FIGURE 14

Select screen parameters

250	how often to stop
1	display frequency
1	displayed variable 1
2	displayed variable 2
3	displayed variable 3
4	displayed variable 4
5	displayed variable 5
0	multiply column 1
0	multiply column 2
0	multiply column 3
0	multiply column 4
0	multiply column 5
0	multiply column 6
0	multiply column 7
0	multiply column 8
no	screen composition

Specify the number k1 such that execution will be stopped after each k1 displayed iterations and runtime menu will be shown. You can get this menu any time by pressing Escape key too

Press Escape key when finished with changes

FIGURE 15

4.9.1.3. Changing type of information displayed on the screen

In order to do this choose MONITOR option from the RUNTIME MENU. The PRIMARY SCREEN MENU will appear which is menu of the second type and contains all parameters relevant to the stepsize selection (see Figure 15). Any of them can be changed now as described in the beginning of the section 4.3.

PRIMARY SCREEN MENU includes the following parameters:

- *how often to stop* — the program will suspend execution of the iterations loop after the number of iterations, defined by this parameter and display RUNTIME MENU
- *display frequency* — integer number m_1, the program will display on the screen only information about each m_1-th iteration
- *displayed variable 1* — the number of the first displayed decision variable, this is important because it could be up to 1 000 decision variables and all of them can not be shown on the screen simultaneously. There is the same number of *displayed variable* parameters as in the current information table header, this number can be changed in the SECONDARY SCREEN MENU
- *multiply column 1* — this is the scale l_1 of the first column as described in the section 4.9. It can be positive or negative integer. If the value of l_1 is positive then original number is divided by 10^{l_1} and then displayed in the corresponding column, if it is negative then the original number is multiplied by 10^{l_1} before being displayed. All other *multiply column* parameters has the same meaning.
- *screen composition* — This is access parameter to the SECONDARY SCREEN MENU and can take two values: *yes* and *no*. In the case of *no* the program will proceed to other menus or to execution of the iterations loop after the *Esc* key is pressed, in the case of *yes* the SECONDARY SCREEN MENU will be accessed. This menu enables to choose what process characteristics will appear on the screen and how many decision variables will be displayed.

SECONDARY SCREEN MENU is also of the second type and how to change parameters in this menu is described in the section 4.3 (see Figure 16). It defines the screen composition during the INTERACTIVE iterations loop. It contains two groups of parameters. The first group defines the process characteristics which can be displayed during iterations. These parameters can have only two values: *yes* and *no*, *yes* means that corresponding parameter will appear on the screen and *no* means that it will not be displayed. This group consists of the following parameters:

- *performance measure* — this is algorithm performance functional $W(s)$ described in the ADAPTIVE 1 option part of the section 4.4
- *function estimate* — the estimate $F(s)$ of the current value $F(x^s)$ of the objective function. The estimate selected in the INFORMATION MENU will be displayed
- *$f(x, \omega)$ observation* — one of the observations of the random function $f(x^s, \omega^s)$ performed on the iteration number s. This option is very useful when the problem is deterministic, then the value of the random function coincides with the value of the objective function
- *stepsize value* — the value of the current stepsize ρ_s
- *gradient norm* — the norm of the average of the step directions ξ^s. The average selected in the INFORMATION MENU will be displayed.
- *constraint violation* — the absolute value of the most violated constraint

The second group of parameters of the SECONDARY SCREEN MENU are the indexes of the displayed decision variables. The total number of these parameters is 8 minus the number of the parameters in the first group with *yes* values. The displayed variable parameters are the same as in the PRIMARY SCREEN MENU and are included in the SECONDARY SCREEN MENU for conveniency.

When the changes are finished press *Esc* key to exit this menu and to go to the next selected menu or to resume iterations loop.

