WORKING PAPER

STOCHASTIC NONLINEAR PROGRAMMING SYSTEM

N. Roenko
V. Loskutov
S. Uryas'ev

October 1989
WP-89-075

IIASA
International Institute
for Applied Systems Analysis

STOCHASTIC NONLINEAR PROGRAMMING SYSTEM

N. Roenko
V. Loskutov
S. Uryas'ev

October 1989
WP-89-075

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

STOCHASTIC NONLINEAR PROGRAMMING SYSTEM

USER'S GUIDE

N. Roenko, V. Loskutov, S. Uryas'ev

F O R E W O R D

This paper contains a detailed description of the Stochastic Nonlinear Programming System (SNLP) intended for solving stochastic optimization problems with simple recourse. This system is a result of collaboration between the Glushkov Institute of Cybernetics (Kiev, USSR) and the International Institute for Applied System Analysis (IIASA) within the framework of the Adaptation and Optimization Project in the System and Decision Sciences Program.

Alexander B. Kurzhanski,
Chairman
System and Decision Sciences Program

C O N T E N T S

1. INTRODUCTION

This paper may be considered as a further development of the IIASA/SDS activity aimed at developing solution methods, data standards, software for stochastic optimization problems etc. The state of activities for software development is reflected in the ADO/SDS library which is being constantly replenished. The SNLP being intended for IBM/PC and compatibles shall add to this library. The SNLP differs to some extent from the analogous systems of the ADO/SDS library.

The Stochastic Nonlinear Programming System (SNLP) was developed above all out of necessity to solve practical problems. Above all these are problems of power systems development accounting for device failures, problems of prospective planning accounting for demand uncertainty, and agricultural problems accounting for weather conditions.

Taking into consideration that the main source of stochastic problems is represented by deterministic problems, the SNLP provides for solving both stochastic and deterministic problems. It also supports the conversion of a deterministic problem into its stochastic analog and vice versa.

There is no exaggeration in stating that two-stage stochastic problems, or problems with recourse, occupy a central place among the range of stochastic optimization problems. Evidence of this is found in the voluminous literature dedicated to these problems, and in the activity in the development of software for their solution. The most widely spread are linear

stochastic problems with recourse, which may be regarded as a certain generalization of linear (LP) and nonlinear (NLP) programming problems with linear constraints accounting for the possible uncertainty of the source data.

As a rule, the values of many parameters in real problems are never absolutely accurate. This inaccuracy may have various nature. For instance, in problems of long-term planning some model coefficients are obtained by forecasting, and in the macroeconomic models some coefficients are obtained by means of aggregate evaluation. Among the sources of the above inaccuracy may be impossibility or extreme sophistication of the precise evaluation of parameters, use of expert estimates concerning the parameter values, etc. For certain types of problems (eg. agricultural production problems) this inaccuracy of parameters may amount to 100% and more.

Disregarding the uncertain nature of parameters and replacing them by specific values (for instance, mathematical expectation) in the solvution of LP or NLP problems may lead to invalid results, due to the problem's instability in relation to the source data. Consequently, the optimization model may be inadequate. Information about the solution stability of a problem may be obtained by means of post-optimum LP-analysis (the latter's abilities, however, are rather limited).

If this adequacy is lacking, the given situation may be tackled by various approaches: optimization by minimax criterium, converting a problem to a stochastic programming one with probability constraints, or to a stochastic problem with

recourse [1].

This paper deals with the latter approach based upon creating the stochastic analog for a nonlinear, and sometimes linear, programming problem (a stochastic problem with simple recourse). This approach assumes that uncertain or inaccurate source data of the model may be interpreted as random values.

The main ideas used in the algorithm for solving stochastic problems are reflected in [4]. A special modification of the well-known stochastic quasigradients algorithm for stochastic problems with recourse is implemented in the SNLP.

This paper contains a description of the Stochastic Nonlinear Programming System (SNLP) intended for solving stochastic problems with recourse.

The SNLP contains a special matrix-graphic editor for entering and modifying data for optimization problems in a user-friendly form.

The SNLP has been influenced by the IIASA activity on "data standartization" for stochastic optimization problems.

At present, activities are being carried out to develop the SNLP in many directions. The SNLP applicability will be extended to the solution of problems with complete recourse as well as dynamic multi-stage problems. The number of distributions processed will also be enlarged. The SNLP will be extended for the a very important feature, sensitivity analysis. The SNLP sphere of usage will be expanded as well. The system is intended to be used in decision support systems.

2. PROBLEM STATEMENTS

The SNLP is intended to solve the following problems:

- linear programming (LP) problems;
- nonlinear programming (NLP) problems under linear constraints;
- stochastic programming (SP) problems with recourse.

The first two types of problems will be called deterministic, the third - stochastic.

The nonlinear programming problems under linear constraints are as follows:

$$\min_x \{ (c,x) + F(x) \} \tag{1}$$

subject to

$$l \le \begin{vmatrix} x \\ Ax \end{vmatrix} \le u, \tag{2}$$

If the function F(x) is absent in (1), the problem turns into a linear programming problem.

Stochastic programming problems with recourse are as follows:

$$\min \{ (c,x) + F(x) + Q(x) \} \tag{3}$$

$$Q(x) = \mathbb{E} \min \{ q^+y^+ + q^-y^- \mid y^+ - y^- = h(w) - T(w)x, \quad 0 \le y^+ \le d^+ , \; 0 \le y^- \le d^- \} \tag{4}$$

subject to

$$l \le \begin{vmatrix} x \\ Ax \end{vmatrix} \le u, \tag{5}$$

where $\mathbb{E}$ stands for mathematical expectation, and w is a set of random values.

For every value of w, the following equation must hold:

$y_i^+(w) \cdot y_i^-(w) = 0$

i.e. one of these two variables must equal 0 for any w.
Here and below i is a entry (element) number of the vector.

The equalities in (4) determine the so-called stochastic constraints. The vector h(w) is called the right hand side of the stochastic constraints or the stochastic right hand side. The matrix T(w) is a technology matrix.

The linear function in (4) determines the value of the recourse function. The vectors q^-, q^+ contain specific recourse coefficients.

Each entry of the vector h(w) may be deterministic, random with a discrete or standard uniform, normal, exponential distribution, or random with the distribution generated in the user's subroutine CALCTH.

The probability properties of matrix T(w) are specified by rows. The matrix T(w) row may consist entirely of deterministic entries. Otherwise all the random (and deterministic) entries of this row are generated in CALCTH.

3. LINK BETWEEN DETERMINISTIC AND STOCHASTIC PROBLEMS

An important source of stochastic problems with recourse is represented by deterministic optimization problems, in particular linear and nonlinear programming problems under constraints containing random values. As a rule, a deterministic problem is solved first, and then, after analyzing the obtained solution, its stochastic analog is created.

To analyze the solution of a stochastic problem it may be useful to compare its solution with that of a deterministic problem. For that purpose the SNLP supports the linkage between deterministic and stochastic problems.

Let us consider this linkage in detail. Let a linear programming problem have the following form:

$$\min_x cx \qquad (6)$$

subject to

$$Ax = b\ , \quad x \geq 0. \qquad (7)$$

Let some entries $a_{i,j}$ and some entries b_i be random values.

Let us extract from (7) the constraints with random values in the A and b rows. Let us form the matrix T(w) and the vector h(w) from these constraints, and the matrix A^1 and the vector b^1 from those remaining. So, A^1 and b^1 consist entirely of deterministic entries, and the matrix T(w) and/or the vector h(w) also have random entries. Each T(w) row or the corresponding h(w) entry should have at least one random value.

Then the constraints (7) may be expressed as follows:

$$A^1 x = b^1 \qquad (8)$$

$$T(w)x = h(w) \qquad . \qquad (9)$$

The requirement for equality (9) to hold in every realization of w random values often leads to the absence of a feasible solution of an LP or NLP problem.

Let us specify the occuring discrepancies as y^+, y^-:

$$y_i^+(w) = \max\,(h_i(w) - [T(w)x]_i, 0) \geq 0$$
$$y_i^-(w) = \max\,([T(w)x]_i - h_i(w), 0) \geq 0 \quad , \tag{10}$$

which express the losses caused by the violation of the constraints (9) motivated by the uncertainty of the source data.

Using the well-known economic interpretation of LP problems ("input - output" models) let us consider the $a_{i,j}$ coefficients as the specific input, the b_i as the available resources, the $y^-(w)$ discrepancies as the losses due to the shortage of resources, and the $y^+(w)$ as the losses due to the surplus of resources. It is obvious that in every realization of w only one kind of losses is observed; therefore for every w:

$$y_i^+(w) \cdot y_i^-(w) = 0.$$

The violation of constraints leads to additional losses for their correction which may be considered proportional to the discrepancies. Let us specify the specific losses due to the shortage and the surplus of resources as q^- and q^+ . Then the expected losses caused by the violation of constraints (9) are as follows:

$$\mathbb{E}\,(q^-y^-(w) + q^+y^+(w)) \tag{11}$$

subject to

$$y^+(w) - y^-(w) = h(w) - T(w)x \quad . \tag{12}$$

Minimizing the expected losses with respect to $y^+(w)$ and $y^-(w)$, we shall obtain the function Q(x) expressing the minimum expected losses for every x.

Adding the function Q(x) to the objective function, we shall obtain a problem with recourse which may be solved by means of

the SNLP.

The constraints (2) may contain stochastic entries, which does not necessarily imply a violation. The method of creating stochastic constraints, and the q^-, q^+ values depending on the type of linear constraints, are shown in Tables 1 and 2.

The method of convert deterministic problems to their stochastic analogs is described in detail because their close relation determines the SNLP usage.

The deterministic analogs of stochastic problems with recourse (NLP and LP problems) are created by inverse conversion, i.e. by averaging the h(w) and T(w) random values, deleting the recourse function Q(x) from the objective function (3) and adding to (5) the following constraints:

$$\begin{aligned} &\bar{T}_i x = \bar{h}_i \quad , && \text{if } q_i^- \neq 0,\ q_i^+ \neq 0 \\ &\bar{T}_i x \geq \bar{h}_i \quad , && \text{if } q_i^- = 0,\ q_i^+ \neq 0 \\ &\bar{h}_i \geq \bar{T}_i x \quad , && \text{if } q_i^- \neq 0,\ q_i^+ = 0 \end{aligned} \qquad (12a)$$

where $\bar{T}_i$ and $\bar{h}_i$ are mathematical expectations of T_i and h_i respectivly.

4. ALGORITHMS FOR SOLVING PROBLEMS

One of the multiplicative simplex methods [2] is implemented for solution of linear programming problems.

For solution of nonlinear programming problems, one of the linearization methods for solving problems with a nonlinear objective function and linear constraints [6] is implemented.

For solving stochastic problems with recourse the stochastic

Table 1

Creating Stochastic Analogs For Deterministic Single - Bounded Constraints $[Ax]_i \leq b_i$ and $[Ax]_i \geq b_i$

Constraint type	Value of q^-	Value of q^+	Value of $h_i(w)$	The "stochastic" row is deleted from A
=	not equal 0	not equal 0	b_i	Yes
$\leq$	greater than 0	0	b_i	Yes
$\geq$	0	greater than 0	b_i	Yes

Table 2

Creating Stochastic Analogs For Deterministic Double-Bounded Constraints $l_i \leq [Ax]_i \leq u_i$

Determin./Stochastic			Value			The "stochastic" row is deleted from A
l_i	u_i	A_i	q_i^-	q_i^+	$h_i(w)$	
any	any	stoch.	greater than 0	0	u_i	Yes
			0	greater than 0	l_i	Yes
stoch.	stoch.	any	greater than 0	0	u_i	Yes
			0	greater than 0	l_i	Yes
deter.	stoch.	deter.	greater than 0	0	u_i	No
stoch.	deter.	deter.	0	greater than 0	l_i	No

quasigradient method with averaging [7], [8], [20] is used:

$$x^{s+1} = \Pi_x(x^s - \rho_s d^s), \qquad s = 0,1,2, \ldots, \qquad (13)$$

where s is the iteration number,

ρ_s is the stepsize,

d^s is the step direction,

x^s is the solution vector estimate (current point),

Π_x is the projection on the set X.

