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

A Nonlinear Dynamic Interactive Decision Analysis and Support System (DIDASS/N)

Grauer, M. and Kaden, S.

IIASA Working Paper

WP-84-023

March 1984

Working Paper

A NONLINEAR DYNAMIC INTERACTIVE DECISION ANALYSIS AND SUPPORT SYSTEM (DIDASS/N)

USER'S GUIDE (MARCH 1984)

Manfred Grauer
Stefan Kaden

March 1984

WP-84-23

International Institute for Applied Systems Analysis
A-2361 Laxenburg, Austria

A NONLINEAR DYNAMIC INTERACTIVE DECISION ANALYSIS AND SUPPORT SYSTEM (DIDASS/N)

USER'S GUIDE (MARCH 1984)

Manfred Grauer
Stefan Kaden

March 1984

WP-84-23

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
2361 Laxenburg, Austria

PREFACE

The Interactive Decision Analysis group at IIASA has developed a decision analysis and support system, called "DIDASS". Based on the Reference Point Approach for multicriteria analysis, it is an attempt to combine the analytical power of the "hard" computer model with the qualitative assessments of the decision maker.

In general, DIDASS is capable of dealing with both linear and nonlinear problems. Theoretical and practical tests for solving nonlinear problems of regional water policies in open-pit mining areas have elucidated the need for an especially designed nonlinear DIDASS version.

Following the presentation of the extended nonlinear version, DIDASS/N is described. DIDASS/N has been developed in collaboration between the Interactive Decision Analysis Group and the Regional Water Policies Project at IIASA.

DIDASS/N has been written in FORTRAN 77. The use of operating-system-dependent statements or commands has been avoided.

Either comments or suggestions concerning the analysis and support system or this guide would be welcome--DIDASS is intended to be useful, useable, and used!

ANDRZEJ WIERZBIEKI	CHESTER COOPER
Program Leader	Program Leader
Systems and Decision Sciences	Institutions and Environmental Policies

CONTENTS

A NONLINEAR DYNAMIC INTERACTIVE DECISION ANALYSIS AND SUPPORT SYSTEM (DIDASS/N)

USER'S GUIDE (MARCH 1984)

Manfred Grauer and Stefan Kaden

1. INTRODUCTION

DIDASS/N is an interactive multicriteria programming package designed for decision support. It is an improved version of DIDASS (May 1983)[5], especially designed for nonlinear multicriteria programming problems, and is based on the reference point approach to multicriteria analysis.

The basic idea of the reference point method is to rank multidimensional decision alternatives q, defined as points in the R^p ($p \geq 2$), relative to a reference point $\bar{q}$ which reflects the preferences of the user.

The ranking of the decision alternatives is based on a partial ordering of the R^p:

$$q^1 \leq q^2; \quad q_i^1 \leq q_i^2 \ ; \quad i = 1,2,\ldots p \ ; \ q^1;q^2 \in R^p \tag{1}$$

The decision problem is to determine an n-vector x of decision variables satisfying all given constraints while taking into account the p-vector of objectives. We will assume that each component of q should be as small as possible.

A *reference point* or *reference objective* is a suggestion $\bar{q}$ supplied by the user which reflects in some sense the "desired level" of the objective. An achievement scalarizing function $s(q-\bar{q})$ defined over the set of objective vectors q is then associated with each reference point $\bar{q}$ [3]. If we regard the function $s(q-\bar{q})$ as the "distance" between the points q and $\bar{q}$, then, intuitively, the problem of minimizing this distance may be interpreted as the problem of finding from within the Pareto set the point $\hat{q}$ "nearest" to the reference point $\bar{q}$. (However, the function s is not necessarily related to the usual notion of distance.) With this interpretation in mind, reference point optimization may be viewed as a way of guiding a sequence $\{\hat{q}^k\}$ of Pareto points generated from a sequence $\{\bar{q}^k\}$ of reference objectives. These sequences are generated through an interactive procedure and should result in a set of attainable efficient points $\{\hat{q}^k\}$ of interest to the user. If the sequence $\{\hat{q}^k\}$ converges, the limit may be seen as the solution to the decision problem.

2. PROBLEM FORMULATION

Let us assume that the decision problem can be clarified by analyzing a nonlinear constrained multicriteria problem in the following form:

$$\min_{x} f(x) = q \geq 0^{1)} \tag{2}$$

1) The objective functions have to be defined in such a way that they are not negative.

subject to:

$$g(x_{nl}) \leq b_1 \tag{3}$$

$$A_1 x_{nl} + A_2 x_l \leq b_2 \tag{4}$$

$$l \leq x = \begin{bmatrix} x_{nl} \\ x_l \end{bmatrix} \leq u \tag{5}$$

where $g(x_{nl}) = \left[g_1(x_{nl}), g_2(x_{nl}), \cdots, g_m(x_{nl})\right]^T$ is a vector of nonlinear constraints and $f(x) = [f_1(x), f_2(x), \cdots, f_p(x)]^T$ in (2) represents the nonlinear performance criteria. Linear objectives are considered as a part of these nonlinear criteria, without being especially treated.

The decision variables (x) are divided into two subsets: a vector of "nonlinear constrainted" variables (x_{nl}) and a vector of "linear constrainted" variables (x_l). It is clear that when g is nonexistent, formulation (2)-(5) is identical with a linear-constrainted multicriteria nonlinear programming problem. An overview of the various ways in which the reference point approach can be used in the nonlinear case is described in [4].

The decision analysis and support system DIDASS/N is based on a two-stage model of the decision-making process. In the first stage - the exploratory stage - the user may get informations about the range of his alternatives, thus giving him an overview of the problem. In the second stage - the search stage - the user works with the system in an interactive way to analyze the efficient alternatives $\{\hat{q}^k\}$ generated by DIDASS/N in response to his reference objectives $\{\bar{q}^k\}$. The initial information for the *exploratory stage* may be provided by calculating the extreme points for each of the objectives in (2) separately. A matrix D_S which yields information on the range of numerical values of each objective is then computed. We shall call this the *decision support matrix.*

$$D_S = \begin{bmatrix} q_1^* & q_2^1 & \cdots & q_p^1 \\ q_1^2 & q_2^* & \cdots & q_p^2 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ q_1^i & q_2^i & \cdots & q_p^i \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ q_1^p & q_2^p & \cdots & q_p^* \end{bmatrix} \tag{6}$$

Row i corresponds to the solution vector x_i which maximizes objective q_i. The vector with elements $q_i^i = q_i^*$, i.e., the diagonal of D_S, represents the *utopia (ideal) point.* This point is not normally attainable (if it were, it would be the solution of the proposed decision problem), but it is presented to the user as an upper guideline to the sequence $\{\bar{q}^k\}$ of reference objectives. Let us consider column i of the matrix D_S. The maximum value in the column is q_i^*. Let q_i^n be the minimum value, where

$$\min_{1 \le j \le p} \left\{ q_i^j \right\} = q_i^n \tag{7}$$

We shall call this the *nadir* value. The vector with elements $q_1^n, q_2^n, \ldots, q_p^n$ represents the *nadir point,* and may be seen as a lower guideline to the values of the user's objectives.

If the range of the objectives is known, instead of computing the decision support matrix the range of the objectives $\{q_i^{\min}\}$, $\{q_i^{\max}\}$ can be used and the effort calculating the matrix D_s avoided. This is useful respectively necessary for dynamic problems with a high number of objectives.