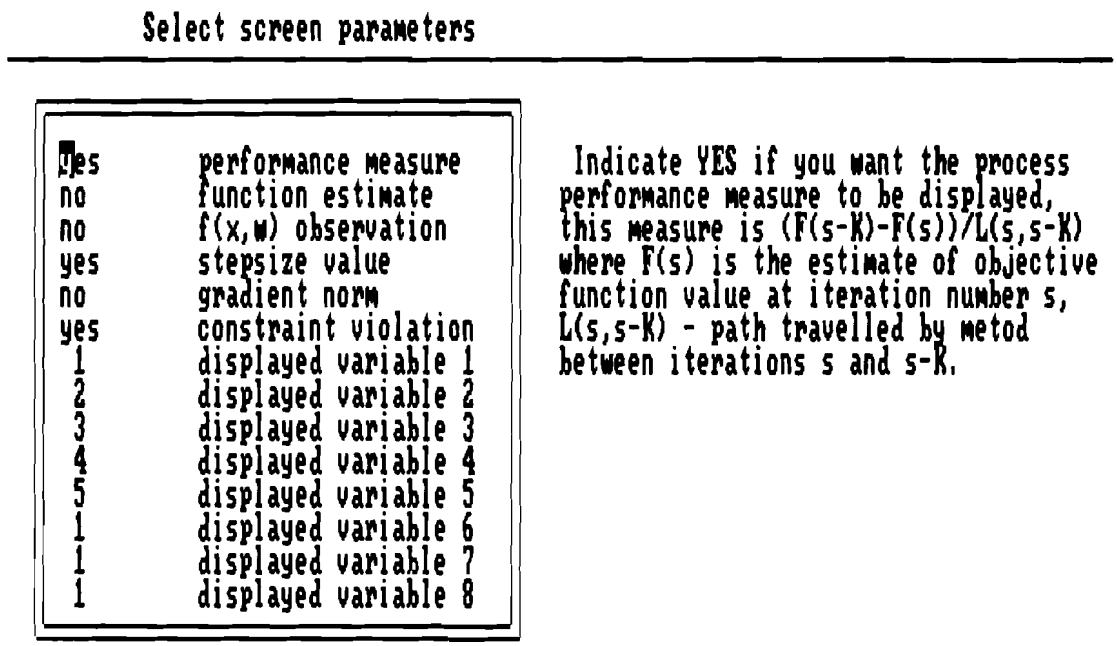


FIGURE 16

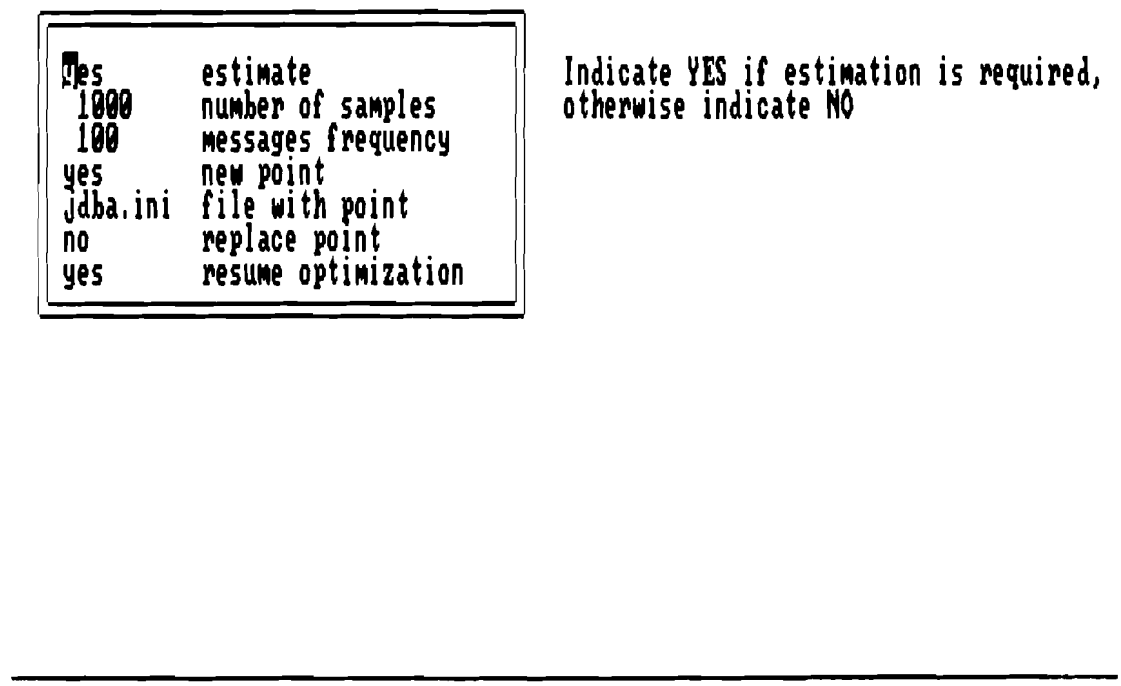


FIGURE 17

4.9.1.4. How to make estimation and/or continue from the new point

In order to do this chose ESTIMATE option from the RUNTIME MENU. The ESTIMATION MENU will appear which is menu of the second type and contains parameters relevant to the estimation and new point selection (see Figure 17). Any of them can be changed now as described in the beginning of the section 4.3. This menu allows to suspend iteration process and estimate the value of the objective function $F(x)$ at the current or any specified point and then resume optimization process from the current or arbitrary specified point. It is possible to change the current point without estimation. Estimation capability is useful when random function evaluation does not take too much time and couple of estimations per problem solution are affordable. In this case estimation helps to understand the pattern of approaching the optimal solution.

The ESTIMATION MENU consists of the following parameters:

- *estimate* — can take values *yes* or *no*, in the case of *yes* estimation will be performed, in the case of *no* there will be no estimation. The *no* value of this parameter is used for the point changing without estimation

- *number of samples* — the total number N of independent observations of random parameters ω^i which will be used to make the estimate of the objective function. The final estimate $F(x, s)$ will be computed as follows:

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

- *messages frequency* — integer number N_1, it defines the frequency with which the intermediate estimates of the objective function value will be displayed

- *new point* — can take values *yes* or *no*. In the case of *no* the estimation will be performed at the current point, in the case of *yes* the estimation will be performed at the point specified by the user.

- *file with point* — this parameter makes sense only if the value of the new point parameter is *yes*. Then it should be the DOS name of the file where the user has put the new point, the contents of this file should be the string of the n real numbers separated by blanks, where n is the dimension of the space of the deci-

sion variables. If the value of this parameter is *screen* then the program will take the input from the screen.

– *replace point* — this parameter can have two values: *yes* and *no*. In the case of *no* the optimization will be finally resumed from the current point, otherwise it will be resumed from the new point.

– *resume optimization* — the value should be *yes* or *no*. In the case of *no* after the end of the estimation the ESTIMATION MENU will be displayed again, thus allowing further estimation, perhaps at the new point. In the case of *yes* no further estimation at this moment is possible and other selected RUNTIME MENU options will be processed and optimization resumed.

When the changes are finished press the *Esc* key to exit this menu and to begin estimation process or resume optimization from current or specified point. In the case of estimation the picture similar to Figure 18 will appear (on this figure estimation was conducted at the new point and then the point is displayed). After estimation process is finished press any key to continue the estimation or to resume optimization as defined in the *resume optimization* parameter.

```
new point
    500.00000000      40.00000000     120.00000000      44.00000000      25.0000
0000
Press Escape key to terminate estimation
       100     774.52770996
       200     772.95947266
       300     778.66210937
       400     764.22045898
       500     784.40319824
       600     800.90301514
       700     811.37976074
       800     814.81536865
       900     825.91162109
      1000     828.72619629
final estimate
    828.72619629
Press any key to continue
```

FIGURE 18

4.9.1.5. How to display process information graphically

Any numerical information which is currently being displayed on the screen can be alternatively displayed graphically. In order to do this choose GRAPH option from the RUNTIME MENU. The RUNTIME MENU will disappear, but the process information table will remain. The two first lines with RUNTIME MENU will disappear and another two lines will appear on their place. These two lines constitute GRAPH SELECTION MENU, which is similar in structure to the RUNTIME menu (see Figure 19). Selection from this menu is made by means of highlighting as described in the section 4.9.1.

Iters 1 2 3 4 5 6 7 8
Make the number of iterations the x variable of the graph

iteration	method performance / 1e0	step size / 1e0	const-raints violat. / 1e0	x(1) / 1e0	x(2) / 1e0	x(3) / 1e0	x(4) / 1e0	x(5) / 1e0
1	0.000	4.000	3.703	499.920	38.100	99.999	23.999	12.999
2	17.094	25.000	45.166	500.000	102.319	252.000	230.555	218.555
3	11.514	25.000	578.133	498.530	38.100	80.443	107.999	95.018
4	2.668	25.000	96.263	496.589	38.100	0.000	0.000	0.000
5	3.146	20.000	185.494	500.000	102.319	192.277	252.000	229.172
6	1.739	16.667	550.470	498.097	38.100	16.696	139.831	83.721
7	1.091	14.286	53.051	496.213	38.100	0.000	29.204	0.000
8	1.420	12.500	156.665	500.000	102.319	207.788	233.168	179.991
9	1.598	11.111	497.970	498.153	38.100	96.731	118.779	44.720
10	1.181	10.000	73.033	496.323	38.100	0.000	17.873	0.000
11	1.296	9.091	167.887	500.000	102.319	191.926	188.999	154.717

Select item by arrow and Return keys, proceed further by Escape or F2 key

FIGURE 19

The GRAPH SELECTION MENU consists of 9 entries, first entry corresponds to the column with iterations number and remaining 8 enumerate columns with the process information. It is necessary to select at least two entries from this menu. The first entry will define the independent variable to constitute the horizontal axis of the graph, while the remaining entries would be dependent variables depicted along the vertical axis. There could be more than one dependent variables, but it is not recommended to choose more than four of them.

After the selection of depicted variables is performed it is necessary to leave GRAPH SELECTION MENU by pressing the *Esc* or F2 key. If the *Esc* key is pressed then the program will draw the graph and continue the iterations loop with information about the process continued to be displayed graphically.