This algorithm is similar to the one in the SQG system [5].

The operation of projection represents a special quadratic programming problem. To solve it, a special modification of the conjugate gradients method is used [6].

To form the direction vector d^{s+1}, the vector d^s and the stochastic quasigradient $g^s(x^s)$ are used. The calculation is executed as follows:

$$d^{s+1} = (g^s + \gamma_s d^s)/(1 + \gamma_s) \quad , \qquad d^0 = 0$$

where γ_s is a scalar factor.

The condition:

$$\langle d^s, g^s \rangle \geq 0$$

is verified in every iteration. If this condition is violated, then $\gamma_s = 0$.

The r_s and γ_s calculatings are based upon [7] and [8].

$$r_s = r_0 \exp(\langle \Delta^s, g^s \rangle / Z^s)$$

$$\gamma_s = \gamma_0 \exp(\langle \Delta^{s-1}, g^s \rangle / Y^s)$$

$$\Delta^s = x^s - x^{s-1} \quad ,$$

where r_0, γ_0 are constants, and Z^s, Y^s are estimates of the $\langle \Delta^s, g^s \rangle$ and $\langle \Delta^{s-1}, g^s \rangle$ average values respectively. One more

peculiarity of the algorithm is the method of calculating the stochastic quasigradient g^s :

$$g^s = c + \hat{F}_x(x) + \hat{Q}_x(x,w)$$

where $\hat{F}_x(x)$ is an element from the subdifferential $\partial_x F(x)$ of convex function $F(x)$ and $\hat{Q}_x(x,w)$ is an element from the subdifferential $\partial_x Q(x,w)$ with respect to the first compounent x.

The recourse function $Q(x)$ is a separable one:

$$Q(x) = \sum_i \mathbb{E}\, Q_i(x,w)$$

$$Q_i(x,w) = \min \{ q_i^+ y_i^+ + q_i^- y_i^- \mid y_i^+ - y_i^- = h_i(w) - [T(w)x]_i ,$$
$$0 \le y_i^+ \le d_i^+, \quad 0 \le y_i^- \le d_i^- \}$$

The stochastic quasigradient of every term of the separable function $Q(x)$ is calculated as follows:

$$\nabla_x Q_i(x,w) = \begin{cases} -q^+ T_i(w), & \text{if } h_i(w) - [\, T_i(w)x \,] \ge 0 \\ q_i^- T_i(w), & \text{if } h_i(w) - [\, T_i(w)x \,] \le 0 \end{cases} \qquad (14)$$

In some cases, if the distribution of $h_i(w)$ is known, we may calculate not only the stochastic quasigradient $\partial_x Q_i(x,w)$, but also the subgradient $\hat{Q}_x(x)$:

$$\hat{Q}_x(x) \in \partial_x Q_i(x) = \mathbb{E}\, \partial_x Q_i(x,w)$$

In that way we may analytically calculate the subgradient for these $Q_i(x)$ through the corresponding formulae. The formulae for the uniform, exponential, normal, and discrete distributions of the entries $h_i(w)$ are shown below (T_i is deterministic in these cases).

If the $h_i(w)$ are uniformly distributed on the $[\alpha_i, \beta_i]$ segment, then the function $Q_i(x)$ is differentiable and

$$\nabla_x Q_i(x) = \begin{cases} -q_i^+ T_i, & \text{if } [Tx]_i \le \alpha_i \\ \left[-q_i^+ + \dfrac{(q_i^+ + q_i^-)}{\beta_i - \alpha_i}([Tx]_i - \alpha_i)\right] T_i, & \text{if } \alpha_i \le [Tx]_i \le \beta_i \\ q_i^- T_i, & \text{if } [Tx]_i \ge \beta_i \end{cases}$$

If the $h_i(w)$ has an exponential distribution with density

$$\rho(\tau) = \begin{cases} \lambda_i e^{-\lambda_i \tau}, & \text{if } \tau \ge 0 \ (\lambda_i \ge 0) \\ 0, & \text{otherwise} \end{cases},$$

then the function $Q_i(x)$ is differentiable and

$$\nabla_x Q_i(x) = \begin{cases} -q_i^+ T_i, & \text{if } [Tx]_i \le 0 \\ (q_i^- - (q_i^+ + q_i^-) e^{-\lambda_i [Tx]_i}) \cdot T_i, & \text{if } [Tx]_i \ge 0 \end{cases}.$$

If $h_i(w)$ has a normal distribution with the average m_i and the deviations σ_i, then the function $Q_i(x)$ is differentiable and

$$\nabla_x Q_i(x) = \left[-q_i^+ + (q_i^+ + q_i^-) N(Z_i)\right] T_i$$

where $Z_i = \dfrac{[Tx]_i - m_i}{\sigma_i}$, and $N(t)$ is the distribution function of the standard normal distribution.

If h_i accepts the values of $a_1, \ldots, a_{k_i}$ with the probabilities of $p_1, \ldots, p_{k_i}$ correspondingly, then $Q_i(x)$ is a piecewise linear convex function and may be represented as follows:

$$Q_i(x) = \max_{L \in [0, k_i]} (S_{iL}[Tx]_i + E_{iL})$$

where $S_{iL} = (q_i^+ + q_i^-) \sum_{l=1}^{L} p_l - q_i^+$,

$$E_{iL} = q_i^+ \sum_{l=1}^{k_i} a_l p_l - (q_i^+ + q_i^-) \sum_{l=1}^{L} a_l p_l$$

Here it is assumed that $\sum_{l=1}^{0} a_l p_l = 0$, $\sum_{l=1}^{0} p_l = 0$.

One more peculiarity of the algorithm is the technique of forming the set X on which the projection is performed.

The feasible set may be represented as follows:

$$K = K_1 \cap K_2,$$

$$K_1 = \{ x: \ l \le \begin{Bmatrix} x \\ Ax \end{Bmatrix} \le u \}$$

$K_2 = \{x: \ \forall w \in \Omega \ \exists \ 0 \le y^+ \le d^+, 0 \le y^- \le d^-, \ y^+ - y^- = h(w) - T(w)x\}$

Here Ω denotes a set of values accepted by the random value w. K_1 stands for a set of fixed constraints, and K_2 is a set of induced constraints.

Using the induced constraints implies certain difficulties. First, the set of induced constraints is specified indirectly. Second, if a point does not belong to the set of induced constraints, then the stochastic quasigradient at this point does not exist.

To overcome these difficulties the L-shape technique is used [16]. The set X^s is formed in a special way at every iteration; it is upon this set that the projection is executed on this set

$$X^s = K_1 \cap K_2^s,$$

where K_2^s is the linear approximation of the set of induced constraints in the vicinity of the point x^s.

constraints in the vicinity of the point x^s.

$$K_2^{s+1} = \begin{cases} K_2^s \ , \quad \text{if} \quad y^+ \le d^+, \ y^- \le d^- \\ K_2^s \cap \{x: \ h_i(w) - [T(w)x]_i \le d_i^+ \ \text{ for } \forall \ i: \ y_i^+ > d_i^-\} \cap \\ \cap \{ \ x: \ [T(w)x]_i - h_i(w) \le d_i^- \quad \text{for } \forall \ y_i^- > d_i^- \ \}. \end{cases}$$

There is a special technique of forming the approximation set K_2^s, which adds new constraints and automatically deletes inactive constraints.

5. INPUT AND OUTPUT DATA

5.1. General Information

The files used by the SNLP system are shown in Table 3. The type of problem under consideration specifies the corresponding set of files, as shown in Table 4.

The files relating to the same problem should be present on the same disk and in the same directory. They should have unique names with varying extensions (for instance TEST.MPS, TEST.PRT, TEST.X).

The file structure in the SNLP depends on the MPS standard in the mathematical programming [9], [10], [14], and reflects an attempt to ensure a certain compatibility with popular systems similar to the SNLP (for instance, MINOS [11], SPORT [12], ADO/SDS library [13]).

To use the SNLP system you need not know the file structure, because all the input data are entered in the SNLP by means of specialized editors in matrix-graphic representation. (It is

Table 3

Optimization Problem Files

File Extension	File Description
.SPC	Specification file for the stochastic solver
.MPS	Linear deterministic components for LP, NLP, SNLP
.STH	Stochastic input data
.X	Initial, current or resulting values of the variable x
.PRT	Problem solution and various diagnostics
.FOR	Source file for subroutines CALCFG and CALCTH
.OBJ	Object file for subroutines CALCFG and CALCTH
.EXE	Optimizers and convertors

Table 4

Problems and Files

File Types	Problems			
	LP	NLP	Stochastic	
			linear, with standard distributions of random values	nonlinear, or with distributions specified by user
.SPC	-	-	+	+
.MPS	+	+	+	+
.STH	-	-	+	+
.X	+	+	+	+
.PRT	+	+	+	+
.FOR	-	+	-	+
.OBJ	-	+	-	+
.EXE	-	+	-	+

also possible to create and modify these files outside the SNLP by means of a text editor).

The SPECS file (.SPC) contains the parameter values for the stochastic optimizer. This file is not utilized while solving LP and NLP problems. The SPECS file structure is analogous with the systems [11], [12].

The MPS file (.MPS) contains information about the linear deterministic components of linear, nonlinear and stochastic programming problems. The MPS file structure is based upon an MPS format which is the standard de facto for linear programming systems [14]. "Standard" MPS files are correctly processed by the SNLP, accounting for some peculiarities.

First: the SNLP enters only the first vector of objective coefficients, the first vector of bounds, the first vector of ranges.

Second: while dealing with a "standard" MPS file, you don't need to specify the name of the matrix A column consisting entirely of zeros, as is the case with the SNLP.

Third: the default left bound in the BOUNDS section of a "standard" MPS file equals 0, while in the SNLP it equals $-\infty$.

The STOCHASTIC file (.STH) contains information about the recourse function Q(x) in a stochastic programming problem with recourse.

The STOCHASTIC file format is similar to the MPS format. It enables a simple conversion of linear and nonlinear programming problems (represented in the MPS format) to their stochastic analogs.

The STOCHASTIC file format was strongly influenced by the IIASA activity to create a "standard format" for stochastic problems with recourse, similar to the MPS format for LP problems ([9], [10]). The SNLP STOCHASTIC file format differs to some extent from the "standard format" to account for the optimization technique features as well as to enhance the efficiency of data entering and processing. Apparently, the "standard format" needs further development for quasigradient type algorithms.

The X file (.X) contains the initial point x^0, the current point x^s, and the solution point x^*. To ensure compatibility, the .X file format was designed to be similar to analogous files in the MINOS and SPORT systems.

The SOLUTION file (.PRT) contains the calculation results.

The FOR file (.FOR) is necessary while creating the user's subroutine CALCFG designed to calculate the nonlinear objective function F(x), or the subroutine CALCTH for the nonstandard distributions of random h(w), T(w). The same .FOR file may also contain the subroutines called from CALCFG and CALCTH. All these subroutines should meet the Microsoft FORTRAN 4.01 requirements [15].

The .OBJ files contain the object CALCFG and CALCTH subroutines as well as the subroutines called from them.

The .EXE files contain the optimization solvers and convertors. LP problems are solved by means of the LPSOLVER.EXE file. SP problems which do not use CALCFG and CALCTH subroutines are solved by means of the STOP.EXE file.

5.2. The Subroutine CALCTH Requirements

Information about the vector h(w) and the matrix T(w) may be presented entirely in the STOCHASTIC file only if all the random entries of the vector h(w) have either a discrete distribution or one of the standard distributions (uniform, normal, exponential), and if the matrix T(w) is deterministic. Otherwise you won't succeed in specifying all the source information in the STOCHASTIC file.