In the *second stage*, the reference point optimization, the following achievement scalarizing functions are maximized according to q and subject to (3 - 5):

$$s(w) = -\frac{1}{\rho}\ln\left[\frac{1}{p}\sum_{i=1}^{p} w_i^{\rho}\right] \tag{8}$$

or

$$s(w) = -\left(\frac{1}{p}\sum_{i=1}^{p} w_i^{\rho}\right)^{1/\rho} \tag{9}$$

with

$$w_i = \gamma_i \frac{\tilde{q}_i - q_i}{\tilde{q}_i - \bar{q}_i} . \tag{10}$$

The solution gives an efficient pointing $q = \hat{q}$, according to a given set of reference points $\bar{q}$. $\tilde{q}$ is a lower limit to the sequence of reference points (the utopia point q_i^* respectively $q_i^{\min}$). γ_i can be used as weighting factor and ρ is an arbitrary coefficient[2)].

$$\rho \geq p \geq 2 \tag{11}$$

This type of achievement scalarizing function meets the following requirements:

-- They yield scaling factors which make additional scaling of objectives unnecessary.

2) For $\rho=2$ we have the Euclidic norm, for $\rho \to \infty$ the Tschebyschev norm.

-- They are smoothly differentiable functions which approximate the nonsmooth function $s = \max_i w_i$.

-- They are strongly order-preserving and weakly order-approximating.

The resulting single-criterion programming problems are solved using the solution package MINOS [1,2].

3. THE DIDASS/N PROGRAM PACKAGE

3.1. Overview

DIDASS/N has been developed in FORTRAN 77. It is structured as a set of modules (subroutines). One of these modules is MINOS/AUGMENTED [1,2], for nonlinear single-objective programming. In Table 1 all used subroutines are assorted.

For input and output data as well as data which might be needed in future model runs, external files are created. Table 2 gives an overview.

In Figure 1 the structure of DIDASS/N with the interrelationship between modules and external files is illustrated. The internal data transfer between the subroutines of DIDASS/N including MINOS (GO) is organized using common blocks. Following parameter statements are implemented:

```
      character *1 l
      character *8 objnam, rhs, bds
      implicit real *8 (a-h, o-z)
      common/help/nwcore,rho,rhs,bds,l(80),nrun
      common/rfp/nc, objnam (100), gam(100),
     *       rfp (100), obj (100), dif (100)
      common/utopia/objmin (100), objmax (100)
      common z (100000)
      data nwcore/100000/.
```

Table 1. DIDASS/N - modules

Name	Contents
didass	main control program
extrem	calculation and output of extreme points (utopia/nadir) for all objectives
effici	calculation of efficient points according to a given set of reference points
intact	interactive correction of data for the calculation of efficient points
varcon	interactive output of variables/constraints for efficient points
vacose	auxiliary subroutine for varcon
readrfp	input of reference point file
yn	input of yes or no as an alternative of next program steps
obmima	input of the range of objectives
find	search of an objective-name
error	error output
chrhbd	changing of the right-hand-side or bounds set of the model
go	MINOS [1,2]
object	calculation of objective functions (nonlinear)
constr	calculation of nonlinear constraints

Consequently, the number of objectives is restricted to 100. If necessary the dimension (100) of the defined arrays may be changed in the subroutines.

Further, the MINOS-array restriction have to be considered (see, [1, 2]).

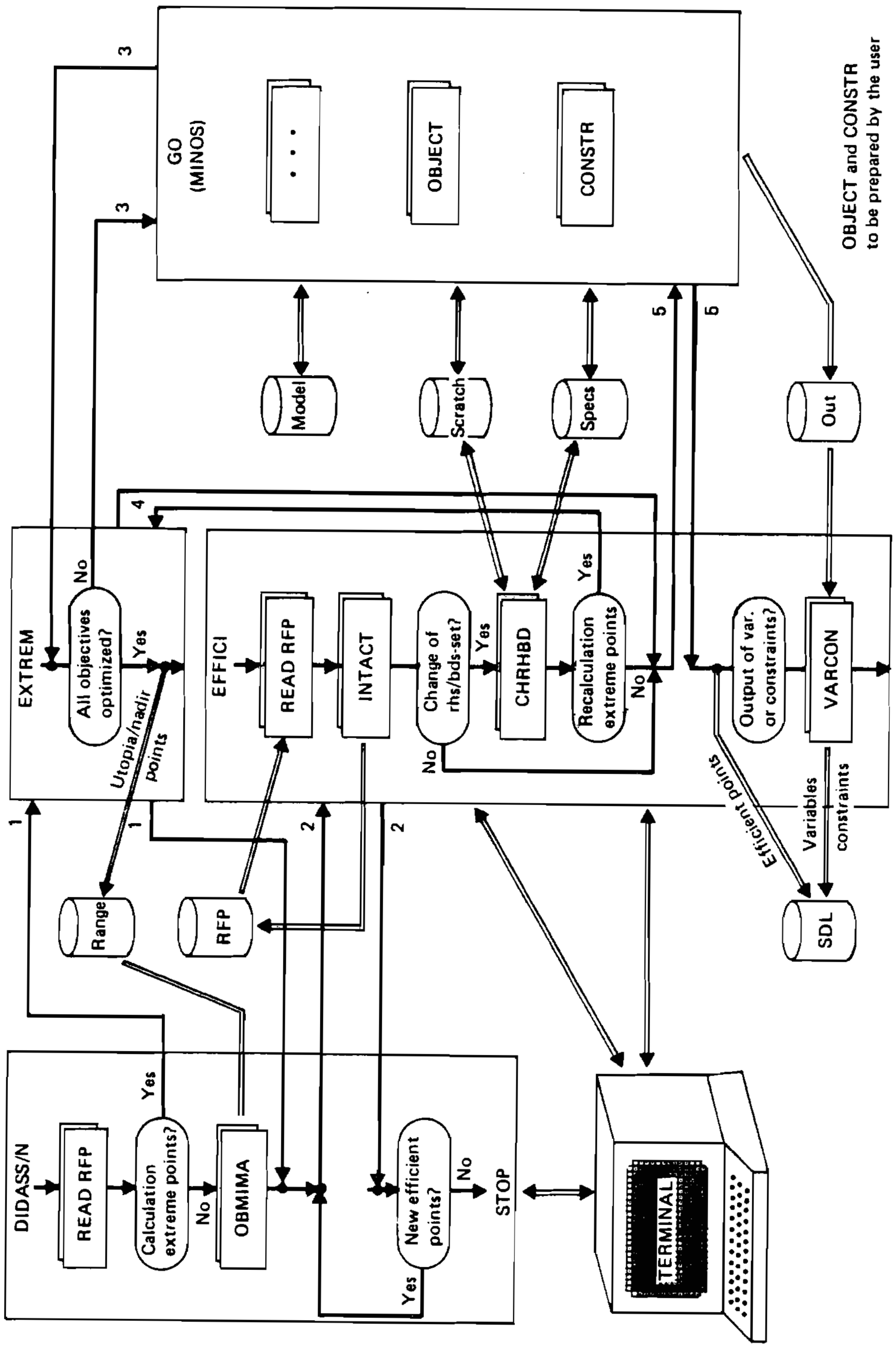


Figure 1. DIDASS/N overview

Table 2. DIDASS/N - file system

unit-number	file name	remarks
2	sol	DIDASS/N - result file
3	rfp	reference point file
4	range	ranges for the objective values as an input or utopia and nadir points as an output
5	specs	specification file according to MINOS [1, 2]
6	out	MINOS-output of the actual last MINOS-run
7	obj	values of the objective functions of the last computed efficient point
8[3]	fort-8, help	scratch-file for MINOS and DIDASS/N (for specs-file processing)
9	model	problem description in MPS-format according to MINOS
14	standard input, output	usually terminal - display

[3]This file is internally opened and closed, if the program run is finished normally. In the case of a program exit due to errors, the files should be removed by the user.