The example of the program performance in the graph mode can be seen in the Figure 20. This is the example of one of the runs of the water resources example from the Example diskette. Graph on the Figure 20 has the number of iterations on the horizontal axis and x_2 on the vertical axis as dependent variable. This picture shows the typical behavior of the stochastic optimization algorithm. The stepsize here is chosen according to ADAPTIVE 1 and CONTROLLED options with FUNCTION ESTIMATE 1 information option and the step direction is chosen according to the GRADIENT and AGGREGATION options. At first the values of the variables oscillate heavily between approximately 38 and 112 and then the program accumulates information and begins to divide stepsize which lead to convergence with oscillations between 38 and 40. This is the most what can be achieved with the algorithm of this type, after iteration number 200 convergence slows down and oscillations become almost stationary diminishing with the rate proportional to the $1/\sqrt{s}$ which is almost invisible. The picture of these is shown on Figure 21, where scales of the graph were changed automatically. This stationary process is shown also on Figure 22, where the axis are different: horizontal axis is x_1 and vertical axis remained x_2. In Figure 23 constraints violation is shown, which again oscillates considerably between 0 and 550, but then converges almost to the constant zero.

Let us return to Figure 20. The first line contains suggestions for the process control. As in the numeric mode pressing the *Esc* key would suspend the process and display RUNTIME MENU with all the options described above. The F1 key will resume numerical display of the process information and F3 key will clear the screen and start drawing anew. This is especially useful when horizontal axis is not the number of iterations because in this case the graphing area becomes congested quite soon and redrawing is needed frequently. The F2 key will give GRAPH SETTINGS MENU which is needed for the changing of the graph pattern and will be discussed later. The second and the third line are devoted to the numerical information. The second line is the simplified header of the numeric information table and the third line displays the current numerical values of the process parameters as selected in the SECONDARY SCREEN MENU (see section 4.9.1.3). The axes are marked with the integer number of the decimal digits, for instance you will never find markings like 0.3456. On the fifth line from above in the upper left corner of the graphing area the scaling parameter * 100 is present. It means that the actual number which marks the vertical axis should be multiplied by 100, for instance the