In this case the user should write his subroutine CALCTH designed to calculate the samples of the h(w) and T(w) random values.

The subroutine CALCTH may look as follows:

```
$LARGE
C                    SUBROUTINE PARAMETERS:
C     N - DIMENSION OF THE SOLUTION VECTOR X; INPUT PARAMETER;
C     NROWT  (or M1) - THE NUMBER OF ROWS IN THE T(w);
C                      INPUT PARAMETER;
C     H(NROWT)  - THE STOCHASTIC RIGHT HAND SIDE H(w);
C                      OUTPUT PARAMETER;
C     T(NROWT,N)  - THE TECHNOLOGY MATRIX T(w);
C                      OUTPUT PARAMETER;
C     CODE    - THE RETURN CODE; OUTPUT PARAMETER;
C
      SUBROUTINE CALCTH (N, NROWT, H, T, CODE)
      IMPLICIT    REAL*8(A-H,O-Z)
      INTEGER     N, NROWT, CODE
      DIMENSION   T(NROWT,N), H(NROWT)
      <simulation model - calculating the samples of
                  h_i(w), T_i(w)>
      <assigning values to the h(w) vector and the T(w) matrix>
      <assigning the return code>
      END
```

The first parameter (N) determines the number of the problem variables (the number of the matrix T(w) columns) - n. The second parameter (NROWT) determines the number of the matrix T(w) rows and the vector h(w) dimension (m1). The third parameter is the one-dimensional array H(NROWT) intended for storing the vector h(w) entries. The fourth parameter is the two-dimensional array T(NROWT,N) intended for storing the matrix T(w) entries. The last parameter (CODE) contains the subroutine CALCTH return code: the zero code denotes normal termination, and any other code denotes abnormal termination.

Before the first call of the subroutine CALCTH, the arrays T(NROWT,N) and H(NROWT) contain information obtained from the STOCHASTIC file. With every call of the subroutine, some T(w) and h(w) entries are calculated and placed into the fields H(I),T(I,J) of the arrays H and T. Between the subroutine calls the contents of the arrays H and T remain unchanged.

To store additional information you may create working arrays in the subroutine CALCTH.

The I indices (the array T row numbers and the array H entry numbers) in the CALCTH subroutine must correspond to the consecutive numbers of the T(w) and h(w) rows in the STOCHASTIC file.

The J indices (the T array column numbers) in the subroutine CALCTH correspond to the consecutive numbers of the columns in the section COLUMNS of the MPS file. The sequence of the column names in the COLUMNS section of the MPS file determines the column numbers of the matrix T(w) and, consequently, of the

variables x_j.

Example of the subroutine CALCTH:

```
$LARGE
      SUBROUTINE CALCTH( N, NROWT, H, T, CODE)
      IMPLICIT   REAL*8(A-H,O-Z)
      INTEGER    N, NROWT, CODE
      REAL*8     H(NROWT), T(NROWT,N)
C  WORKING VARIABLES
      INTEGER    NU,NN,I
      DOUBLE PRECISION DSEEDU
      REAL  RU(8)
      DATA DSEEDU/1.D9/
      NU=8
C OBTAINING THE N UNIFORMLY DISTRIBUTED NUMBERS ON SEGMENT [0,1]
      CALL GGUBS(DSEEDU,NU,RU)
C CALCULATING RANDOM ENTRIES OF THE MATRIX T THROUGH THE FORMULA
C (B-A)*X+A  WHERE X IS A RANDOM VALUE UNIFORMLY DISTRIBUTED ON
C SEGMENT [0,1], AND [A,B] IS DISTRIBUTION RANGE OF ENTRY T[I,J]
      T(1,1)=RU(1) + 3.5D0
      T(1,2)=2.D0*RU(2) + 8.D0
      T(1,3)=2.D0*RU(3) + 6.D0
      T(1,4)=2.D0*RU(4) + 9.5D0
      T(2,1)=0.4D0*RU(5) + 0.8D0
      T(2,2)=0.4D0*RU(6) + 0.8D0
      T(2,3)=RU(7) + 2.5D0
      T(2,4)=8.D0*RU(8) + 36D0
      CODE=0
      RETURN
      END
C UNIFORM DISTRIBUTION GENERATOR
      SUBROUTINE GGUBS(DSEEDU,N,R)
      INTEGER N,I
      REAL R(N)
      DOUBLE PRECISION DSEEDU, D2P31M, D2P31
      DATA D2P31M/2147483647.D0/, D2P31/2147483648.D0/
```

```
      DO 5 I=1,N
         DSEEDU = DMOD(16807.D0*DSEEDU,D2P31M)
    5 R(I)=DSEEDU/D2P31
      RETURN
      END
```

5.3. The Subroutine CALCFG Requirements

If the objective function (1) contains the nonlinear function F(x), the user should create the subroutine CALCFG designed to calculate the values of the function F(x) at any given point x. To enhance the efficiency of the SNLP, you may make in this subroutine the calculation of those components of the function F(x) subgradient which are known analytically.

With every call of the subroutine CALCFG, it receives the vector x value and returns the values of the function F(x) and of some components of its subgradient as a result. The remaining components are approximated by the finite differences.

The subroutine CALCFG may look as follows:

```
$LARGE
C  THE SUBROUTINE CALCFG (DOUBLE PRECISION) CALCULATES THE F(x)
C   VALUES OF THE OBJECTIVE FUNCTION AND THE CORRESPONDING
C  COMPONENTS OF ITS SUBGRADIENT
C     PARAMETERS
C         MODE (INPUT) - FLAG OF THE SUBGRADIENT COMPONENTS
C      CALCULATION MODE:
C         MODE = 0 - ONLY THE FUNCTION F(x) VALUE IS CALCULATED;
C         THE SUBGRADIENT IS NOT CALCULATED;
C         MODE = 2 - BOTH THE FUNCTION F(x) AND ITS SUBGRADIENT
C         COMPONENTS ARE CALCULATED
C      N  (INPUT)  - THE VECTOR X DIMENSION;
C      X  (INPUT)  - THE VECTOR X;
C      F  (OUTPUT) - THE F(x) VALUE;
```

```
C      G  (OUTPUT) - THE SUBGRADIENT;
C  NSTATE (INPUT) - INFORMATION ABOUT CALLS OF THE SUBROUTINE:
C       NSTATE  = 0 - THE CALL OF THE SUBROUTINE IS NEITHER THE
C                     FIRST NOR THE LAST;
C       NSTATE  = 1 - THE CALL OF THE SUBROUTINE IS THE FIRST ONE
C       NSTATE  = 2 - THE CALL OF THE SUBROUTINE IS THE LAST ONE
C  CODE (OUTPUT) - THE SUBROUTINE RETURN CODE:
C                  0 - NORMAL TERMINATION;
C     OTHER THAN 0 - ABEND TERMINATION.
C
      SUBROUTINE          CALCFG( MODE, N, X, F, G, NSTATE, CODE )
      IMPLICIT            REAL*8(A-H,O-Z)
      INTEGER             MODE, N, NSTATE, LW, CODE
      REAL*8              F, X(N), G(N)
      IF  (NSTATE .EQ. 1) GO TO 10
      IF  (NSTATE .EQ. 2) GO TO 20
                          GO TO 30
 10     <Actions during the first call of the subroutine CALCFG>
                          GO TO 30
      IF  (NSTATE .NE. 2) GO TO 20
 20     <Actions during the last call of the subroutine CALCFG>
 30     CONTINUE
C       THE F(x) VALUE CALCULATION
      . . .
        F= <        ...       >
C       THE SUBGRADIENT COMPONENTS CALCULATION
       IF    (MODE .EQ. 0) GO TO 50
       DO    45,I=1,N
             G(I) = ...
 45     CONTINUE
C
 50     CONTINUE
        <assigning the return code>
        RETURN
        END
```

Between subsequent calls of the subroutine CALCFG, the

values of the variable F and of the array G are not saved.

The correspondence between the names of the variables x_i and the numbers I of the entries X(I) is determined as follows. The sequence of the column names in the section COLUMNS of the MPS file determines the correspondence between the name of the variable x_i (i.e. the column name of T(w) and A) and the subroutine's index I of the vector X and of the stochastic subgradient.

Example of the subroutine CALCFG:

```
$LARGE
      SUBROUTINE CALCFG (MODE, N, X, F, G, NSTATE, CODE)
      IMPLICIT          REAL*8(A-H,O-Z)
      INTEGER           MODE, N, NSTATE, LW, CODE
      REAL*8            F, X(N), G(N)
C     N=3
C     F(x) = SUMi( i*x*x/2)
C       THE F(x) VALUE CALCULATION
      F = 0.
      DO 11 I=1,N
 11        F = F + I*X(I)*X(I)/2.
C         THE SUBGRADIENT COMPONENTS CALCULATION
            G(1) = 1.*X(1)
            G(2) = 2.*X(2)
C         THE THIRD SUBGRADIENT COMPONENT IS CALCULATED WITHIN
C         THE APPLICATION PROGRAM (FINITE DIFFERENCES)
      CODE = 0
      RETURN
      END
```

6. SYSTEM SETUP

The SNLP comes on the distribution floppy disks (Table 5). The list of the SNLP files and their purpose are shown in Tables 6 and 7.

The SNLP is installed by means of the SETUP routine. To execute it insert the CONV disk and type the command:

SETUP .

The installation consists of selecting a set of required files and copying them to working floppy disks or a hard disk. If a hard disk is used, all the SNLP files should be allocated in the same directory.

If you consider this version of the SNLP excessive, or the file allocation on the disks impractical, you may delete some files later on, or relocate them to working disks.

7. HOW TO RUN THE SNLP

7.1. SNLP Functions

For solving problems, the following SNLP functions may be used:

- creating and modifying linear deterministic source data by means of the ME subsystem;
- creating and modifying the subroutine CALCFG (nonlinear components for the optimization problem);
- solving linear and nonlinear problems by means of the deterministic optimization subsystem;
- converting a deterministic problem to its stochastic analog by means of the CONVDS subsystem;

Table 5

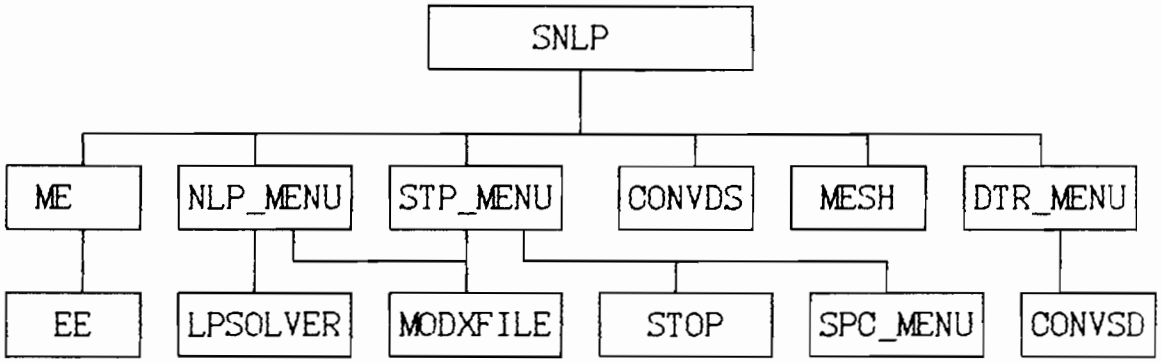
Contents of the Distribution Floppy Disks

Disk Name	Contents	Purpose
MENU	SNLP.EXE	Various menus, editors and auxili-
	SNLP.HLP	ary files
	NLP_MENU.EXE	
	STP_MENU.EXE	
	SPC_MENU.EXE	
	MODXFILE.EXE	
	PATTERN.0	
ME	ME.EXE	Matrix editors
	MESH.EXE	
	EE.EXE	
	EED.HLP	
CONV	CONVDS.EXE	Convertors
	DTR_MENU.EXE	
	CONVSD.EXE	
	SETUP.EXE	
LP	LPSOLVER.EXE	LP solver
STOP	STOP.EXE	Stochastic solver
DLIB	NLP.LIB	Deterministic optimization library
	NLP.OBJ	
SLIB	STOP.LIB	Stochastic optimization library
	STOP.OBJ	
	CONVSD.OBJ	

Table 6

.EXE Files of SNLP

Name	Purpose
SETUP	Setup of the system
SNLP	SNLP main menu
ME	.MPS files martix editor
NLP_MENU	Menu for deterministic optimization
CONVDS	Deterministic-to-stochastic problem convertor
MESH	Stochastic files matrix editor
STP_MENU	Menu for stochastic optimization
SPC_MENU	.SPC files editor
DTR_MENU	Menu for stochastic-to-deterministic problem conversion
CONVSD	Stochastic-to-deterministic problem convertor
MODXFILE	.X files editor
EE	Text editor
LP_SOLVER	LP solver
STOP	Stochastic solver



Hierarchial Structure of the SNLP

Fig. 1

Table 7

SNLP Files

Name	Purpose
SNLP.HLP	Help file
PATTERN.0	Gage for the subroutines CALCFG and CALCTH
EED.HLP	Help file for the EE.EXE
NLP.LIB	Deterministic optimization library
NLP.OBJ	Main subroutine for deterministic solver
STOP.LIB	Stochastic optimization library
STOP.OBJ	Main subroutine for stochastic solver
CONVSD.OBJ	Menu for stochastic-to-deterministic problem convertor

- creating and modifying linear stochastic components for a stochastic problem by means of the MESH subsystem;

- creating and modifying the subroutine CALCTH (simulation of different distributions);

- solving stochastic optimization problems by means of the stochastic optimization subsystem;

- creating the deterministic analog of a stochastic problem by means of the CONVSD subsystem.