3.2. Data Required from the User

The data preparation is based on the data preparation for the MINOS system [1, 2]. Following data files are required:

specs – specification file (see section 3.2.1)

model – problem description in MPS-format (see section 3.2.2)

rfp – reference point file (see section 3.2.3)

range – range file for the objective values, if the extreme values for all objectives shall not be computed (see section 3.2.4).

The nonlinear objectives and constraints have to be written as FORTRAN statements within the subroutines

object - objective functions (see section 3.2.5)

constr - constraints (see section 3.2.6)

3.2.1. SPECS - file

In principle, for the preparation of the specs-file the MINOS-users guide [1] and -manual [2] should be used. There are no restrictions to the MINOS capabilities. Certain parameters have to be or may be used by the user via a list of problem specifications. They are assumed to be a deck of 80-character card images. Each card contains a sequence of items produced in free format (i.e., separated by at least one blank or =) with keywords and numbers. Blank cards are allowed, and comments may occur after an asterisk (*).

Following a standard specs file for DIDASS/N is given. Values, which have to be inserted by the user are characterized by < >.

begin	<name of the problem>
minimize	
nonlinear constraints	<m, number of nonlinear constraints>
nonlinear jacobian vars	<number of variables $\{x_{ns}\}$ in nonlinear constraints>
nonlinear objective vars	<number of variables {x} altogether>
bounds	<name of bounds data set, usually bnd>
rhs	<name of right hand-side data set, usually rhs>
rows	<over-estimate of number of the number m of constraints>
columns	<over-estimate of number of variables $\{x\}$>
elements	<over-estimate of the number of nonzero elements in the linear constraints $\{A_1,A_2\}$>
objective = object	
problem no.	<problem number for subroutines constr and object>
mps file	9
solution	yes

```
* the following values may be changed by the user
* according to the numerical problem

aijtol                  0.000001
difference interval     1.0e-6
dj tolerance            1.0e-6
feasibility tolerance   1.0e-6
linesearch toler        0.1
lower bound             0
iterations              1000
major iterations        10
minor iterations        20
penalty parameter       0.1
radius of conver        0.01
row tolerance           1.0e-6
*
*
superbasics             <number of super basic variables, normally
                         not greater than number of variables + 1>
hessian dimension       <number of variables + 1>
jacobian                dense 4)
print level (jflxi)     <amount of information to file out,
                         typical value 1>
derivative level        <3 - objective and constraint
                             gradients are known
                          2 - constraint gradients are known>

call function rountines when optimal
end
```

4) This determines the manner in which the constraint gradients are evaluated and stored. For complicated problems with a great number of variables 'sparse' should be used, for the consequences see MINOS [2].

3.2.2. MODEL - File

The data specifying the constraints (3)-(5) have to be prepared in standard MPS format. For details see MINOS [1, 2]. The following has to be considered (compare section 4, examples):

- Nonlinear constraints have to be listed first.
- The ordering of variables must be the same as in the x-array of the subroutines CONSTR and OBJECT (see below). Variables which occur in the nonlinear constraints have to be listed first.

-- All variables should be specified by upper and lower bounds.

-- For constraints ranges should be defined.

-- A set of initial variables should be given.

3.2.3. RFP - File

The reference point file contains for all objectives $i=1,p$

<name objective i> <reference point $\bar{q}_i$> <wrighting factor γ_i>.

In the first line the coefficient ρ, see section 2, has to be added. The format is

(2x ,a8 ,2x ,3f 12.5) .

The last line must contain dots (....) as characters 5-8 (compare section 4, examples).

3.2.4. RANGE - File

The range file contains for all objectives $i=1,p$

<name objective i> <$q_i^{\min}$> <$q_i^{\max}$>

in the format (2x , a 8 , 2f 12.5).

3.2.5. Subroutine OBJECT

The objectives $f_i(x)$ have to be programmed in FORTRAN-statements in the subroutine OBJECT. The following one-dimensional arrays have to be used.

obj - values of the objective functions

x - values of the variables (x(1) corresponds to column 1 in the constraints of the model-file, etc.).

Usually the gradients are calculated automatically. Appendix 2.2 shows the subroutine object for the test examples of section 4. For complecated functions the corresponding gradients may be programmed. Therefore, the one-dimensional array

g - gradients of objective functions

has to be used. This is demonstrated for the example TEST 2 in Appendix 2.2. There it is also illustrated, how instead of the x-array the original variable names of the model-file may be used for the subroutines OBJECT as well as CONSTR. In this case, two additional subroutines OBJGRA and VALIST (see Appendix 2.2) are needed.

In contrast to MINOS in DIDASS/N it is not allowed to define the objective functions and their gradients partial in the MPS file (model-file), because more than one objective function has to be considered.

3.2.6. Subroutine CONSTR

The nonlinear constraints $g_i(x)$,$i=1,...,m$ and the corresponding Jacobian matrix $J(x)$ have to be programmed in FORTRAN-standard in the subroutine CONSTR. The following arrays have to be used:

one-dimensional:

g - values of the constraints (g(1) corresponds to the row 1 in the constraints of the model-file, etc.)

x - values of the variables (x(1) corresponds to column 1 in the constraints of the model-file, etc.).

two-dimensional:

gj - Jacobian matrix.

A partial specification of the Jacobian matrix in the model-file is possible (see MINOS [1, 2]). Such a specification is necessary, if the Jacobian matrix shall be stored in "sparse" mode.

In Appendix 2.1 the subroutine CONSTR for the test examples of section 4 is demonstrated. In Appendix 2.2 the subroutine CONSTR is shown using the original variable names of the model-file.

3.3. Interactive Use/Error Messages

The DIDASS/N package has been designed with special regard to its interactive use. During the program runs the user may change data, generate different program modes and define the amount of data output. The interactions are controlled by a sequence of questions, alternatives and data requirements to be decided by the users. An easy way to get known with these interactions is the practical test. The system is so organized, that wrong actions do not lead to program failures. If a reaction to a given request is wrong, this request will be given again. For instance, if an objective name is input which does not exist in the model-file, a new objective name is required.

For the interactions special keywords are used. It is possible to type the complete keywords or their abbreviation (underlined by an asterisk).

Example: enter (yes *or* *no*)!
                    *   *

The answer may be y/n but also yes/no. The complete list of all interactive capabilities is given in Appendix 1, see also the list of examples of section 4, Appendix 3.

During the DIDASS/N run error messages as indicated in Table 3 are possible.