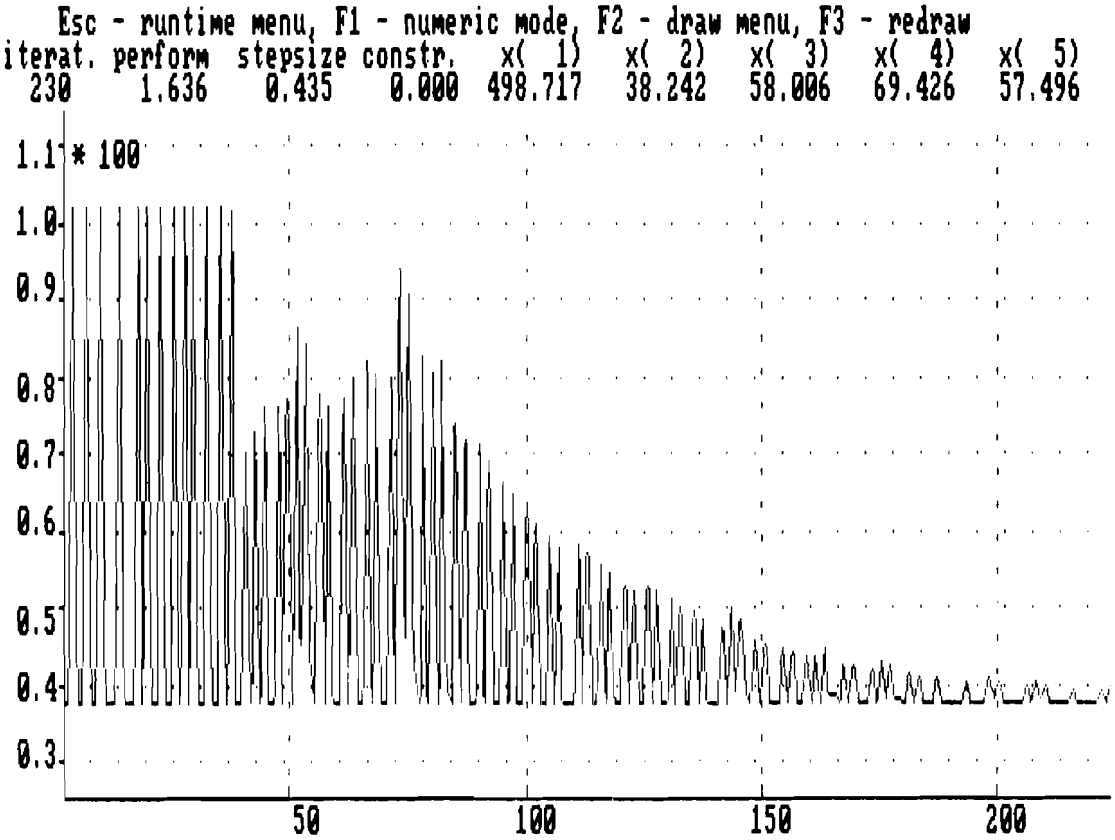


FIGURE 20

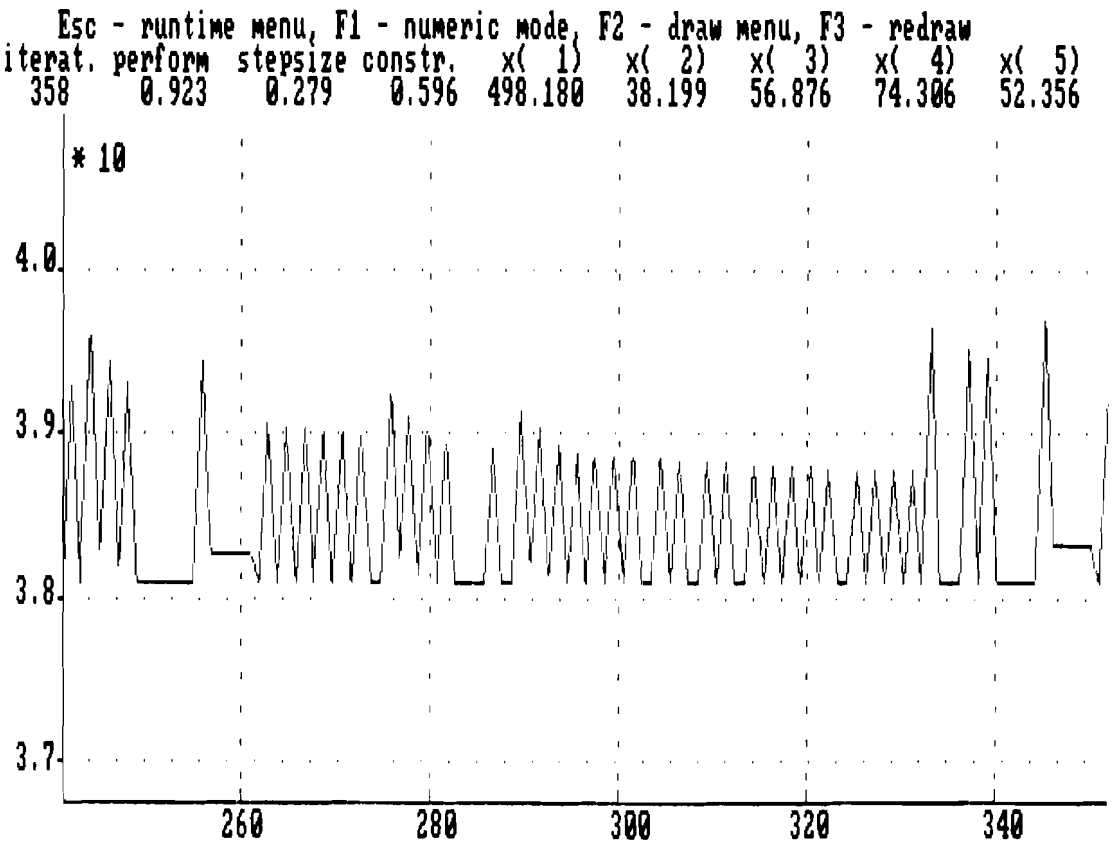


FIGURE 21

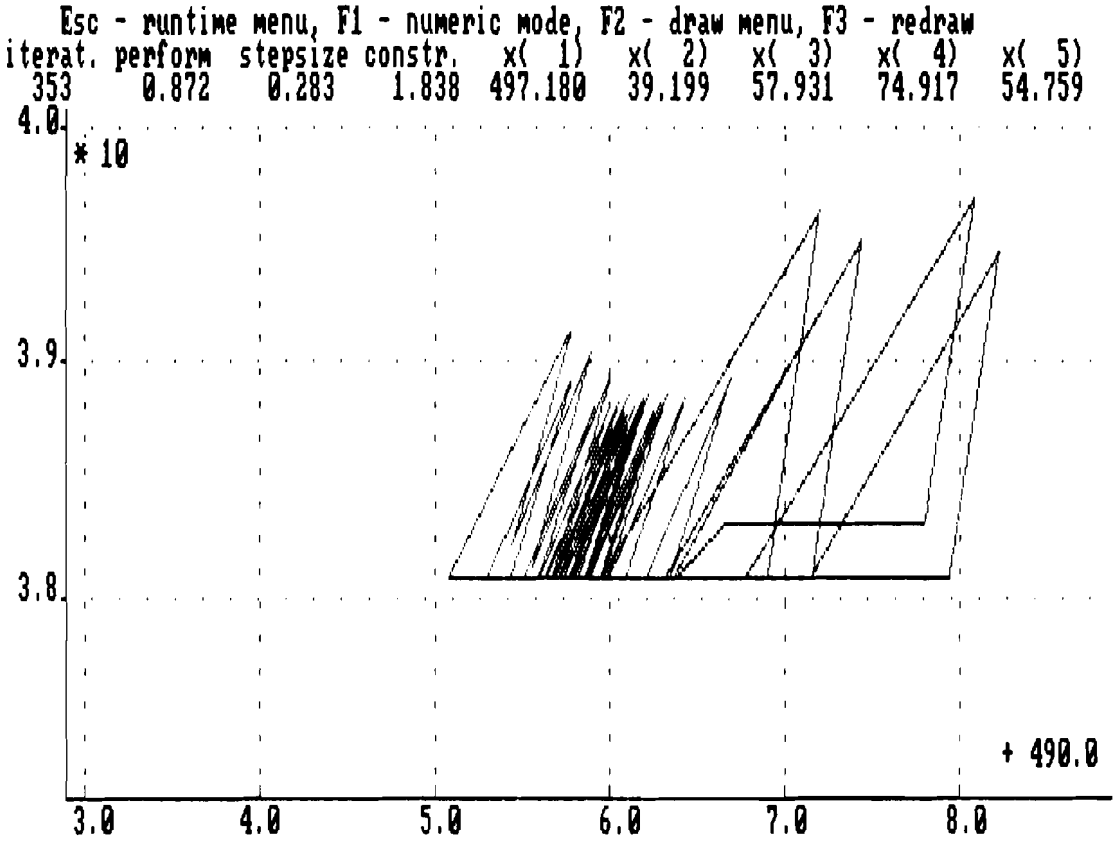


FIGURE 22

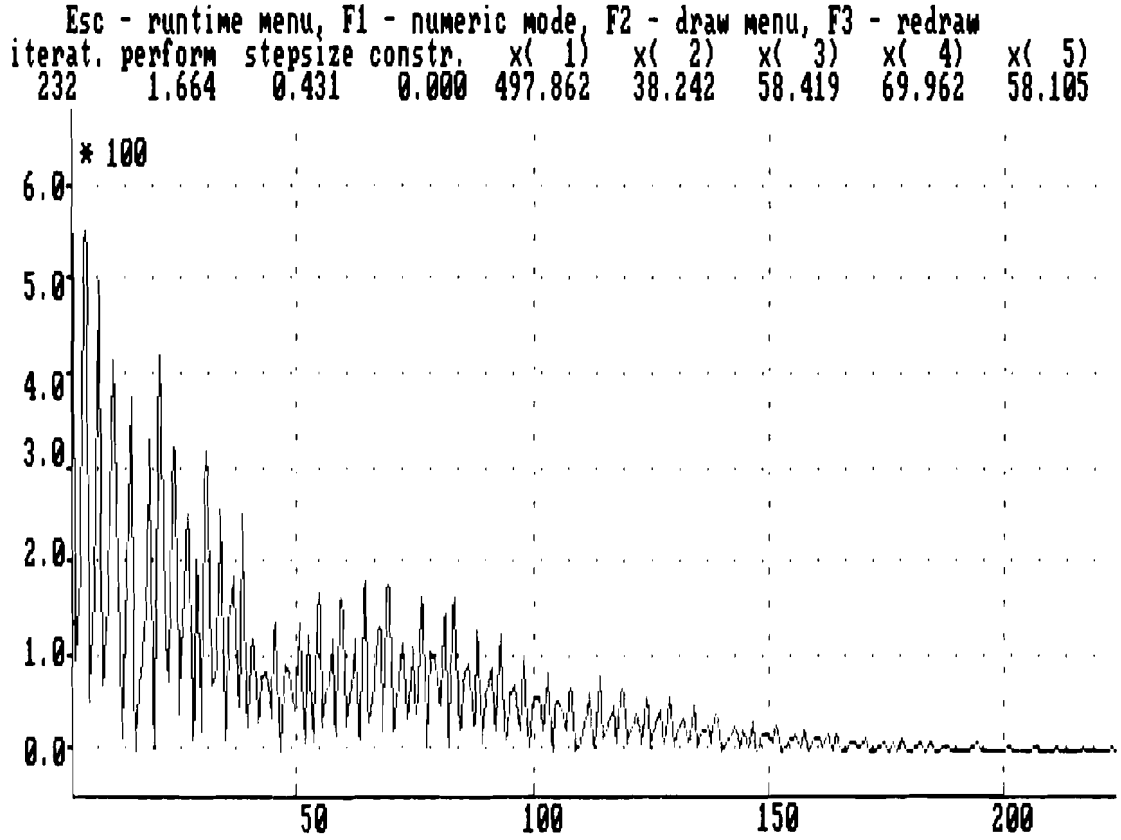


FIGURE 23

number 1.1 opposite the scaling parameter is actually 110. Besides scaling parameter there could be also shifting parameter, for instance it is present in the Figure 22, in the lower right corner of the graphing area above the horizontal axis. Its value is 490.0 and it means that it is necessary to add this number to values which mark horizontal axis. Sometimes both scaling and shifting parameters are present.