7.2. Starting the System

You may access the SNLP either through its main menu (SNLP.EXE file) or through its any other .EXE file. All the system .EXE files are started by means of the command:

< .EXE file > [< problem specification >] .

List of the system .EXE files and their functions is shown in Table 6. The SNLP has a hierarchial structure shown in Figure 1.

The problem specification is the full name of the problem, allowing to identify it and all the related files. The problem specification is the specification of its files without extensions. On the contrary, the file specification is the problem specification with one of the feasible extensions.

The problem specification is as follows:

[< disk >:] [< path >] [< problem name >] ,

eg. C:\SNLP\PROBLEM1 , A:\STPR2 , LINEAR.

If you specify only the problem name without specifying the disk and directory, then the current disk and directory values are assumed. If the problem specification is not defined, the

system issues a query.

It is advisable to specify the disk and directory for the SNLP in the PATH command of the AUTOEXEC.BAT file.

7.3. Menu Manipulations

You may access the SNLP system .EXE files either via the menu or from MS/DOS. You may select the menu item using arrows, as well as PgUp, PgDn, HOME and END keys and then pressing ENTER. To exit the menu, select the "EXIT" item or press ESC.

7.4. Main Menu

To access the main menu, call the SNLP.EXE file. As a result two windows are displayed (Fig. 2).

The lower window contains the main menu. The upper one contains the specification of the problem being solved. If the call of the SNLP didn't contain the problem specification, the PATH field shows the current path, and the NAME field is empty.

7.5. Setting the Problem Specification

You may specify or change the name of the problem being solved by selecting the main menu item:

"Specify problem".

As a result the following view is displayed (Fig. 3). The PATH field contains the current disk and directory names, and the NAME field contains the problem name.

To specify the PATH field you should move the cursor there, and then enter a new name.

The problem name in the NAME field may be entered or changed in two ways. The first way is to move the cursor to this field

```
Current Problem
 PATH:  C:\SNLP
 NAME:
```

```
Specify         Problem
Create/Modify Deterministic Data (.mps file)
Solve           Deterministic Problem
Transfer Data From Deterministic To Stochastic Problem
Create/Modify Stochastic    Data ( .sth files)
Solve           Stochastic    Problem
Transfer Data From Stochastic To Deterministic Problem
View/Modify     Files
DOS             Access
Exit
```

Select Menu Item And Press ENTER

SNLP Main Menu

Fig. 2

```
Current Problem
 PATH:  C:\SNLP
 NAME:
```

F1-HELP F2-LIST OF PROBLEMS ESC-EXIT

Setting The Problem Specification

Fig. 3

Name	.SPC	.MPS	.STH	.X	.PRT	.FOR	.OBJ	.EXE
TEST	Y	Y	Y	Y	Y	Y	Y	
TST2	Y	Y	Y	Y				
TST3	Y	Y	Y	Y				
TSTD3	Y	Y		Y				
TSTS3	Y	Y	Y	Y				
MANUF	Y	Y	Y	Y				
TSTD2		Y		Y				
QP		Y		Y		Y		
LP		Y		Y	Y			

Select Problem And Press ENTER ESC-QUIT

List of Problems and Their Files

Fig. 4

and enter a new name. The second way is to press F2, and the list of the problems with the related files is displayed, as shown, for instance, in Fig. 4. You may specify the problem name by selecting any name from this list.

7.6. Creating and Modifying Linear Deterministic Source Data

You may create, view and modify the source data of LP problems, as well as linear deterministic components of NLP and SP problems by means of the ME matrix editor.

To start the ME, select the main menu item:

"Create/Modify Deterministic Data (.MPS file)" ,

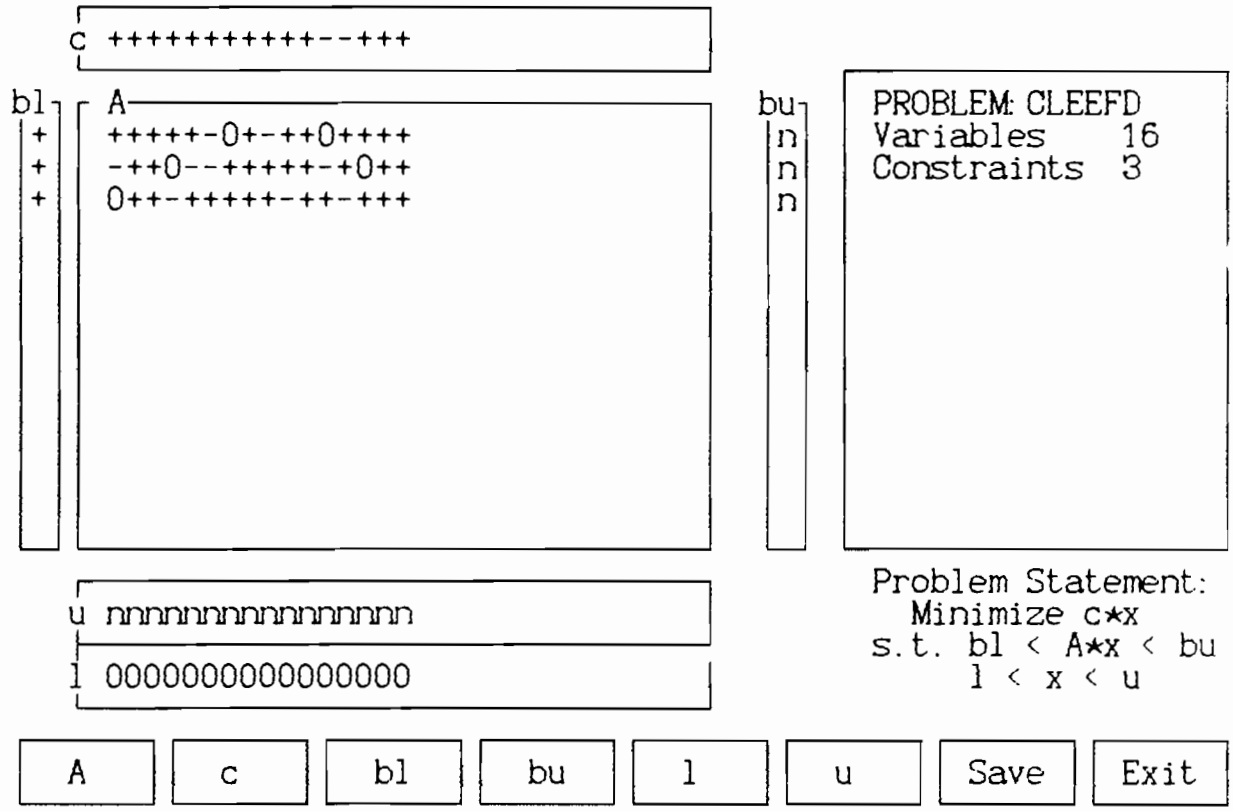
or type the DOS command:

ME [< problem specification >] .

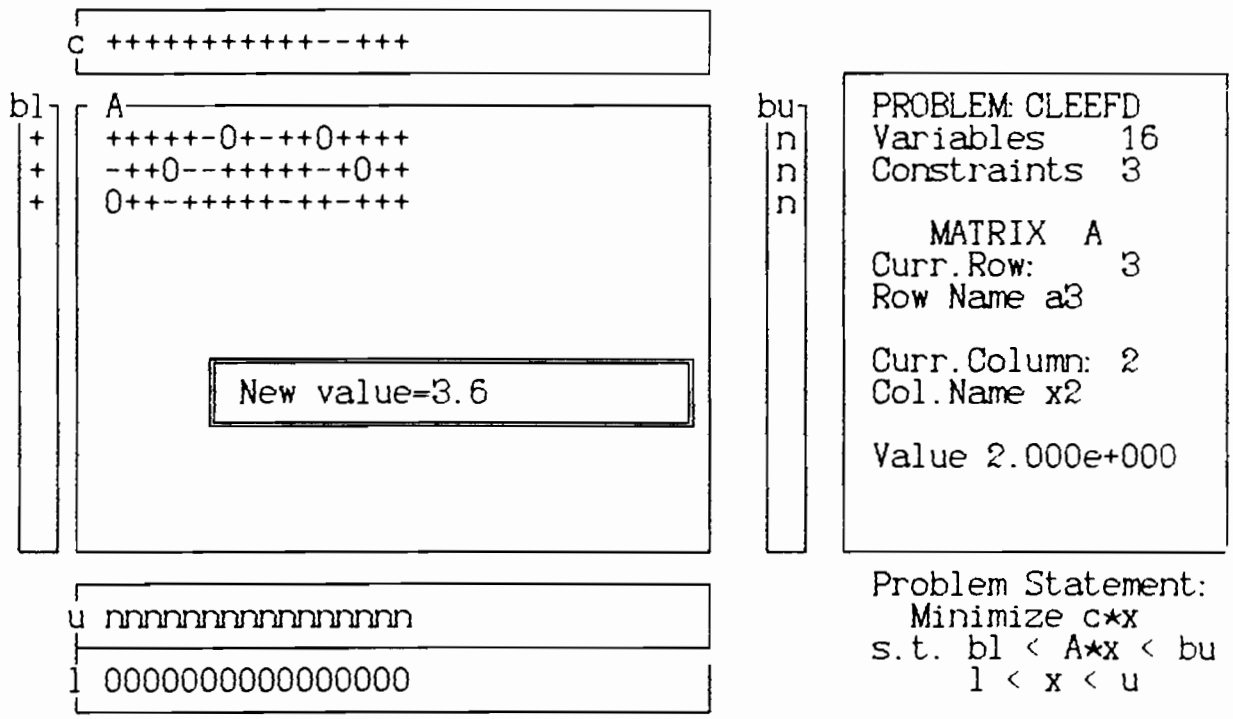
The ME executes the following functions:

- adding a variable (the matrix A column);
- adding a constraint (the matrix A row);
- deleting a variable;
- deleting a constraint;
- renaming a variable;
- renaming a constraint;
- changing the entry values of the matrix A and the vectors c, l, u, b_l, b_u (here b_l, b_u are bounds in linear constraints, and l, u are bounds on the variables);
- sampling statistics about the problem.