Table 3. Error messages

Message	Cause/consequence
**** ERROR incorrect reference point file (.... missing?)	missing end data ... in rfp file /Stop
**** ERROR no rhs entry in specs file	missing rhs entry in specs-file /Stop
**** ERROR no bounds entry in specs file	mission bounds entry in specs file /Stop
**** ERROR not enough ranges	number of ranges in range file less than number of objectives/Stop
**** ERROR wrong order of ranges	order of ranges not adequate to the objective names /Stop
**** ERROR no solution in MINOS out file	single-objective optimization in MINOS not finished /continue
**** ERROR no optimal solution	no optimal solution of single-objective optimization in MINOS/continue

3.4. Data Output

The main results of DIDASS/N are listed on the screen of a terminal (standard unit). This includes

-- table of the results of the calculation of extreme points for all objectives

-- table of efficient points (objective values)

-- values of variables and constraints for efficient points.

In the Appendix 3 the terminal output is illustrated for the examples of section 4.

Additional to the terminal output, following output files are created:

out - MINOS-output depending on the value of print level in the specs file, see [1,2] and section 3.2.1.

sol - DIDASS/N results, a copy of the terminal output including the actual data of the DIDASS/N run, see Appendix 3 for the test examples.

range - Name, utopia and nadir point of all objectives, format (2x , a8 , 2g12.5).

The output of results (Values of objective functions, constraints and variables) is printed in the format g12.5.

That means numbers greater than 0.1 and less than 10^6 are printed in f-format otherwise in e-format.

3.5. Implementation

3.5.1. IIASA Operating System (UNIX)

For the internal use at IIASA the DIDASS/N package is available as a directory including the files, listed in Table 4.

For the compiling, linking and loading the executable files **dida_t.tra** respectively dida_tc.tra should be used. For instance **dida_t.tra**:

```
xf77    constr_t.f object_t.f didass.0 nonlp.0
          - lxU77 5) -a dida_t
```

5) *Date*, subroutine of the UNIX utility library.

Table 4. DIDASS/N - directory

Blocks	file-name	contents
4 6	constr_t.f constr_t.0	subroutine constr for test 1 and test 2 (Appendix 2.1)
4 7	constr_tc.f constr_tc.0	subroutine constr for test 2 (Appendix 2.2)
988	dida-t*	didass/n, loaded for test 1 and test 2
1	dida_t.tra*	exectable file for preparing dida_t*
1 1	dida_t1.run* dida_t2.run*	executable files for program run test 1 and test 2
988	dida_tc2*	didass/n, loaded for test 2, programmed objective functions gradients
1	dida_tc2.run*	executable file for program run test 2/c
1	dida_tc2.tra*	executable file for preparing dida_tc*
31 47	didassn.f didassn.0	didassn programs
6 7	model.t1 model.t2	model file for test 1 and test 2
1048	nonlp.0	MINOS/AUGMENTED
1 1 1	obj.t1 obj.t2 obj.tc2	efficient points of the last session, test 1, test 2, test 2/c
5 5	object_t.f object_t.0	subroutine object for test 1 and test 2 (Appendix 2.1)
8 7	object_tc.f object_tc.0	subroutine object for test 2/c (Appendix 2.2)
28 50 74	out.t1 out.t2 out.tc2	MINOS - outfile, test 1, test 2 and test 2/c
1 1 1	range.t1 range.t2 range.tc2	range-file for test 1, test 2 and test 2/c
1 1	rfp.t1 rfp.t2	rfp-file for test 1 and test 2
3 5 7	sol.t1 sol.t2 sol.tc2	sol-file for test 1, test 2 and test 2/c
2 2 2	specs.t1 specs.t2 specs.tc2	specs-file for test 1, test 2 and test 2/c

To start DIDASS/N the executable file dida_ $\begin{Bmatrix} t1 \\ t2 \\ tc2 \end{Bmatrix}$. run has to be actualized, defining the input/output files.[6)]

didassn 2 = =_ 'sol' 3 = 'rfp' 4 = 'range' 5 = 'specs' 6 = 'out' 7 = 'obj' 9 = 'model' 14 ==.

In this case, the new solution is added to the sol-file. For the filenames, the actual filenames have to be inserted. For instance dida_t1.run

dida_t 2 ==_ sol.t1 3 = rfp.t1 4 = range.t1 5 = specs.t1 6 = out.t1 7 = obj.t1 9 = model.t1 14 ==.

6) *Usearg*, IIASA specific subroutine which permits the assignment of files at run time.

3.5.2. External Use

The current version of the DIDASS/N package has been designed specifically to be portable, and it has therefore been written completely in FORTRAN 77, avoiding the use of operating-system-dependent statements or commands.

The DIDASS/N source code including data files for the examples is normally supplied by IIASA on tape (9-track, unlabeled, ebcdic, upper case, 800 bpi, block size 800 characters, record length 80 characters) under the names listed in Table 5.

Table 5. DIDASS/N - source tape

file-name	contents
constr_t.f constr_tc.f	constr for test 1 and 2 constr for test 2/c
didassn.f	didass/n package
model.t1 model.t2	model files for test 1 model files for test 2
nonlp.t	MINOS/AUGMENTED
object_t.f object_tc.f	object for test 1 and 2 object for test 2/c
rfp.t1 rfp.t2	rfp file for test 1 rfp file for test 2
specs.t1 specs.t2 specs.tc2	specs file for test 1 specs file for test 2 specs file for test 2/c

To prepare DIDASS/N the user must compile link and load to following FORTRAN files:

consr_t.f object_t.f didassn.f nonlp.f

The input and output files are presumed under the names of Table 1. Therefore, in the main program the following file-definitions are included.

```
iin = 14     (standard input)
ion = 14     (standard output)
open (2, file = 'sol')
open (3, file = 'rfp')
open (4, file = 'range')
open (5, file = 'specs')
open (6, file = 'out')
open (7, file = 'obj')
open (9, file = 'model')
open (14, file = '/dev/tty')
```

This may be changed, if necessary. It has also to be proved, whether a subroutine date

date (datum)

datum:

24 character string with the current date and time in ascii form

is available. If necessary, the program lines

```
character*24 datum
call date (datum)
write (2,201) datum
```

have to be changed in the main program didass.

4. TEST-EXAMPLES

4.1. Solving a Problem with Linear Constraints and Nonlinear Objectives

To demonstrate the use of DIDASS/N, in the first test example test 1 the following quadratic programming problem is described.

$$\min \left\{ \begin{array}{c} (x_1-3)^2 + (x_2-2)^2 + (x_2-6)^2 + (x_7-4)^2 = obj\,1 \\ 0.5(x_3-4)^2 + (x_8-6)^2 + (x_9-11)^2 = obj\,2 \\ (x_4-1)^2 + (x_5-8)^2 + (x_{11}-4)^2 + (x_{12}-1)^2 + (x_{10}-8)^2 = obj\,3 \end{array} \right\} \quad (12)$$

subject to:

$$
\begin{aligned}
2x_1+0.5x_2-x_6+x_{11} &= 6 \\
x_1+2x_2-x_7+x_{11} &= 0 \\
x_3+0.5x_6-x_8-x_{12} &= 0 \\
0.5x_3+x_6-x_9+x_{12} &= 0 \\
x_4+0.5x_5+0.5x_8-x_{10} &= 0 \\
2x_4+x_5-x_8-x_{11} &= 0 \\
3x_4-x_5+x_8-x_{12} &= 0 \\
2x_1+x_2+2x_{11} &\le 9 \\
2x_6+3x_{12} &\le 13 \\
5x_4+3x_5 &\le 16 \\
3x_4+2x_5+3x_8 &\le 13 \\
3x_1+2x_2 &\le 14
\end{aligned}
\tag{13}
$$

and

$$x_i \ge 0, \ i=1,2,\dots,12 \tag{14}$$

$$x_1 \le 2, \ x_2 \le 6, \ x_3 \le 3, \ x_4 \le 2, \ x_5 \le 4, \ x_6 \le 4, \ x_8 \le 3, \ x_{11} \le 3, \ x_{12} \le 2$$

In Appendix 2.1 the subroutines constr_t.f and object_t.f are presented (nprob=1). The corresponding input files

specs.t1, model.t1 and rfp.t1,

a copy of the terminal input/output as well as the output files

sol.t1 and range.t1

are given in Appendix 3.1.