In order to change graph appearance it is necessary to invoke GRAPH SETTINGS MENU by pressing the F2 key any time during graphing mode. For instance curves on Figure 20 are too edgy and in order to investigate stationary process the horizontal scale was increased and vertical scale was decreased which produced Figure 21. The majority of the GRAPH SETTINGS MENU parameters will be rarely changed by the user, except *scale x axis* and *scale y axis* parameters. This menu is of the second type and the parameter values are modified as described at the beginning of the section 4.3 (Figure 24).

Menu parameters:

- *zoom* — the real number a_1 which defines magnification of the area around current point with preserves scaling proportions. Use this parameter when exploring convergence pattern in deterministic problems when successive steps becomes progressively smaller

- *previous iterations* — this parameter defines the number m_1 of the previous iterations which will be depicted after switching from numeric to the graphing mode. These last iterations are used also for automatic scaling. Important role in what follows play the span of the graphed variables during the last m_1 iterations, which will be denoted $sp(x)$ or $sp(y_k)$, $k = 1, \cdots 4$.

- *scale x axis* — the real positive coefficient a_x which defines the scale of the horizontal axis. The span $sp(x)$ of the independent x variable occupies $\gamma_x = 0.9 a_x a_1$ part of the horizontal graphing space. If $\gamma_x > 1$ then the part that fits into the whole horizontal graphing space is shown

- *scale y axis* — the real positive coefficient a_y which defines the scale of the vertical axis. If automatic scaling is selected then the span $sp(y)$ of the *main y variable* occupies $\gamma_y = 0.9 a_y a_1 b_i$ part of the vertical graphing space, where b_i is the scale of the *main y variable* and i is its number. If $\gamma_y > 1$ then the part that fits into the whole vertical graphing space is shown

- *main y variable* — This is the dependent variable which is used for the vertical scaling of the graph. Values of all other dependent variables are not shown on the vertical axis.
- *scale y variable 1* — this is the scale of the first selected dependent variable, for explanation see the description of the *scale y axis* parameter. All *scale y variable i* parameters have the similar meaning.
- *automatic shifting* — this parameter can be either *yes* or *no*. In the case of *yes* spans of all dependent variables are centered around the central horizontal line of the graphing space. In the case of *no* the total span of all variables is computed which is used for scaling
- *screen marking* — this parameter can be either *yes* or *no*. In the case of *yes* the graphing area will be marked with ticks as shown on the Figures 20–23. In the case of *no* there would be no marks on the graphing area.
- *automatic scaling* — all the dependent variables are scaled to the size of the *main y variable* span

For the most of the graphing within SQG-PC the values of the last three parameters should be set to *yes*. In Figure 25 the most common change of the graphing parameters is shown which produced Figure 21.

Leave the GRAPH SETTINGS MENU after finishing with changes by pressing the *Esc* key.