After specifying the problem name and reading the data, seven windows and a menu are displayed (Fig. 5). Six windows at the left side of the screen contain the matrix representation of the problem. Each of these windows contains information about



ME Subsystem Menu
Fig. 5



F1-Help ENTER-Change Value F2-Insert Row F3-Delete Row
F4-Insert Column F5-Delete Column F6-Append Row F7-Append Column
R-Rename Row C-Rename Column ESC-Main Menu F8-Statistics

Tackling the Matrix A
Fig. 6

one of the following problem parameters: the vectors b_l, b_u, l, u, c, and the matrix A. The vector and matrix entries are represented in the windows by means of the following characters:

0 - corresponding to a zero entry;

+ - corresponding to a positive entry;

- - corresponding to a negative entry;

n - corresponding to an absent (infinite) bound.

The window at the right side of the screen contains the current information about the object being viewed and about the value of its current entry.

You may view and change the problem entries as follows. Select one of the objects, and jump into the corresponding window. Then move the cursor from one entry to another using arrows. As a result, information about the current entry is displayed. For the matrix A, for instance, it is the number and name of the current entry in row and column, as well as its value (Fig. 6).

While tackling the matrix A, use the following keys in addition to arrows:

- HOME - to jump to the first row of the matrix;
- END - to jump to the last row of the matrix;
- CTRL+LEFT - to jump to the left over 10 columns;
- CTRL+RIGHT - to jump to the right over 10 columns;
- F1 - to display information about available commands;
- F2 - to insert a new row;
- F3 - to delete a row;
- F4 - to insert a new column;

- F5 - to delete a column;
- F6 - to append a row;
- F7 - to append a column;
- F8 - to give statistics about the problem;
- ENTER - to change the current entry value;
- C - to rename the current column;
- R - to rename the current row.

While tackling the vector l use in addition the following keys:

- F9 - to initialize all the vector entries by zeroes;
- F10 - to initialize all the vector entries by "minus infinity" (-∞).

While tackling the vector c, use in addition the F9 key to reverse the sign of all the vector entries.

To save the data in the MPS file, move the cursor to the "SAVE" position in the top-level ME menu and press ENTER.

7.7. Deterministic Optimization Subsystem

7.7.1. General Information about the Subsystem

The subsystem is intended to solve LP and NLP problems with linear constraints.

To call the subsystem, select the SNLP main menu item:

"Solve Deterministic Problem"

or type the DOS command:

NLP_MENU [< problem specification >] .

Right after the startup the subsystem prompts:

LINEAR or NONLINEAR .

Your reply motivates jumping to one of the two menus (Fig. 7 and

LINEAR DETERMINISTIC OPTIMIZATION

```
Set Up      X_Initial      ( .x file)
Start       Optimization   Subsystem
View        Solution       ( .prt file)
Exit
```

Select Item and Press ENTER

Menu for Solving Linear Programming Problems

Fig . 7

NONLINEAR DETERMINISTIC OPTIMIZATION

```
User's   Subroutine     (CALCFG in .for file)
Set Up   X_Initial      ( .x file)
Start    Optimization   Subsystem
View     Solution       ( .prt file)
Exit
```

Select Item and Press ENTER

Menu for Solving Nonlinear Programming Problems

Fig. 8

```
                USER's SUBROUTINE SERVICE
  To EDIT the user's subroutine (y/n) ?          [N]
U To COMPILE the user's subroutine (y/n) ?       [N]
S To LINK and CREATE the problem solver (y/n) ? [N]
S
View     Solution       ( .prt file)
Exit
```

Reply: Y/N Press ENTER to Confirm

Menu for Creating the CALCFG Subroutine and the Solver

Fig. 9

Fig. 8). The first one is for solving linear problems by means of the LPSOLVER. The second menu is for solving nonlinear problems. To solve an NLP problem, the user should create his CALCFG subroutine in the .FOR file, compile it, and then link it together with the NLP.LIB library. The obtained .EXE file should have the name

< problem name >.EXE .

It is actually the solver of the given NLP problem.

7.7.2. Creating the CALCFG Subroutine and the NLP Solver

You may create your CALCFG subroutine in the .FOR file and then also create the NLP solver by means of the NLP_MENU.EXE file by selecting the menu item:

"USER's Subroutine (CALCFG in .FOR file)" .

As a result, three queries are displayed (Fig. 9). Replies to these queries enable a step-by-step execution of the following actions:

- viewing, creating and modifying the .FOR file containing the CALCFG subroutine;
- compiling the .FOR file containing the CALCFG subroutine;
- linking together the solver's subroutine (NLP.LIB file) and the subroutine CALCFG (.OBJ file) to create the .EXE file of the NLP solver;

To view, create, and modify the .FOR file, reply "y" to the query:

"To Edit the user's subroutine (y/n)?"

The .FOR file may contain not only the CALCFG subroutine

itself, but all its auxiliary subroutines as well.

Further actions are defined by the state of the screen which contains two windows and a menu (Fig.10).

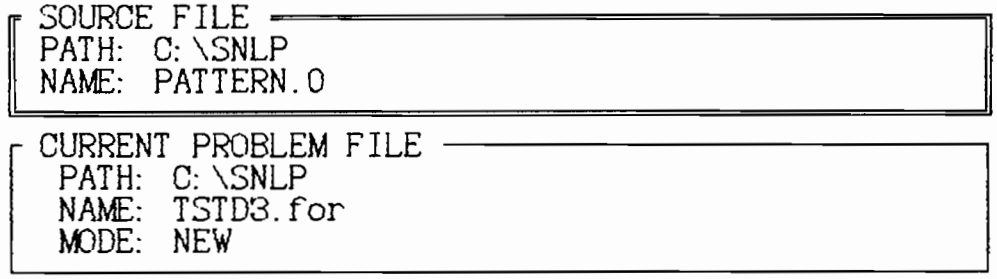
The upper window ("source file") contains the specification of the file used to create the .FOR file of the current problem. The lower one contains the .FOR file specification for the current problem. This file will be used later while creating the solver.

If the .FOR file of the current problem already exists, the contents of both windows are the same. If this file is created anew, the upper window contains the specification of the PATTERN.O file, which contains the gage for the CALCFG and CALCTH subroutines.

You may change the "source file" specification in the upper window in two ways. The first way is to jump in turns into the PATH and NAME fields and to enter the new disk, directory and file names into them.

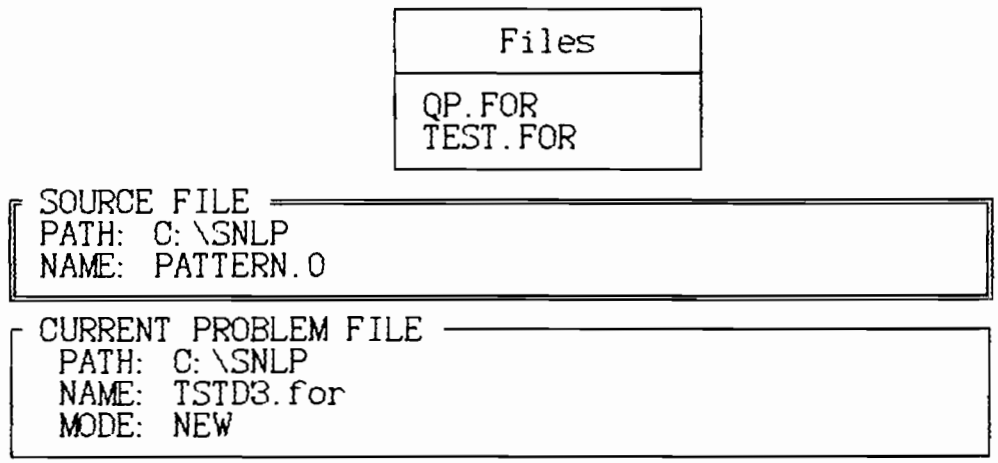
The second way is to enter the required directory name into the PATH field of the upper window and press F2. After that a list of the .FOR files, similar to that shown on Fig. 11, is displayed. Then you can select the "source file" you need.

Why should you use "alien" .FOR files? First, the current problem may be a modification of another problem, and the current subroutine CALCFG is created based upon an already existing subroutine CALCFG. Second, if the subroutine CALCFG for the stochastic analog of your deterministic problem was already created, then it must be the same for both problems.



Menu for Chosing an "Alien" .FOR File

Fig. 10



Chosing an "Alien" .FOR File from the List

Fig. 11

Variables Numbers	Variables Names	Lower Bounds	Variables Values	Upper Bounds
1	COL1	0.000E+000	2.333E+003	+ 0.100E+0
2	COL2	0.000E+000	2.000E+000	+ 0.100E+0

The .X File Editor

Fig. 12

Now that the "source file" name is specified, you may proceed to creating or modifying the .FOR file of the current problem.

This may be done in two ways. The first way consists simply of copying the "source file" without any modifications. To do this, press F3.

The second way is as follows. Call the specialized EE.EXE text editor by pressing ENTER. If the .FOR file is created anew, the PATTERN.0 gage file becomes the source file. After that you may modify and save this file by means of the EE text editor.

To compile the .FOR file, reply "y" to the query:

"To COMPILE the user's subroutine (y/n)?"

As a result the FORTRAN Microsoft 4.01 compiler is called. To do this successfully, install FORTRAN Microsoft 4.01 before using the SNLP as described in [15]. You must specify the correct environment for the FORTRAN Microsoft 4.01, otherwise MS/DOS will not be able to find the files of the compiler and its library.

Brief information about the environment for the FORTRAN Microsoft 4.01.

The following should be specified in the CONFIG.SYS file:

```
files   = 20
buffers = 10        ;
```

The path for the compiler should be specified in the PATH command in the AUTOEXEC.BAT file, for instance,

```
PATH = C:\MSF\BIN ;
```

Directories should be created: for the compiler (eg.

C:\MSF\BIN), for the libraries (eg. C:\MSF\LIB) and for the temporary files (eg. C:\MSF\TMP);

Environment variables LIB and TMP for the library and temporary files should be specified in the AUTOEXEC.BAT file, for instance,

```
SET LIB = C:\MSF\LIB
SET TMP = C:\MSF\TMP ;
```

In the library directory you should have the LLIBFORE.LIB library.

You may call the FORTRAN Microsoft compiler with the following options:

/AL - "large" compiler model (and with the $LARGE metacommand);

/FS - the .LST file name;

/FO - the .OBJ file name;

/c - compilation only (without linkage);

/FPc- using the emulation mode for arithmetic operations.

If you compile the .FOR file out of the SNLP, you may also specify the additional options of the compiler.

To link the .OBJ file together with the SNLP library NLP.LIB and to create the .EXE file for solving an NLP problem, reply "y" to the query:

"To LINK and CREATE the problem solver (y/n)?"

The LINK Microsoft linkage editor is called as a result. The LINK.EXE file should be present in the FORTRAN compiler directory (C:\MSF\BIN in the above example). It is advisable to use the LINK Microsoft version 3.51 and higher.

For a successful linkage, the NLP.LIB library should be present in the same directory as the .EXE files of the SNLP. No special options are specified for the linkage editor. The linkage may be executed out of the SNLP as well.

7.7.3. Creating the Initial Point File

You may create and modify the .X file containing the x^0 initial point for the algorithm for solving an NLP problem by means of the MODXFILE.EXE X-file editor.

This editor may be called from the SNLP.EXE by selecting the following item from the NLP or LP problems menu:

"Set Up X_Initial (.X file)"

or by typing the command:

MODXFILE [<problem specification>].

The X-file editor displays two windows and a menu, similar to those described for the .FOR files.

The upper window contains the specification of an "alien" .X file used for creating the x^0 initial point for the current problem.

If the .X file of the current problem already exists, the contents of the windows are the same. But if it is created anew, the NAME field in the upper window is empty.

You may change the "source file" specification in the upper window in two ways. The first one is to jump in turns into the PATH and NAME fields and to enter the new disk, directory and file names into them.