The consumed computing time (VAX) was 3:18 sec.

4.2. Solving a Problem with Nonlinear Constraints and Objectives

The second example is related to a practical problem--the analysis of regional water policies in open-pit mining areas [6]. A simplified test area had been chosen, which is shown in Figure 2.

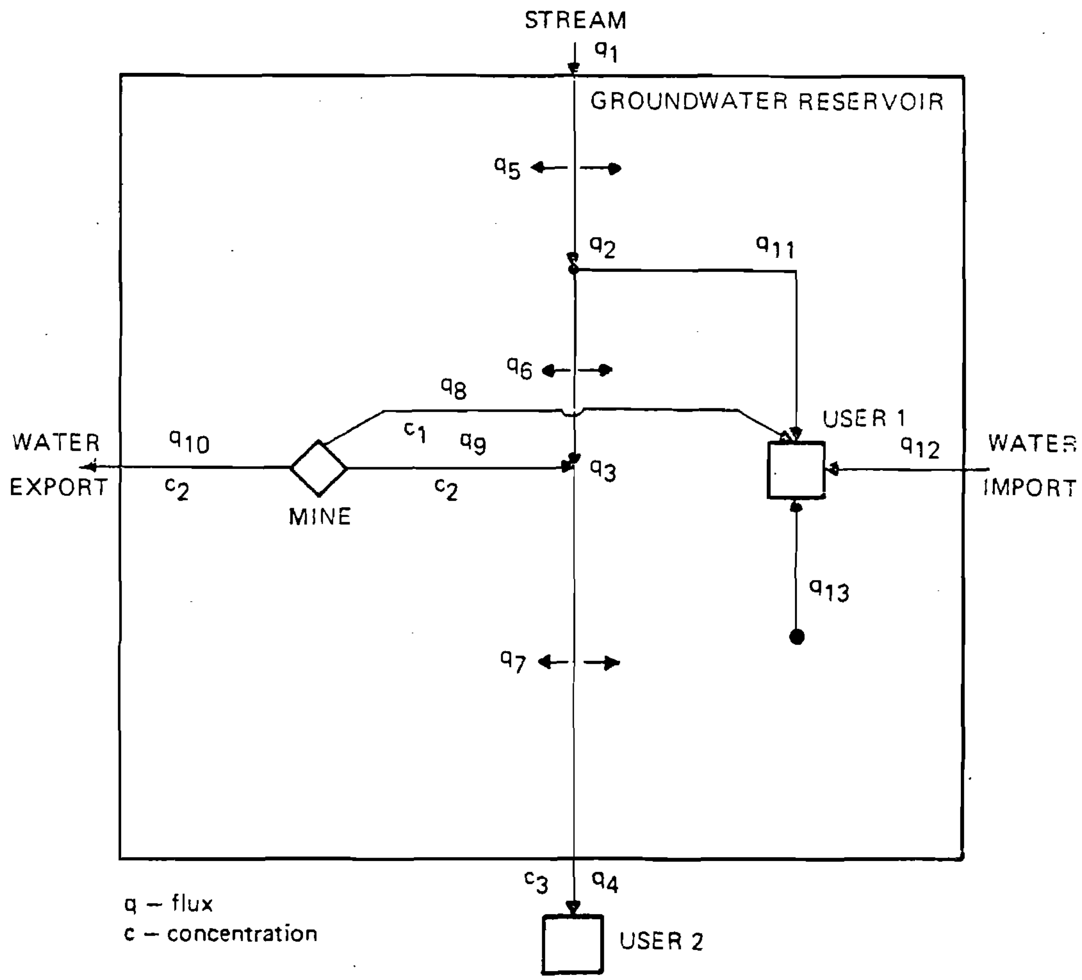


Figure 2. Schematized test area for test example test 2.

The main impacts on the water resources system are:

-- regional lowering of groundwater table which essential effects the river flow (infiltration losses) as well as a groundwater-waterwork in this region;

-- high mineralized mine drainage water which is needed for river flow augmentation but effects the downstream water use.

Possible technological alternatives are for instance:

- water import for water supply and/or flow augmentation
- export of high mineralized water
- selective mine drainage
- treatment of high mineralized water.

The following nonlineaar static model has been used.

Objective Functions

Minimizing deviation between water supply and demand

$$obj1 = 1000-(q_8+q_{11}+q_{12}+q_{13}) \quad \text{USER1} \qquad (15)$$

$$obj2 = 1000-q_4 \quad \text{USER2}$$

Minimizing costs for water supply USER 1

$$obj4 = (1.0+0.01.c_1).q_8+1.5q_{11}+q_{12}+q_{13}$$

Minimizing costs for mine drainage

$$obj3 = 2q_8+q_9+1.5q_{10}+400.0$$

Minimizing costs for water supply USER 2

$$obj5 = 0.01.q_4.c_3+500.0$$

Constraints

Flux balance for river sections

$$150-q_5-q_2 = 0 \quad q_2-q_6-q_{11}-q_3 = 0 \quad q_3+q_9-q_7-q_4 = 0$$

Groundwater tables (response functions)

$$30 > h_1 = 50-0.5(q_9+q_{10})-0.1q_8-0.01q_{13}+0.001(q_9^2+q_{10}^2)$$

$$+0.0002q_8^2+0.1q_5+0.3q_6+0.2q_7 \qquad (16)$$

$$60 < h_2 = 80-0.2q_{13}-0.1(q_8+q_9+q_{10})$$

$$+0.01q_5+0.02q_6+0.03q_7$$

Bank filtration

$$q_5 = 27-20\exp(-0.01(q_8+q_9+q_{10})-0.001q_{13}+0.002q_6+0.01q_7)$$

$$q_6 = 22.2-20\exp(-0.02(q_8+q_9+q_{10})-0.002q_{13}+0.001q_5+0.001q_7)$$

$$q_7 = 44.2-40\exp(-0.02(q_8+q_9+q_{10})-0.005q_{13}+0.001q_5+0.002q_6)$$

Minieralization

$$c_1 > 100+0.1q_8 \quad c_2 > 200+0.2(q_9+q_{10}) \quad c_3 . q_4 < c_2 . q_9$$

Bounds

$$0 \le q_i \le 200, \ i=1,13 \quad c_i \ge 0, \ i=1.3 \qquad (17)$$

$$c_1 < 500 \quad c_2 < 1000 \quad c_3 < 200$$

In Appendix 2, the subroutines constr_t.f and object_t.f are described (nprob=2). The subroutine object is shown for the case of automatic computed objective functions gradients (Appendix 2.2) and programmed objective functions gradients (Appendix 2.3).

The corresponding input files for the last case specs.tc2, model.t2 and rfp.t2

and a copy of the terminal input/output are listed in Appendix 3.2.