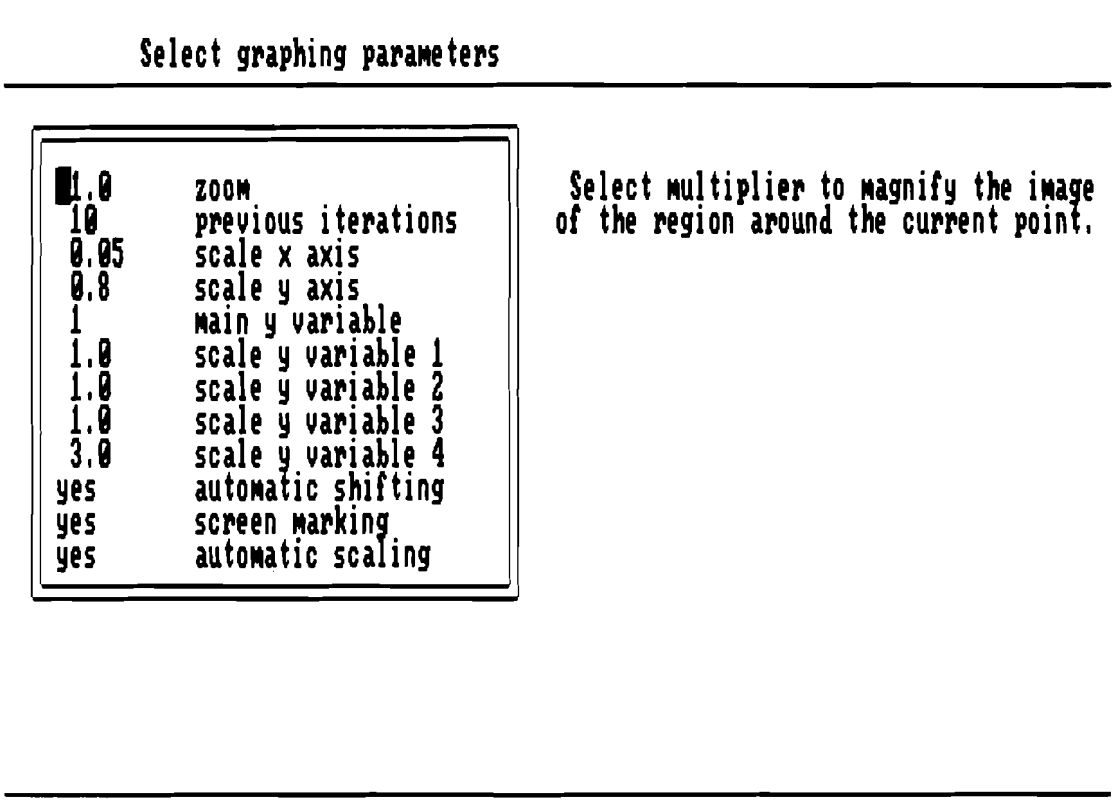


FIGURE 24

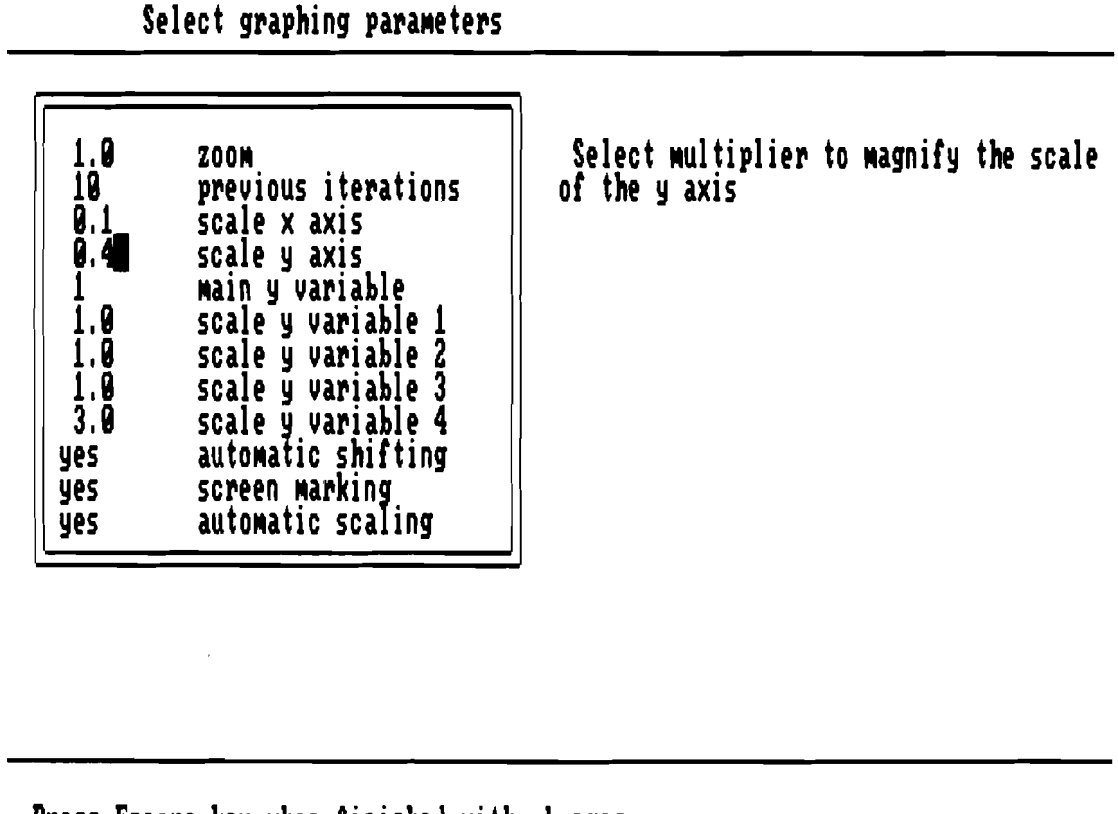


FIGURE 25

4.9.1.6. Changing penalty coefficient

This makes sense only if GENERAL LINEAR option is selected in the CONSTRAINTS MENU. In order to do this choose CONSTRAIN option from the RUNTIME MENU. In this case there is only one relevant parameter and instead of the whole menu only one line on the place of the RUNTIME MENU will appear and the current table with the process information will be retained (see Figure 26). The manipulation with this single line is exactly the same as described at the beginning of the section 4.3. When the changes are finished press the *Esc* key to exit this menu and to go to the next selected menu or to resume iterations loop.

4.9.1.7. Quitting iterations loop and changing algorithm

To do this choose the QUIT option from the RUNTIME MENU. Then after exiting from the RUNTIME menu the MAIN MENU will appear again (see section 4.2). At that point everything assessible from the MAIN MENU can be changed, including change of the stepsize and step direction options, switch to the AUTOMATIC mode and quitting altogether.

4.10. Termination in the INTERACTIVE mode

In order to terminate the program execution in the INTERACTIVE mode choose the QUIT option from the MAIN MENU. This option has priority over all other options and its execution will be started immediately after exiting from the MAIN MENU. The TERMINATION MENU will appear which defines the final actions of the program (see Figure 27). This menu is of the first type and selection from it is made by the means of highlighting as described at the beginning of section 4.2.

5.00 penalty coefficient

iteration	method performance / 1e0	step size / 1e0	constraints violat. / 1e0	x(1) / 1e0	x(2) / 1e0	x(3) / 1e0	x(4) / 1e0	x(5) / 1e0
1	0.000	4.000	3.703	499.920	38.100	99.999	23.999	12.999
2	17.094	25.000	45.166	500.000	102.319	252.000	230.555	218.555
3	11.514	25.000	578.133	498.530	38.100	80.443	107.999	95.018
4	2.668	25.000	96.263	496.589	38.100	0.000	0.000	0.000
5	3.146	20.000	185.494	500.000	102.319	192.277	252.000	229.172
6	1.739	16.667	550.470	498.097	38.100	16.696	139.831	83.721
7	1.091	14.286	53.051	496.213	38.100	0.000	29.204	0.000
8	1.420	12.500	156.665	500.000	102.319	207.788	233.168	179.991
9	1.598	11.111	497.970	498.153	38.100	96.731	118.779	44.720
10	1.181	10.000	73.033	496.323	38.100	0.000	17.873	0.000

FIGURE 26

Stochastic QuasiGradients TERMINATION

VARIABLES

PARAMETERS

Display the final values of variables

Select item by arrow keys or by the key with the first letter of the desired option, confirm your choice by RETURN key, exit menu by pressing Esc key

FIGURE 27

Menu options:

- VARIABLES — if this option is selected then the final point will be displayed on the screen, otherwise it will not be displayed. Irrespectively of this option selection it will be stored in the file with the name specified in the INITIALIZATION MENU.
- PARAMETERS — If this option is selected then changes to the algorithm parameters made by the user during the current session will be retained and become the new default values of the algorithm parameters. Otherwise all the changes made by the user will be discarded and a new session will start with the previous default values.

After making the selection exit this menu by pressing the *Esc* key, which will terminate the program and return to DOS.

4.11. AUTOMATIC mode

This mode should be chosen after parameters of the algorithm are tuned in the INTERACTIVE mode. To choose this mode select the HIDE option from the MAIN MENU. Then after exiting the MAIN MENU and the processing of other options the STOP MENU will appear, which defines the stopping criterion (see Figure 28). This menu is of the second type and defines the values of parameters, how to make changes in these values are described at the beginning of section 4.3.

Menu parameters:

- *number of iterations* — the maximal amount of iterations algorithm is allowed to perform. This is the first stopping criterion of the program.
- *minimal stepsize* — The minimal allowed value of the stepsize. If the stepsize remains below tis level for the successive number of iterations defined by *iterations to check* parameter then the second stopping criterion of the program is satisfied

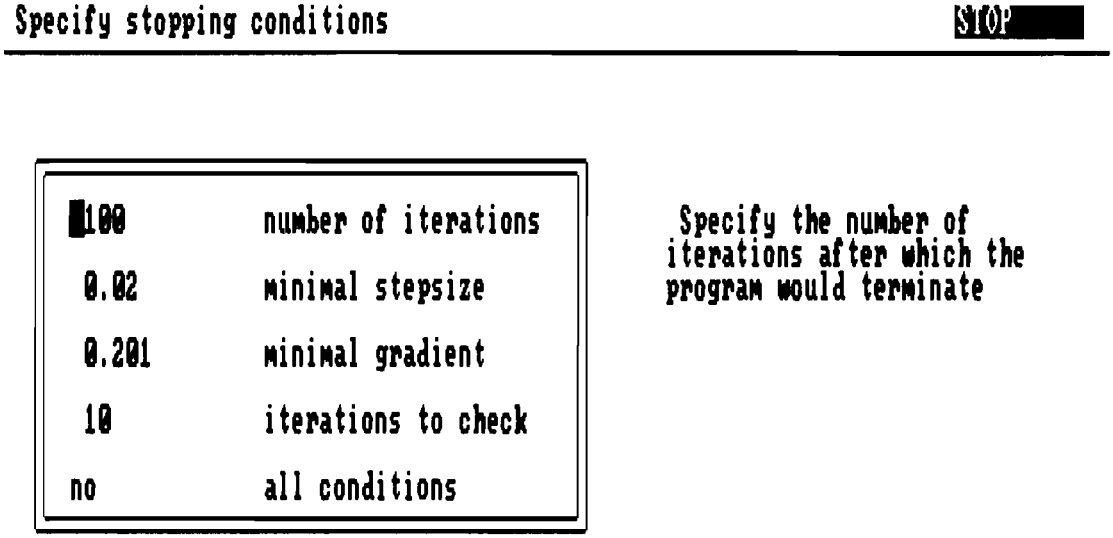


FIGURE 28

- *minimal gradient* — The minimal allowed value of the norm of the step directions average. Type of the average is defined in the INFORMATION MENU. If this norm remains below the level specified by the *minimal gradient* parameter for the successive number of iterations defined by the *iterations to check* parameter then the third stopping criterion of the program is satisfied.