The second way is to enter the required directory name into

the PATH field of the upper window and press F2. After that a list of the .X files, similar to that shown for .FOR files in Fig. 11, is displayed. Then you can select the "source file" you need.

Why should you use the "alien" .X files? Certainly, it is natural to use your own .X file containing the results of the previous calculations to create the initial point x^0. But nevertheless there are at least three reasons to take the .X file from another problem and use it as the initial point of the current problem.

First, the current problem may be similar to another problem, and solutions of both problems may be roughly equal. Thus the solution x^* of one problem may be a good initial point x^0 for another one.

Second, if we have already solved a stochastic problem, we may use the obtained solution as the initial point for its deterministic analog.

Third, if you have no idea about the "good" point x^0, then you may quickly solve the linear analog of the current nonlinear problem (without the function F(x)), and then use the obtained solution as the initial point for this NLP problem. The linear analog of a nonlinear problem is created in a very simple way. All you have to do is to select LINEAR while initializing the deterministic optimization subsystem. Then the linear problem solver (LPSOLVER) is used without calling the CALCFG.

Now you may proceed to creating or modifying the .X file of the current problem.

This may be done in two ways. The first way consists simply of copying the "source file" without any modifications. To do it press F3.

The second way is as follows. Call the specialized X-file editor by pressing ENTER. If the .X file is created anew, it is initialized to 0. Then you may modify and save this file by means of the X-file editor. Naturally the source and the current problems may be the same.

You should keep in mind that to create the .X file you must keep ready the .MPS file for this problem. The names of variables x_i in the .X file of the source problem must correspond to the column names in the .MPS file for the current problem.

The X-file editor displays a table similar to the one shown in Fig. 12. Each line of the table corresponds to one variable x_i. The X-file editor can only change the fields of the values. To exercise it you should move the cursor to the required line and enter a new value for the variable.

To exit the X-file editor, press ESC.

It is possible to use .X files of other problems and modify them through of the SNLP as well.

The .X file is not utilized by the LP solver. Nevertheless, intermediate results and the problem solution are written into it.

7.7.4. Solving LP and NLP Problems

After creating the .MPS, .X and .EXE files of the current problem, you may proceed to solving it. The NLP solver has the

name:

<problem name>.EXE ,

and the LP solver has the name:

LPSOLVER.EXE .

These solvers may be started from the SNLP by selecting the menu item:

"Solve the problem"

or by typing the following DOS command:

< .EXE file> [<problem specification>] .

In every algorithm iteration the current x^s point is written into the .X file. The problem solution and diagnostic messages are written into the .PRT file.

7.7.5. Viewing the Solution File

You may view the problem solution saved in the .PRT file by selecting the menu item:

"View Solution (.PRT file)"

in the NLP_MENU.EXE file.

Naturally you may view the .PRT file throughf the SNLP as well.

7.7.6. Exit

To exit the menu (and return from the NLP_MENU.EXE file) select the menu item "Exit".

7.8. Converting Deterministic Problems to Their Stochastic Analogs

Deterministic problems are converted to their stochastic

analogs by means of the CONVDS subsystem called the "stochastic convertor". This subsystem is intended to create the .MPS file and the draft .STH file of the stochastic problem from the .MPS file of the source deterministic problem. The .MPS file is intended for linear deterministic components of the stochastic problem, and the draft .STH file is intended for "stochastic" components present in the recourse function Q(x). The draft .STH file contains only those numeric values which may be taken from the .MPS file of the source deterministic problem. All the indefinite fields are marked by asterisks ("*").

To start the CONVDS, select the main menu item:

"Transfer Data from Deterministic to Stochastic Problem" ,

or type the DOS command:

CONVDS [<specification of stochastic problem being created>].

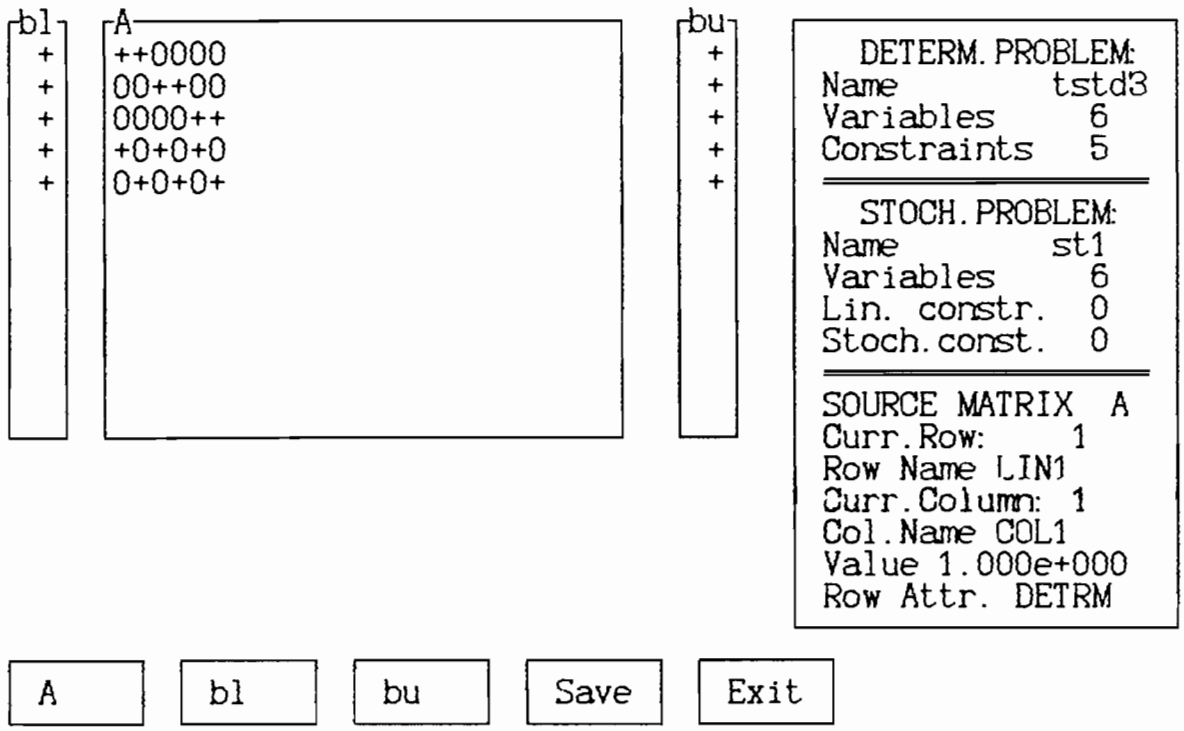
The subsystem CONVDS executes the following functions:

- viewing the constraints of the source problem;

- assigning "stochastic" (STOCH) or "deterministic" (DETERM) attributes to the matrix A rows and the bounds b_l , b_u entries of the source problem;

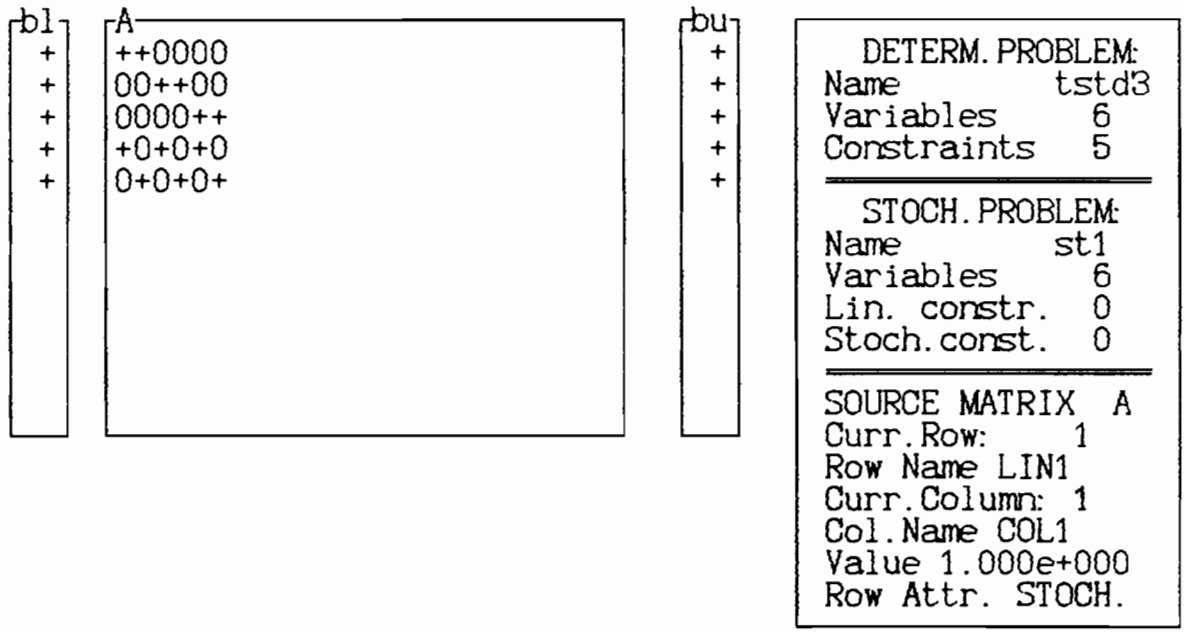
- saving the .MPS and .STH files of the created stochastic problem.

After reading the source data, four windows and a menu are displayed (Fig. 13). The three windows at the left side of the screen contain the matrix representation of the linear constraints. The window at the right side of the screen contains information about the source problem, the problem being created, and the current entry values being viewed.



The CONVDS Subsystem Menu

Fig. 13



Operating the Matrix A

Fig. 14

You may view these entries in the same way as in the ME (Fig. 14). In addition, the following keys are used:

- S - to assign the "stochastic" attribute;
- D - to assign the "deterministic" attribute.

7.9. Creating and Modifying Linear Stochastic Source Data

You may create, view and modify linear stochastic components of a stochastic programming problem by means of the MESH matrix editor.

To start the MESH select the main menu item:

"Create/Modify Stochastic Data (.STH file)" ,

or type the DOS command:

MESH [<problem specification>] .

One of the following three cases may occur when starting the MESH editor:

- the problem does not exist, and its .STH file is absent;
- the problem is being created (either right after the CONVDS stochastic convertor operation, or after the preceeding operating session of the MESH), and the draft .STH file contains indefinite fields marked by asterisks ("*");
- the problem exists, and its .STH file is ready for use.

In all these cases the .MPS file must be ready to use and present in the same directory as the .STH file. You should keep in mind that the names of the matrix T(w) columns in the .STH file are defined in the .MPS file.