For one run including the calculation of extreme points and an efficient point the CPU-time on the VAX was 6:48 sec. The program version with automatic computed function gradients has consumed 6:38 sec. The numerical results are identical.

REFERENCES

[1] B.A. Murtagh and M.A. Saunders, "MINOS/USER'S GUIDE", Technical Report SOL-77-9 Systems Optimization Laboratory, Stanford University (1977).

[2] B.A. Murtagh and M.A. Saunders, "Minos/Augmented", Technical Report SOL-80-14, Systems Optimization Laboratory, Stanford University (1980).

[3] A. Wierzbicki, "A mathematical basis for satisficing decision making", pp. 465-485 in *Organizations: Multiple Agents with Multiple Criteria*, Ed. J.N. Morse, Springer-Verlag, Berlin, New York (1981).

[4] M. Grauer, "Reference point optimization - the nonlinear case", pp. 126-135 in *Essays and surveys on Multiple Criteria Decision Making*, Ed. P. Hansen, Springer-Verlag, New York (1983).

[5] M. Grauer, "A dynamic interactive decision analysis and support system (DIDASS), user's guide (May 1983)", WP-83-60, IIASA, June (1983).

[6] S. Kaden, "Analysis of regional water policies in open-pit mining areas - a multicriteria approach", presented at the IIASA Workshop on Interactive Decision Analysis and Interpretative Computer Intelligence, Laxenburg, 20-23.9.1983.

APPENDIX 1: DIDASS/N - INTERACTIVE CAPABILITIES

The interactive capabilities are characterized in form of a flow scheme. Following symbols are used:

\>   < text or result-output of the terminal

[ ] terminal input

<   > name or value for the input

\>text<
enter the problem name (<1 line, 80 char.)

[<problem name>]

calculation of extreme points?
enter (yes or no)!

[yes]

\>results of calculation of extreme points for all objectives<

[no]

(1) →

input of ranges from unit 4
\>text<
you can now:

list the reference point file, enter (list)

exit from editing status, enter (exit)

change the reference point file, enter (rfp)

copy last efficient points, enter (copy)

change the scale coefficient, enter (scal)