- *iterations to check* — The successive number of iterations to check for the second and the third stopping criterion

- *all conditions* — this parameter can take values *yes* and *no*. In case of *yes* program termination will occur only after all three stopping criterions are satisfied. In the case of *no* the program would terminate after arbitrary one of the stopping criterions will be satisfied.

Exit this menu by pressing the *Esc* key. The program will display after that PARAMETER DEFINITION MENU where the values of the algorithm parameters can be changed (see section 4.8). After that the program will clear the screen and continue execution independently from the user until the stopping criterions would be satisfied and terminate. However, the user can regain control and invoke INTERACTIVE mode by pressing the *Esc* key. This will produce RUNTIME MENU (see section 4.9).

APPENDIX

This is the problem which is contained on the Example diskette and has its origins in the water resources management. For a detailed explanation and comparison of different solution techniques see [17].

$$\text{minimize } F(x) = E_\omega f(x, \omega) = x_0 + c\, E_\omega\left\{\max\left\{0, \max_{1=2,3,4}\{\omega_{i-1} + d_i - x_i\}\right\}\right\}$$

subject to constraints

$$\begin{aligned}
x_1 + x_2 &\le 156.448\\
x_1 + x_2 + x_3 &\le 201.866\\
x_1 + x_2 + x_3 + x_4 &\le 225.297\\
x_0 + x_1 &\ge 512.886\\
x_0 + x_1 + x_2 &\ge 592.872\\
x_0 + x_1 + x_2 + x_3 &\ge 654.152\\
x_0 + x_1 + x_2 + x_3 + x_4 &\ge 720.183
\end{aligned}$$

$$\begin{aligned}
100.0 &\le x_0 \le 500.0\\
38.1 &\le x_1 \le 102.319\\
00.0 &\le x_2 \le 252.0\\
00.0 &\le x_3 \le 252.0\\
00.0 &\le x_4 \le 252.0
\end{aligned}$$

where $d_i = 12.7$, $i = 2, 3, 4$. The random vector $\omega = (\omega_1, \omega_2, \omega_3)$ is distributed normally with

expectations $e = (20.2, 27.37, 10.65)$

standard deviations $q = (8.61, 10.65, 6.00)$

$$\text{and correlation matrix} \begin{bmatrix} 1. & 0.360 & 0.125 \\ 0.360 & 1. & 0.571 \\ 0.125 & 0.571 & 1. \end{bmatrix},$$

penalty coefficient c was equal 100. The convenient feature of this problem is that we can easily obtain very good lower bound for solution by minimizing x_0 subject to constraints stated above. This gives $F(x^*) \geq 494.88$ where x^* is the optimal point.

Due to the stochastic nature of the problem it was not possible to obtain the optimal solution with precision, say, 5 decimal digits. Moreover, the objective function of this problem is quite insensitive around optimal point to the changes in some of the variables. To validate results of SQG-PC the test runs on the VAX 780 were used. The version of the program which is installed on the VAX differs from SQG-PC among other things in that it uses projection for constraints handling. The Tables 1 and 2 represent results of two runs with different sequences of random numbers. The stepsize rule was ADAPTIVE 1 and starting point (1000, 100, 100, 100, 100). Each iteration required one observation of random variables. The last column of the table represents the estimate of the value of the objective function $F(x)$ using 10000 independent observations of random variables. Such extensive estimation is impossible on a PC, but here it gives an idea as to how the method approaches the optimum.

TABLE 1

Step number	Stepsize	x_0	x_1	x_2	x_3	x_4	$F(x)$
20	5.000	494.886	56.324	56.324	56.324	56.324	524.380
40	5.000	494.886	40.657	57.329	61.280	66.031	504.566
60	2.500	494.886	40.657	57.329	61.280	66.031	504.566
80	1.250	494.886	40.657	57.329	61.280	66.031	504.566
100	0.625	498.438	38.881	55.553	94.306	36.557	502.248
120	0.625	493.189	39.662	57.021	87.596	41.018	497.002
140	0.313	494.886	40.314	57.672	86.945	40.366	495.706
200	0.313	494.886	40.314	59.672	86.945	40.366	792.706
220	0.313	497.969	39.116	55.788	72.861	57.533	499.343
140	0.313	497.344	39.428	56.100	88.173	41.595	498.337
260	0.313	494.886	40.657	57.329	86.912	40.366	495.782
400	0.313	494.886	40.657	57.329	86.945	40.366	495.782
420	0.313	499.844	38.178	54.850	89.423	42.845	501.282
440	0.156	498.281	38.959	55.631	88.642	42.064	499.416
460	0.156	496.719	39.741	56.413	87.861	41.283	497.641
480	0.156	495.156	40.522	57.194	87.080	40.502	496.001
500	0.156	494.886	40.657	57.329	86.945	40.366	495.782
520	0.156	494.886	40.657	57.329	86.945	40.366	492.782
540	0.156	494.886	38.100	68.598	82.589	36.010	498.586
560	0.156	494.886	38.100	63.390	77.380	46.427	495.158
1000	0.156	494.886	38.100	63.390	77.380	46.427	495.158

TABLE 2

Step number	Stepsize	x_0	x_1	x_2	x_3	x_4	$F(x)$
20	5.000	494.886	56.324	56.324	56.324	56.324	524.380
40	2.500	494.886	48.993	48.993	61.280	66.031	513.800
60	2.500	496.000	40.100	56.772	61.280	67.145	505.757
80	1.250	497.000	39.600	56.272	61.280	68.145	506.897
100	1.250	494.886	40.657	57.329	61.280	66.031	504.566
200	1.250	494.886	40.657	57.329	61.280	66.031	504.566
220	1.250	499.500	38.350	55.022	108.494	23.431	736.529
240	0.625	494.886	40.657	57.329	74.812	52.499	495.735
420	0.625	494.886	40.657	57.329	74.812	52.499	495.735
440	0.625	496.875	39.662	56.335	75.806	53.494	497.946
460	0.313	494.886	40.657	57.329	74.812	52.499	495.735
940	0.156	494.886	40.657	57.329	74.812	52.499	495.735
1000	0.156	494.886	40.657	57.329	74.812	52.499	495.735