The MESH executes the following functions:

- adding a variable (the T(w) matrix column);
- adding a constraint (the T(w) matrix row);

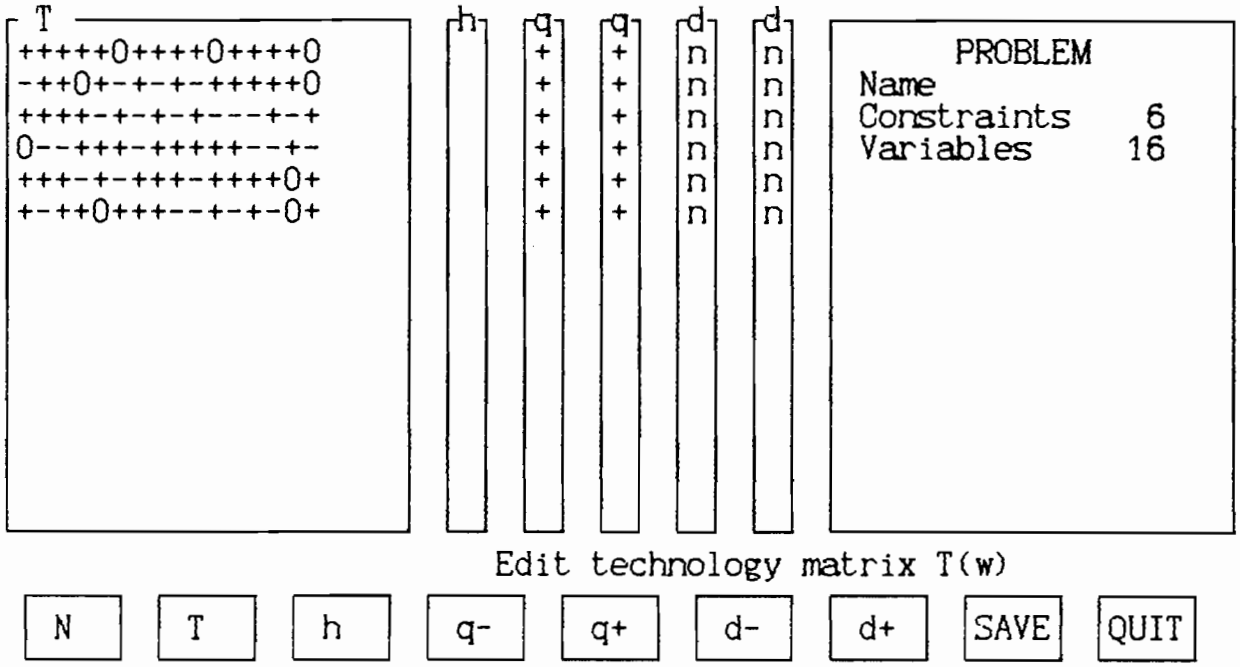
- deleting a variable;
- deleting a constraint;
- renaming a variable;
- renaming a constraint;
- changing the entry values of the matrix T(w) and of the vectors q^-, q^+, d^-, d^+, h(w);
- specifying the distribution type for the h(w) random entries;
- giving statistics about the problem.

You should keep in mind that after deleting, adding or renaming the T(w) matrix column the user is responsible for setting up the correspondence between the column names of the .MPS file and those of the .STH file. To correctly use this modified .STH file later on, you should execute similar modifications in the corresponding .MPS file.

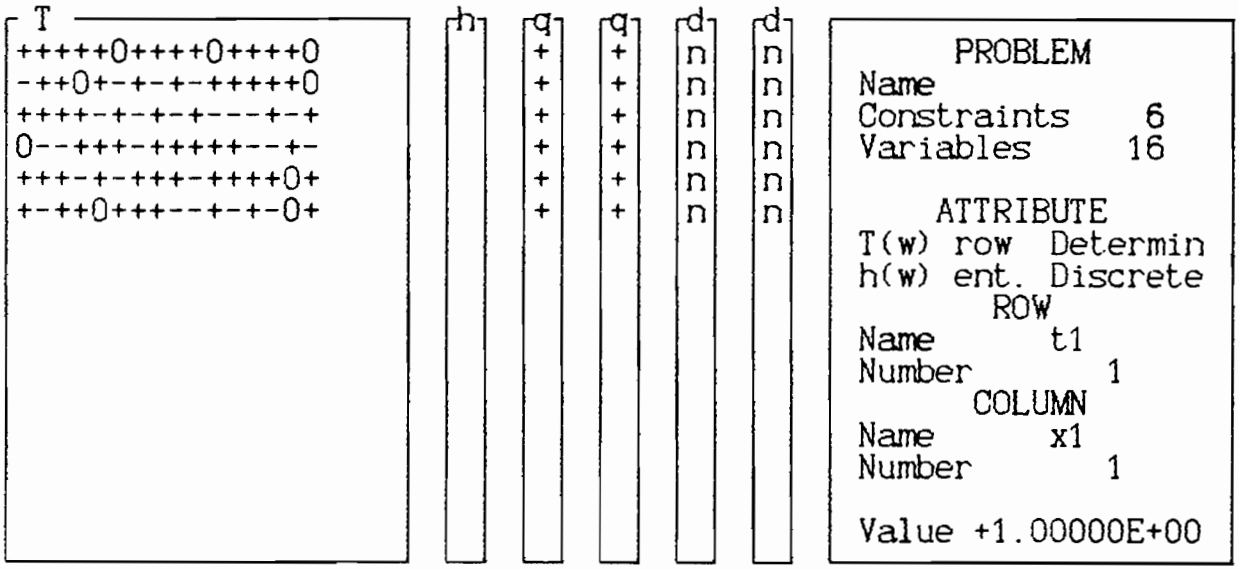
After entering the source data from the .STH file, seven windows and a menu are displayed (Fig. 15). The six windows at the left side of the screen contain the matrix representation of the stochastic constraints. Each of these windows contains information about one of the problem parameters.

The window at the right side of the screen displays information about the object being viewed and its current entry value.

You may view and change the current entries in the same way as in the ME (Fig. 16). While tackling q^-, q^+ use in addition the F9 key to initialize all the vector entries to zeroes. While tackling d^-, d^+ use in addition the F10 key to initialize all



Operating the Matrix T(w)
Fig. 15



Operating Stochastic Components of the Problem
Fig. 16

the vector entries to "infinity" (+∞).

While tackling the vector h(w), you may enter, by pressing the "A" key, a special mode in which you may display and change the distribution type of random values as well as their parameters.

7.10. Stochastic Optimization Subsystem

7.10.1. General Information about the Subsystem

The subsystem is intended for solving stochastic programming problems with recourse.

To start the subsystem select the SNLP main menu item:

"Solve Stochastic Problem" ,

or type the DOS command:

STP_MENU [<problem specification>] .

Right after the startup the subsystem prompts:

"To Support Connection With User's Subroutine (y/n)?"

Your reply motivates jumping to one of the two menus (Fig. 17 and Fig. 18). In the first one, the problem is solved by the STOP.EXE solver without the subroutines CALCFG and CALCTH. The second menu contains one more line for creating these subroutines in the .FOR file, compiling them and then linking them together with the STOP.LIB library. The obtained solver should have the name:

<problem name>.EXE .

7.10.2. Creating the Subroutines CALCFG and CALCTH and the Stochastic Solver

The user's subroutines in the .FOR file and the problem solver

STOCHASTIC OPTIMIZATION WITH RECOURSE

```
User's       Subroutines  (CALCFG and CALCTH in .for file)
Algorithm    Parameters Specification ( .spc file)
Set Up       X_Initial       ( .x file)
Stochastic   Optimization (STOP subsystem)
View         Solution        ( .prt file)
Exit
```

Select Item and Press ENTER

Press ESC to exit

Stochastic Optimization Subsystem Menu
(with linking the CALCFG and CALCTH subroutines)

Fig. 17

STOCHASTIC OPTIMIZATION WITH RECOURSE

```
Algorithm    Parameters Specification   ( .spc file)
Set Up       X_Initial       ( .x file)
Stochastic   Optimization (STOP subsystem)
View         Solution        ( .prt file)
Exit
```

Select Item and Press ENTER

Press ESC to exit

Stochastic Optimization Subsystem Menu
(without linking the CALCFG and CALCTH subroutines)

Fig. 18

ALGORITHM PARAMETERS SPECIFICATION

OLD Specs-file

```
     initial      step                value          1
     algorithm    iterations          limit          50
     output       level          for  solution       2
Al   required     accuracy       of   solution       0.001
Se   feasible     tolerance      for  constraints    1E-006
St   averaging    estimation          parameter      1
Vi   subgradient  approximation       parameter      0.001
Ex
```

Select Item to Modify and Press ENTER   Press ESC to exit

Algorithm Parameter Specification

Fig. 19

are created by the STP_MENU.EXE file in the same way as in the deterministic optimization subsystem. For a successful linkage, the STOP.LIB library should be present in the same directory as the SNLP.

7.10.3. Creating the Initial Point File

The .X file is created in the same way as in the deterministic optimization subsystem. You may use as the initial point of the current stochastic problem either the solution point of its linear stochastic analog (without the CALCFG subroutine), the solution point of its deterministic analog, or the solution point of a similar problem.

7.10.4. Assigning Values to the Algorithm Parameters

To assign values to the optimization algorithm parameters, select the menu item:

"Algorithm Parameters Specification (.spc file)"

in the SPC_MENU.EXE file, or type the DOS command:

SPC_MENU [<problem specification>].

As a result a window is displayed, each line of which contains the description of a parameter and its current value (Fig. 19).

The algorithm parameters are stored in the .SPC file. If the .SPC file is created anew, the algorithm parameters accept the default values. To change the parameter values, select the line you need and press ENTER. Then the cursor jumps into the value field, and a description window is displayed at the bottom of the screen. After entering the new parameter value, you may

proceed to changing another parameter.

Press ESC to save the parameter values in the .SPC file and to quit.

7.10.5. Solving Stochastic Programming Problems

After creating the .MPS, .STH, .X, .SPC and .EXE files of the current problem, you may proceed to solving it. The stochastic solver containing the subroutines CALCFG and CALCTH has the name:

<problem name>.EXE ,

and the solver without the subroutines has the name:

STOP.EXE .

These solvers may be started from the SNLP by selecting the menu item:

"Solve the problem" ,

or by typing the following DOS command:

< .EXE file> [<problem specification>] .

In every algorithm iteration the current x^s point is written into the .X file. The problem solution and diagnostic messages are written into the .PRT file.

While in output level 4 the system displays the graph whose X axis represents the algorithm iteration numbers k = 1, 2,..., and Y axis represents the values of the objective function evaluation $F_k(x^k, w^k)$ in these iterations.

You may use the following keys to change the graph and to control the algorithm:

S - change the current step value;

R - clear the screen and redraw the graph with automatic

rescaling on the Y axis;

* - multiply the scale factor on the Y axis by 1.5;

/ - divide the scale factor on the Y axis by 1.5;

Q - quit the optimizer.

7.10.6. Viewing the Solution File

You may view the problem solution saved in the .PRT file by selecting the menu item:

"View Solution (.PRT file)"

in the STP_MENU.EXE file.

You may view the .PRT file out of the SNLP as well.

7.10.7. Exit

To exit the menu (and return from the STP_MENU.EXE file) select the menu item "Exit".

7.11. Converting Stochastic Problems to Their Deterministic Analogs

Problems are converted by means of the subsystem consisting of two files: DTR_MENU.EXE and CONVSD.EXE. The subsystem is called a "deterministic convertor". This convertor is intended to create the .MPS file of the deterministic problem from the .MPS and .STH files of the source stochastic problem. This .MPS file contains the linear constraints (2) produced from the linear constraints of the stochastic analog (5) and stochastic constraints produced by averaging the h(w) and T(w) entries, as in (12a).

To start the subsystem select the main menu item:

"Transfer Data from Stochastic to Deterministic Problem",

or type the DOS command:

CONVSD [<deterministic problem specification>]

or

DTR_MENU [<deterministic problem specification>]

The DTR_MENU.EXE file is intended to create and call the problem convertor, and the CONVSD.EXE file is the convertor without the subroutine CALCTH.

Right after starting the DTR_MENU.EXE file, the following query is displayed:

"To Support Connection With User's Subroutine CALCTH (y/n)?"

Your reply motivates jumping to one of the two menus (Fig. 20 and Fig. 21). In the first one, the problem is converted by the CONVSD.EXE file without the subroutine CALCTH. The second menu contains one more line for creating the subroutine CALCTH in the FOR file, compiling it and then linking it together with the STOP.LIB library. The obtained convertor should have the name:

<deterministic problem name>.EXE

You should keep in mind that the required .FOR file with the CALCTH subroutine may be created simply by copying the .FOR file for the problem exposed to determinization. The .FOR file compilation and linkage are executed in the same way as for creating the stochastic problem solvers.