change the control coefficient, enter (rho) list

>table of actual reference points<

rfp

enter obj.name or (*), if exit

<obj.name>

enter new rfp. value

<new rfp. value>

>old value, new value<

copy

previous solution copied as rfp.
>table of actual reference points<

scal

enter obj.name or (*), if exit

<obj.name>

enter new scale coefficient

<new scale coef.>

>old value, new value<

rho

enter new value for rho

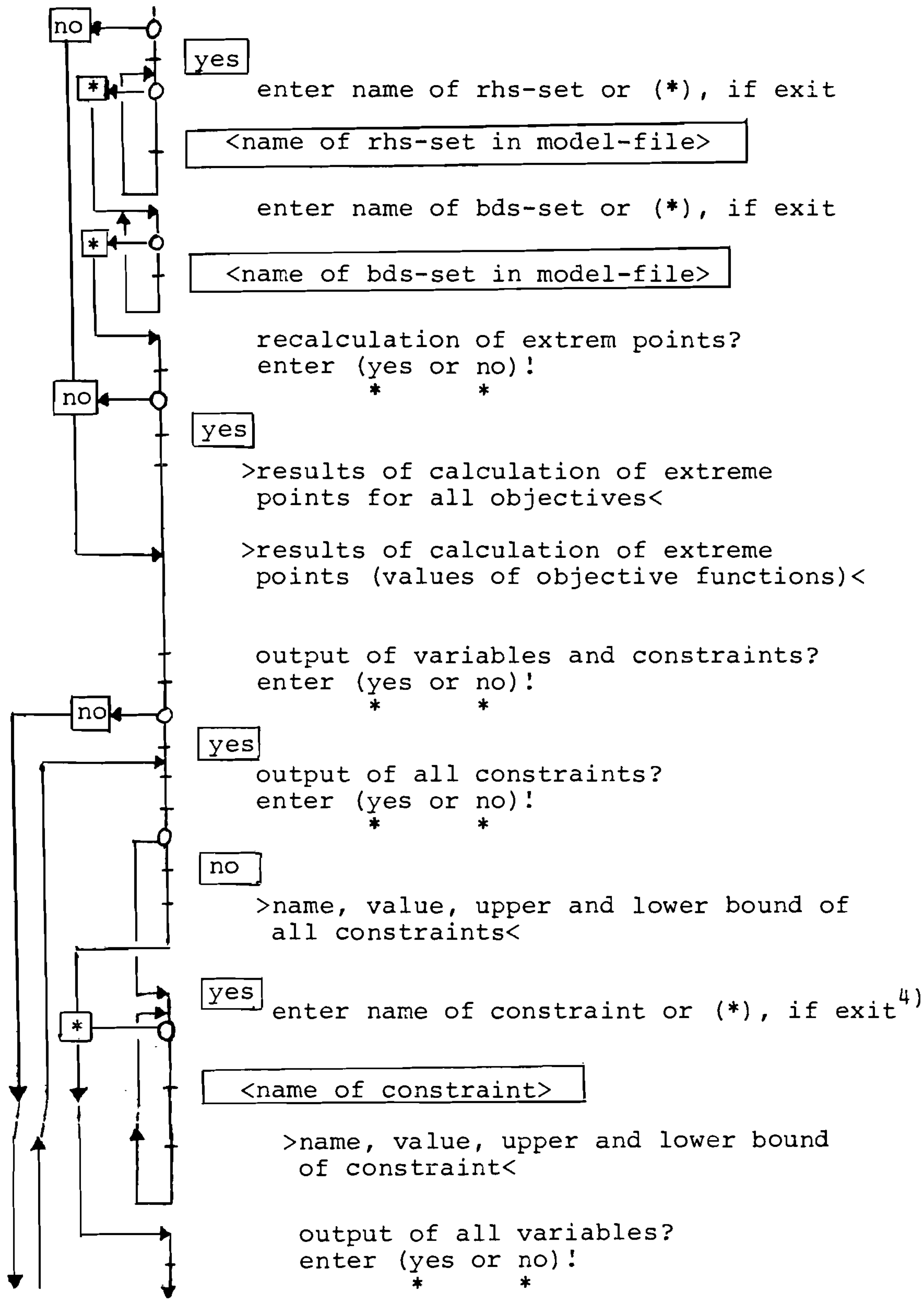
<new rho>

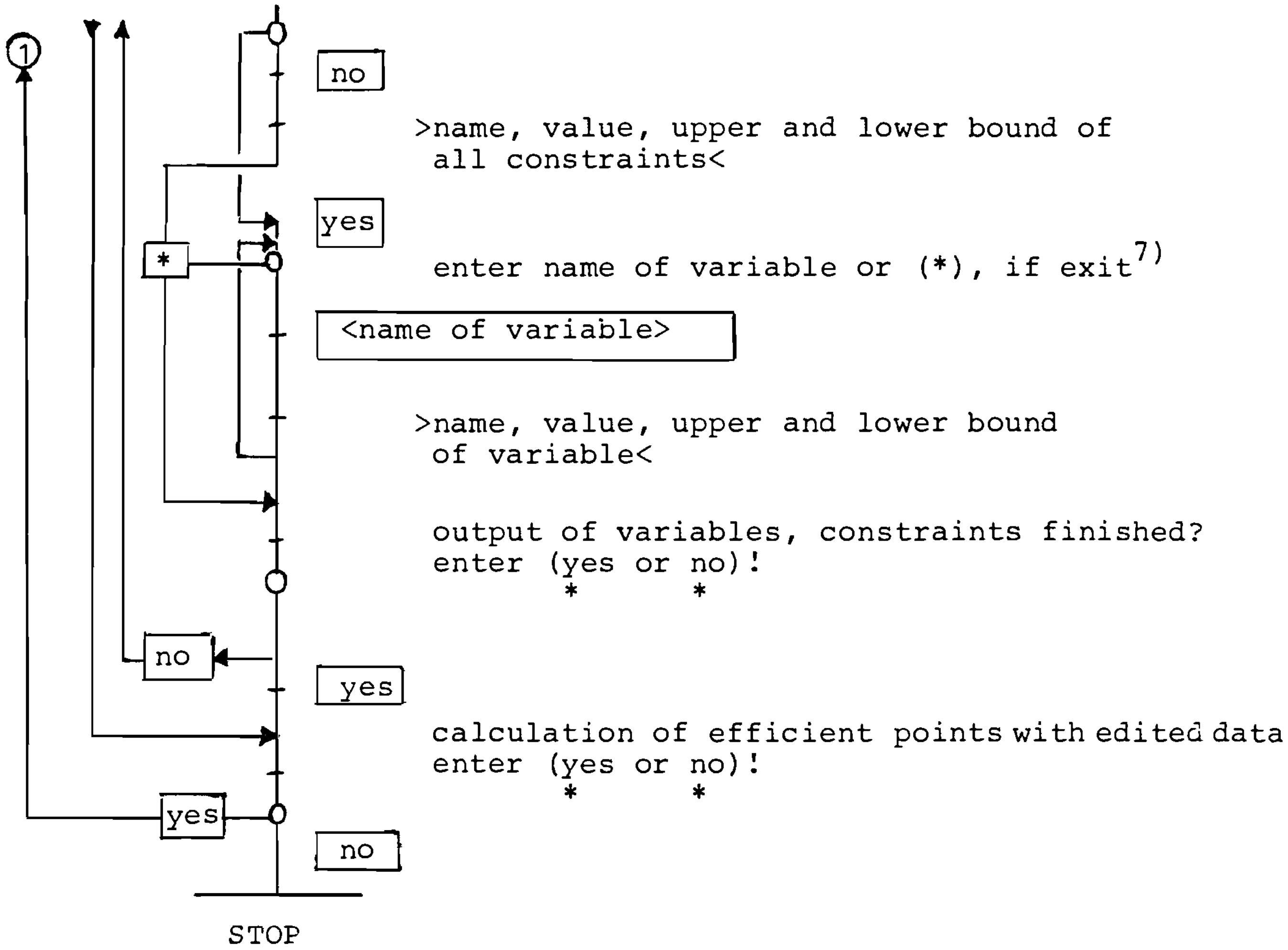
>old value, new value<

exit

```
enter (list | copy | scal | rho | rfp | exit)!
       *      *      *      **    *     *
```

```
change the rhs and/or bounds set?
enter (yes or no)!
       *      *
```


no
yes
enter name of rhs-set or (*), if exit
*
<name of rhs-set in model-file>
enter name of bds-set or (*), if exit
*
<name of bds-set in model-file>
recalculation of extrem points?
enter (yes or no)!
no
yes
>results of calculation of extreme points for all objectives<
>results of calculation of extreme points (values of objective functions)<
output of variables and constraints?
enter (yes or no)!
no
yes
output of all constraints?
enter (yes or no)!
no
>name, value, upper and lower bound of all constraints<
yes
enter name of constraint or (*), if exit4)
*
<name of constraint>
>name, value, upper and lower bound of constraint<
output of all variables?
enter (yes or no)!



[7] The search for variables and constraints is done in the same order as the values are stored, that means for instance x(1) would not be founded after a search for x(2), etc. In such a case, the output of variables/constraints may be started again and x(1) be searched.

APPENDIX 2: SUBROUTINES CONSTR.F AND OBJECT.F

2.1: CONSTR_T.F, OBJECT_T.F
GRADIENTS AUTOMATICALLY COMPUTED

2.3: CONSTR_TC.F, OBJECT _TC.F
OBJECTIVE FUNCTION GRADIENTS PROGRAMMED

In the listings those program lines are signed which have to be prepared by the user.

APPENDIX 2.1: CONST_T.F, OBJECT_T.F
OBJECTIVE FUNCTION GRADIENTS AUTOMATICALLY
COMPUTED

```
      subroutine constr( mode,m,n,njac,x,g,gj,nstate,nprob )
c
c *** mode   - if mode=-1 termination
c *** m      - number of nonlinear constraints
c *** n      - number of nonlinear variables nl
c *** njac   - m*n
c *** x      - values of nonlinear variables
c *** g      - constraints
c *** gj     - jacobian matrix of constraints
c *** nstate - status parameter
c *** nprob  - problem number
c
      implicit real*8(a-h,o-z)
      real*8 x(n),g(m),gj(m,n)
      goto (200,100),nprob
c
c *** test2
c
100   do 1 i=1,m
      do 1 j=1,n
      gj(i,j)=0.0
1     continue
      g(1)=0.5*(x(6)+x(7))+0.1*x(5)+0.01*x(8)
     *      -0.001*(x(6)**2+x(7)**2)-0.0002*x(5)**2
     *      -0.1*x(2)-0.3*x(3)-0.2*x(4)
      gj(1,2)=-0.1
      gj(1,3)=-0.3
      gj(1,4)=-0.2
      gj(1,5)=0.1-0.0004*x(5)
      gj(1,6)=0.5-0.002*x(6)
      gj(1,7)=0.5-0.002*x(7)
      gj(1,8)=0.01
c
      d=x(5)+x(6)+x(7)
      c=exp(-0.01*d-0.001*x(8)+0.002*x(3)+0.001*x(4))
      g(2)=x(2)+20*c
      gj(2,2)=1.0
      gj(2,3)=0.04*c
      gj(2,4)=0.02*c
      gj(2,5)=-0.2*c
      gj(2,6)=gj(2,5)
      gj(2,7)=gj(2,6)
      gj(2,8)=-0.02*c
```

```
c
      c=exp(-0.02*d-0.002*x(8)+0.001*x(2)+0.001*x(4))
      g(3)=x(3)+20*c
      gj(3,2)=0.02*c
      gj(3,3)=1.0
      gj(3,4)=gj(3,2)
      gj(3,5)=-0.4*c
      gj(3,6)=gj(3,5)
      gj(3,7)=gj(3,6)
      gj(3,8)=-0.04*c
c
      c=exp(-0.02*d-0.005*x(8)+0.001*x(2)+0.002*x(3))
      g(4)=x(4)+40*c
      gj(4,2)=0.04*c
      gj(4,3)=0.08*c
      gj(4,4)=1.0
      gj(4,5)=-0.8*c
      gj(4,6)=gj(4,5)
      gj(4,7)=gj(4,6)
      gj(4,8)=-0.2*c
c
      g(5)=x(1)*x(10)-x(6)*x(9)
      gj(5,1)=x(10)
      gj(5,6)=-x(9)
      gj(5,9)=-x(6)
      gj(5,10)=x(1)
200   return
      end
c


      subroutine object( mode,n,x,f,g,nstate,nprob)
c
c *** calculation of objective functions
c
c *** mode   - if mode=-1 termination
c *** n      - number of nonlinear variables
c *** x      - values of nonlinear variables
c *** f      - objective function
c *** g      - gradient vector
c *** nstate - status parameter
c *** nprob  - problem number
c
      implicit real*8 (a-h,o-z)
      dimension x(n),g(n)
      character*1 l
      character*8 objnam,rhs,bds
      common/help/nwcore,rho,rhs,bds,l(80),nrun
      common/rfp/nc,objnam(100),gam(100),
     *           rfp(100),obj(100),dif(100)
      common/utopia/objmin(100),objmax(100)
      common z(100000)
c
c *** Insert here the criteria functions in FORTRAN-statements.
c
      goto (1,2),nprob
```

```
c
c *** This is the testproblem test1 with quadratic criteria
c *** functions and linear constraints.
c
1     obj(1)=((x(1)-3)**2+(x(2)-2)**2+(x(6)-6)**2+(x(7)-4)**2)
      obj(2)=(0.5*(x(3)-4)**2+(x(8)-6)**2+(x(9)-11)**2)
      obj(3)=((x(4)-1)**2+(x(5)-8)**2+(x(11)-4)**2
     *        +(x(12)-1)**2+(x(10)-8)**2)
      goto 3
c
c *** This is the testproblem test2 with nonlinear criteria
c *** functions and nonlinear constraints
c
2     obj(1)=1000.0-x(5)-x(8)-x(13)-x(15)
      obj(2)=1000.0-x(1)
      obj(3)=(1.0+0.01*x(14))*x(5)+x(8)+x(13)+1.5*x(15)+500.0
      obj(4)=2*x(5)+x(6)+1.5*x(7)+400.0
      obj(5)=0.01*x(1)*x(10)+500.0
c
3     if (nstate .eq. 2 ) return
      if (nrun .ne. 1 ) goto 20
c
c *** quadratic scalarizing function is used for the calculation
c *** of the decision support matrix.
c
      f=0.0
      do 10 k=1,nc
          c=rfp(k)
          if(dabs(c).lt.1.) c=1.
          c=gam(k)*obj(k)/c
          f=c*c+f
10    continue
      return
c
c *** The automatic scaled achievement variables are calculated.
c
20    if (nstate.ne.1) goto 60
      do 30 i=1,nc
          if (rfp(i) .le. objmin(i)) goto 40
          dif(i)=.5*objmin(i)
30    continue
      goto 60
40    continue
      do 50 i=1,nc
          dif(i)=.5*rfp(i)
50    continue
c
c *** The achievement scalarizing function has to be inserted
c
60    s=.0
      do 70 i=1,nc
          w=((dif(i)-obj(i))/(dif(i)-rfp(i)))*gam(i)
          s=s+w**rho
70    continue
      s=s/nc
      goto (80,90),nprob
```

```
c
c *** test1
c
c
c *** The logarithmic scalarizing function is used
c
80     f=(dlog(s))/rho
       return
c
c *** test2
c
90     f=s**(1/rho)
       return
       end
```