In what follows there is the listing of the Fortran code defining this example.

```
      function uf(n,x)
      dimension r(10),d(10)
          dimension x(n)
      common/k1k1k1/nr,pen,d
      common/k1k1k2/k
                                    common/omeg/lomeg,momeg
      if (k.eq.3) go to 300
c
c      Reading data
c
      ifpr=2
          ifid=3
      open(ifid,file='jd.dat')
          open(ifpr,file='jd.rec')
      read(ifid,*) nr
      write(ifpr,*) 'number of random variables'
      write(ifpr,*) nr
      read(ifid,*) pen
      write(ifpr,*) 'penalty coefficient'
      write(ifpr,*) pen
      read(ifid,*) (d(i),i=1,nr)
      write(ifpr,*) 'rhs constants in random inequality'
      write(ifpr,*) (d(i),i=1,nr)
          close(ifid)
          close(ifpr)
      k=3
  300 continue
c
c        function evaluation
c
      if(lomeg.eq.0) go to 301
      if(momeg.ne.1) go to 302
      momeg=0
  301 call ranv(nr,r)
  302 continue
      tid=0.
      do 100 i=1,nr
      sr=r(i)-x(i+2)+d(i)
      if(sr.gt.tid) tid=sr
  100 continue
      uf=x(1)+pen*tid
      return
      end

      subroutine ug(n,x,g)
      dimension x(n),g(n)
      dimension r(10),d(10)
      common/k1k1k1/nr,pen,d
      common/k1k1k2/k
      if (k.ne.0) go to 300
c
c      Reading data
c
      ifpr=2
          ifid=3
          ico=0
      open(ifid,file='jd.dat')
          open(ifpr,file='jd.rec')
      read(ifid,*) nr
      write(ifpr,*) 'number of random variables'
      write(ifpr,*) nr
      read(ifid,*) pen
      write(ifpr,*) 'penalty coefficient'
```

```
      write(ifpr,*) pen
      read(ifid,*) (d(i),i=1,nr)
      write(ifpr,*) 'rhs constants in random inequality'
      write(ifpr,*) (d(i),i=1,nr)
          close(ifid)
          close(ifpr)
      k=3
  300 continue
c
c      Computing gradient
c
      ico=ico+1
      call ranv(nr,r)
      tid=0.
      nid=1
      g(1)=1.
      g(2)=0.
      do 100 i=1,nr
      sr=r(i)-x(i+2)+d(i)
      g(i+2)=0.
      if(sr.le.tid) go to 100
      tid=sr
      nid=i
  100 continue
      if(tid.gt.0.000001)g(nid+2)=-pen
      return
      end

      subroutine ranv(k,rvec)
      dimension rvec(k),ex(3),sig(3,3),rr(3)
      if(kk.ne.0) go to 300
          open(3,file='nor.dat')
          open(2,file='nor.rec')
          read(3,*) nnor
          write(2,*) 'number of uniform observations',nnor
      read(3,*) (ex(i),i=1,k)
      write(2,*) 'expectations ',(ex(i),i=1,k)
      read(3,*) ((sig(i,j),i=1,k),j=1,k)
      write(2,*) 'variance square root ',((sig(i,j),i=1,k),j=1,k)
          close(3)
          close(2)
          ssor=nnor
          amm=2.*sqrt(3.)/sqrt(ssor)
          am2=0.5*ssor
      kk=1
  300 continue
c
c       computing random vector
c

          do 111 i2=1,k
  111 rr(i2)=0.
          do 110 i1=1,nnor
          do 200 i=1,3
          rr(i)=rr(i)+uran(dummy)
  200 continue
  110 continue
      do 112 i2=1,k
          rr(i2)=(rr(i2)-am2)*amm
  112 continue
      do 113 i1=1,k
          rvec(i1)=0.
          do 113 i2=1,k
```

```
113 rvec(i1)=rvec(i1)+rr(i2)*sig(i1,i2)
    do 101 i=1,k
101 rvec(i)=rvec(i)+ex(i)
    return
    end

    function uran(x)
    data k/0/
    if(k.eq.1) go to 300
    x1=3.14159263
    x2=(5.**8/2.**21)/(5.**9/2.**21)
    k=1
300 continue
    s=x1+x2
    x1=x2
    if(s-4.)2,2,1
  1 s=s-4.
  2 uran=s/4.
    x2=s
    return
    end
```

REFERENCES

[1] Dantzig, G. and A. Mandansky (1961): On the solution of two-state linear programs under uncertainty. *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1 Univ. California Press, Berkeley, 165–176.

[2] Ermoliev, Yu. and R. J-B Wets (eds) (1988): Numerical Techniques for Stochastic Optimization Problems. Forthcoming, Springer-Verlag, Heidelberg.

[3] Wets, R. (1983): Stochastic programming: solution techniques and approximation schemes. In: A. Bachem, M. Groetschel and B. Korte, eds., *Mathematical Programming: The State-of-the-Art*, Springer-Verlag, Berlin, 566–603.

[4] Ermoliev, Yu. (1976): *Methods of stochastic programming* (in Russian). Nauka, Moskva.

[5] Kall, P. (1979): Computational methods for solving two-stage stochastic linear programs. *Z. Angew.-Math. Phys.* **30**, 261–271.

[6] Prékopa, A. (1987): Numerical solution of probabilistic constrained programming problems. In: Ermoliev, Yu. and R. Wets eds. *Numerical techniques for stochastic optimization problems*. Springer, Berlin.

[7] Ermoliev, Yu.M. 1983: Stochastic Quasi-Gradient Methods and their Applications to Systems Optimization. Stochastics, No. 4.

[8] Robbins, H. and S. Monro (1951): Stochastic Approximation Methods. Ann. Math. Statist., **22**, 400–407.

[9] Ermoliev, Yu. M., G. Leonardi and J. Vira (1981): The Stochastic Quasi-Gradient Methods Applied to a Facility Location Model. Working Paper WP-81-14, Laxenburg, Austria, International Institutte for Applied Systems Analysis.

[10] Gaivoronski, A. (1987): Stochastic quasigradient methods and their implementation. In: Ermoliev, Yu. and R. Wets eds. *Numerical techniques for stochastic optimization problems*. Springer, Heidelberg.

[11] Gupal, A.M. (1979): Stochastic Methods for Solving Non-Smooth Extremal Problems. Naukova Dumka, Kiev (in Russian).

[12] Urjas'ev, S.P. (1986): Stochastic Quasi-Gradient Algorithms with Adaptively Controlled Parameters. Working Paper WP-86-32 IIASA, Laxenburg, Austria.

[13] Ruszczynski, A. and W. Syski (1986): A method of aggregate stochastic subgradients with on-line stepsize rules for convex stochastic programming problems. *Mathematical Programming Study* **28**, 113–131.

[14] Marti, K. and E. Fuchs (1986): Computation of descent directions and efficient points in stochastic optimization problems. *Mathematical Programming Study* **28**, 132–156.

[15] Pflug, G. (1983): On the determination of stepsize in stochastic quasigradient methods. Collaborative Paper CP-83-25, IIASA, Laxenburg, Austria.

[16] Kushner, H. and D. Clark (1978): *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, New York.

[17] Dupačová, J., A. Gaivoronski, Z. Kos and T. Szántai (1986): Stochastic Programming in Water Resources System Planning: A Case Study and A Comparison of Solution Techniques. Working Paper WP-86-40, IIASA, Laxenburg, Austria.

[18] Rockafellar, R.T. and R. J-B Wets (1986): A Lagrangain finite generation technique for solving linear quadratic problems in stochastic programming. *Mathematical Programming Study* **28**, 63–93.

[19] Birge, J. and R. J-B Wets (1986): Designing approximation schemes for stochastic optimization problems, in particular for stochastic programs with recourse. *Mathematical Programming Study* **27**, 54–102.

[20] Nazareth, J. and R. Wets (1986): Algorithms for stochastic programs: the case of nonstochastic tender. *Mathematical Programming Study* **28**, 1–28.

[21] Robinson, S.M. (1987): Bundle–Based Decomposition: Conditions for Convergence. Working Paper WP-87-80, IIASA, Laxenburg, Austria.