To call the problem convertor, select the menu item:

"Transfer Stochastic to Deterministic Problem (.MPS file)",

or type the DOS command:

< .EXE file> [<problem name>]

7.12. Viewing and Modifying Files

STOCHASTIC-TO-DETERMINISTIC CONVERSION

```
Transfer Stochastic To Deterministic Problem ( .mps file)
View     Diagnostics ( .prt file)
Exit
```

Select Item And Press ENTER

Press ESC to exit

Stochastic-to-Deterministic Problem Conversion Subsystem's

Main Menu (without linking the CALCTH subroutine)

Fig. 20

STOCHASTIC-TO-DETERMINISTIC CONVERSION

```
User's   Subroutine  ( CALCTH in  .for file)
Transfer Stochastic To Deterministic Problem ( .mps file)
View     Diagnostics ( .prt file)
Exit
```

Select Item and Press ENTER

Press ESC to exit

Stochastic-to-Deterministic Problem Conversion Subsystem's

Main Menu (with linking the CALCTH subroutine)

Fig. 21

You may view and modify files by means of the EE.EXE text editor. To start this editor select the main menu item:

"View/Modify Files" ,

or type the DOS command:

EE [<file specification>]

7.13. Access to the Operating System

While using the SNLP you may need to execute some actions not supported by the SNLP (eg. renaming files etc). To do it you should exit to DOS by selecting the main menu item:

"DOS Access" .

To return to the SNLP, type the EXIT command.

7.14. Exit from the Main Menu

To return from the SNLP select the main menu item:

"EXIT" ,

or press ESC.

8. ONE EXAMPLE

Let us consider the following deterministic and stochastic problems based on the classic example of a stochastic problem with simple recourse [18], [19].

In the deterministic problem, an airline wishes to allocate airplanes of various types among its routes to satisfy an average passenger demand, in such a way as to minimize operating costs.

Choose x_j ($j = 1,\ldots,17$) to minimize

$$\sum^{17} c_j x_j$$

$$j=1$$

subject to

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq b_1$$
$$x_6 + x_7 + x_8 + x_9 \leq b_2$$
$$x_{10} + x_{11} + x_{12} \leq b_3$$
$$x_{13} + x_{14} + x_{15} + x_{16} + x_{17} \leq b_4$$
$$x_j \geq 0 , \quad j = 1,\ldots,17$$
$$t_1 x_1 + t_{13} x_{13} \leq h_1$$
$$t_2 x_2 + t_6 x_6 + t_{10} x_{10} + t_{14} x_{14} \leq h_2$$
$$t_3 x_3 + t_7 x_7 + t_{15} x_{15} \leq h_3$$
$$t_4 x_4 + t_8 x_8 + t_{11} x_{11} + t_{16} x_{16} \leq h_4$$
$$t_5 x_5 + t_9 x_9 + t_{12} x_{12} + t_{17} x_{17} \leq h_5$$

We may use the SNLP to solve this problem, and we shall find out that it is infeasible. Then we may use the SNLP to convert this deterministic problem to its stochastic analog, which may be expressed in the following way.

In the stochastic problem an airline wishes to allocate airplanes of various types among its routes to satisfy an uncertain passenger demand, in such a way as to minimize operating costs plus the lost revenue from passengers turned away.

Choose x_j $(j = 1,\ldots,17)$ to minimize

$$\sum_{j=1}^{17} c_j x_j + E\left\{ \sum_{k=1}^{5} q_k^+ \quad y_k^+ \right\}$$

subject to

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq b_1$$
$$x_6 + x_7 + x_8 + x_9 \leq b_2$$
$$x_{10} + x_{11} + x_{12} \leq b_3$$

$$x_{13} + x_{14} + x_{15} + x_{16} + x_{17} \leq b_4$$

$$x_j \geq 0 , \quad j = 1,\ldots,17$$

$$y_k^+ \geq 0 , \quad y_k^- \geq 0$$

$$y_1^+ - y_1^- = h_1 - (t_1 x_1 + t_{13} x_{13})$$

$$y_2^+ - y_2^- = h_2 - (t_2 x_2 + t_6 x_6 + t_{10} x_{10} + t_{14} x_{14})$$

$$y_3^+ - y_3^- = h_3 - (t_3 x_3 + t_7 x_7 + t_{15} x_{15})$$

$$y_4^+ - y_4^- = h_4 - (t_4 x_4 + t_8 x_8 + t_{11} x_{11} + t_{16} x_{16})$$

$$y_5^+ - y_5^- = h_5 - (t_5 x_5 + t_9 x_9 + t_{12} x_{12} + t_{17} x_{17})$$

$x_1,\ldots,x_5$: type 1 aircraft assigned to routes 1,...,5;

$x_6,\ldots,x_9$: type 2 aircraft assigned to routes 2,...,5;

x_{10},x_{11},x_{12}: type 3 aircraft assigned to routes 2, 4, 5;

$x_{13},\ldots,x_{17}$: type 4 aircraft assigned to routes 1,...,5;

b_i: number of aircraft available of type i = 1,...,4;

c_j: cost of operating aircraft/route j = 1,...,17;

q_k: revenue lost per passenger turned away on route k = 1,...,5;

y_k^+: empty seats on route k;

y_k^-: passengers turned away on route k;

t_j: passenger capacity on aircraft/route j;

h_k: passenger demand for route k.

Data:

c = [18,21,18,16,10,15,16,14,9,10,9,6,17,16,17,15,10]

q = [13,13,7,7,1]

b = [10,19,25,15]

h_k are discretely distributed as follows:

h_1 ~ [200,220,250,270,300] w.p. (0.2,0.05,0.35,0.2,0.2)

h_2 ~ [50,150] w.p. (0.3,0.7)

$h_3 \sim$ [140,160,180,200,220] w.p. (0.1,0.2,0.4,0.2,0.1)

$h_4 \sim$ [10,50,80,100,340] w.p. (0.2,0.2,0.3,0.2,0.1)

h_5 = [580,600,620] w.p. (0.1,0.8,0.1)

t = [τ_1,τ_2,28,23,81,10,14,15,57,5,7,29,9,11,22,17,55]

$\tau_1 = 21 - \xi_1$

$\tau_2 = 21 - \xi_2$

ξ_1 has Poisson distribution with parameter 5.

ξ_2 has Poisson distribution with parameter 6.

Assume that type 1 aircraft on routes 1, 2 carry, in addition to passengers, some load (eg. mail) with volume of ξ_1 and ξ_2. The load traffic has Poisson distribution.

If we set instead of t_1 and t_2 their average values, the problem will be exactly the same as the one from [18], [19].

Solution:

Calculated to one decimal place accuracy

Aircraft Type	1	2	3	4
Route				
1	$x_1 = 10$	*	*	$x_{13} = 11.9$
2	$x_2 = 0$	$x_6 = 0$	$x_{10} = 3.42$	$x_{14} = 0$
3	$x_3 = 0$	$x_7 = 9.05$	*	$x_{15} = 3.1$
4	$x_4 = 0$	$x_8 = 3.98$	$x_{11} = 5.75$	$x_{16} = 0$
5	$x_5 = 0$	$x_9 = 5.95$	$x_{12} = 8.97$	$x_{17} = 0$

9. SUMMARY

The SNLP system was developed within the framework of the ADO/SDS project according to the agreement between IIASA and the Glushkov Institute of Cybernetics.

The system is one of the results of the IIASA activity carried out for many years and aimed at creating the ADO/SDS Collection of Stochastic Programming Codes [13]. This activity influenced the SNLP in the following two aspects:

First, the stochastic optimization algorithm in the SNLP is based on the quasigradient techniques, though it uses some ideas of other approaches to solving stochastic problems with recourse. For instance, the set of induced constraints is approximated by a polyhedron as in the L-shaped method [16]. When the distribution type of the right side h(w) of the stochastic constraints is standard (normal, exponential, uniform, discrete), the corresponding recourse functions are calculated in the explicit analytical form [17].

Second, the data formats in the SNLP were developed in accordance with the ideas of IIASA's "standard format". But the SNLP failed to keep within the frame of this format. Apparently the "standard format" needs some development for quasigradient type algorithms.

Activity aimed at developing some modifications based on the SNLP is being carried out. Among them we may name the Stochastic Linear Programming System (SLP) being developed now at the Glushkov Institute of Cybernetics. In comparison with the SNLP, the problem statements in the SLP are more simple:

- the nonlinear function F(x) is absent, and there is no subroutine CALCFG;

- there are no distributions generated by users in the subroutine CALCTH.

These simplifications provide for an explicit analytical calculation of the recourse function's subgradients. It enables use of the methods of convex programming for solving stochastic problems with recourse resulting in raising the efficiency of calculations.

In comparison with the SNLP, the SLP system provides for a greater number of standard distributions (up to 20), and enables using only partial information about the distributions in the form of the first two moments. Also, the SLP provides for executing a post-optimal analysis for linear and stochastic problems.

The authors express their appreciation to Prof. J.M. Ermoliev, scientific director of the project, as well as to A. Nosov, L. Murga, V. Kononchuk, A. Kozlov and E. Platon, collaborators of the Glushkov Institute of Cybernetics, for their valuable contribution.

10. REFERENCES

1. G. Dantzig and Mandansky (1961): On the Solution of Two-stage Linear Programs under Uncertainty. Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability, Vol.1, Univ. California Press, Berkeley, 165-176
2. E.G. Golstein and D.B. Udin (1969): Linear Programing (theory, methods and applications). Nauka, Moscow (in Russian).
3. B. Pshenichni (1983): Linearization Methods. Nauka, Moscow (in Russian)
4. Y. Ermoliev and R. J-B Wets (eds) (1988): Numerical Techniques for Stochastic Optimization . Springer-Verlag, Heidelberg.

5. A.Gaivoronski (1988): Interactive Program SQG-PC for Solving Stochastic Programming Problems on IBM PC/XT/AT Compatibles: User Guide. Working Paper WP-88-11, IIASA, Laxenburg, Austria.

6. B.Pshenichni and U.Danilin (1981): Numerical Methods in Extremal Problems. Nauka, Moscow (in Russian).

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

8. A.Ruszynski and W. Syski (1986): A Method of Aggregate Stochastic Subgradients with On-Line Stepsize Rules For Convex Stochastic Programming Problems. Mathematical Programming Study 28, pp. 113-131.

9. J.Edwards, J.Birge, A.King, L.Nazareth (1985): Standard Input Format for Computer Codes which Solve Stochastic Programs with Recourse and a Library of Utilities to Simplify Its Use. Working Paper WP-86-32 IIASA, Laxenburg, Austria.

10. J.Birge, M.Dempster, H.Gassman, E.Gunn, A.King, S.Wallace (1987): A Standard Input Format for Multiperiod Stochastic Linear Programs. Working Paper WP-87-118 IIASA, Laxenburg, Austria.

11. B.A.Murtagh and M.A.Saunders (1977): MINOS 5.0 User's Guide, Report No.SOL 83-20, Stanford Optimization Laboratory, Stanford University (1983).

12. L.Nazareth (1987): Design and Implementation of a Stochastic Programming Optimizer with Recourse and Tenders. Working

Paper WP-85-063 IIASA, Laxenburg, Austria.

13. J.Edwards (ed.) (1985): Documentation for the ADO/SDS Collection of Stochastic Programming Codes. Working Paper WP-86-32 IIASA, Laxenburg, Austria.
14. IBM (International Business Machines, Inc.) (1972): Mathematical Programming Subsystem - Extended (MPSX) and Generalized Upper Bounding (GUB) Program Description document number SH20-0968-1.
15. Microsoft Corporation, FORTRAN Optimizing Compiler, v.4.01. User's Guide, Document No.410500001-400-R06-0187.
16. R.M.Van Slyke, R.Wets (1969): L-shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. SIAM J. Appl. Math., 17, No.4, pp. 638-663.
17. W.T.Ziemba: Stochastic Programming with Simple Recourse. Mathematical Programming in Theory and Practice, P.L.Hammer and G.Zoutendijk (eds), pp. 213-273.
18. G.Dantzig (1963): Linear Programming and Extensions, Princeton University Press, pp. 572-597.
19. A.J.King: Stochastic Programming Problems - Examples from the Literature, pp. 1-3.
20. Y.Ermoliev: Stochastic Quasigradiant Methods and Their Applications to System Optimization, Stochastics 9, 1-36, 1983.