APPENDIX 2.2: CONST_TC.F, OBJECT_TC.F
OBJECTIVE FUNCTIONS GRADIENTS PROGRAMMED

```
       subroutine constr( mode m,n,njac,x,g,gj,nstate,nprob )
c
c *** mode    - if mode=-1 termination
c *** m       - number of nonlinear constraints
c *** n       - number of nonlinear variables nl
c *** njac    - m*n
c *** x       - values of nonlinear variables
c *** g       - constraints
c *** gj      - jacobian matrix of constraints
c *** nstate  - status parameter
c *** nprob   - problem number
c
       implicit real*8(a-h,o-z)
       real*8 x(n),g(m),gj(m,n),v(10)
       equivalence (v(1),q4),(v(2),q5),(v(3),q6),(v(4),q7),
      *            (v(5),q8),(v(6),q9),(v(7),q10),(v(8),q13),
      *            (v(9),c2),(v(10),c3)
c
c *** test2
c
       call valist(10,x,v)
       do 1 i=1,m
       do 1 j=1, n
       gj(i,j)=0.0
1      continue
```

```
c
c *** gwtabl
c
      g(1)=0.5*(q9+q10)+0.1*q8+0.01*q13
     *     -0.001*(q9**2+q10**2)-0.0002*q8**2
     *     -0.1*q5-0.3*q6-0.2*q7
      gj(1,2)=-0.1
      gj(1,3)=-0.3
      gj(1,4)=-0.2
      gj(1,5)=0.1-0.0004*q8
      gj(1,6)=0.5-0.002*q9
      gj(1,7)=0.5-0.002*q10
      gj(1,8)=0.01
c
c *** bafill
c
      d=q8+q9+q10
      c=exp(-0.01*d-0.001*q13+0.002*q6+0.001*q7)
      g(2)=q5+20*c
      gj(2,2)=1.0
      gj(2,3)=0.04*c
      gj(2,4)=0.02*c
      gj(2,5)=-0.2*c
      gj(2,6)=gj(2,5)
      gj(2,7)=gj(2,6)
      gj(2,8)=-0.02*c
c
c *** bafil2
c
      c=exp(-0.02*d-0.002*q13+0.001*q5+0.001*q7)
      g(3)=q6+20*c
      gj(3,2)=0.02*c
      gj(3,3)=1.0
      gj(3,4)=gj(3,2)
      gj(3,5)=-0.4*c
      gj(3,6)=gj(3,5)
      gj(3,7)=gj(3,6)
      gj(3,8)=-0.04*c
c
c *** bafil3
c
      c=exp(-0.02*d-0.005*q13+0.001*q5+0.002*q6)
      g(4)=q7+40*c
      gj(4,2)=0.04*c
      gj(4,3)=0.08*c
      gj(4,4)=1.0
      gj(4,5)=-0.8*c
      gj(4,6)=gj(4,5)
      gj(4,7)=gj(4,6)
      gj(4,8)=-0.2*c
c
c *** qualil
c
      g(5)=q4*c3-q9*c2
      gj(5,1)=c3
      gj(5,6)=-c2
      gj(5,9)=-q9
      gj(5,10)=q4
200   return
      end
```

```
      subroutine object(mode,n,x,f,g,nstate,nprob)
c
c *** calculation of objective functions
c
c *** mode   - if mode=-1 termination
c *** n      - number of nonlinear variables
c *** x      - values of nonlinear variables
c *** f      - objective function
c *** g      - gradient vector
c *** nstate - status parameter
c *** nprob  - problem number
c
      implicit real*8 (a-h,o-z)
      dimension x(n),g(n),v(15)
      character*1 l
      character*8 objnam,rhs,bds
      common/help/nwcore,rho,rhs,bds,l(80),nrun
      common/rfp/nc,objnam(100),gam(100),
     *           rfp(100),obj(100),dif(100)
      common/utopia/objmin(100),objmax(100)
      common z(100000)
      equivalence (v(1),q4),(v(2),q5),(v(3),q6),(v(4),q7),
     *            (v(5),q8),(v(6),q9),(v(7),q10),(v(8),q13),
     *            (v(9),c2),(v(10),c3),(v(11),q3),(v(12),q2),
     *            (v(13),q11),(v(14),c1),(v(15),q12)
c
c *** Insert here the criteria functions in FORTRAN-statements.
c
c
c *** This is the testproblem test2 with nonlinear criteria
c *** functions and nonlinear constraints
c
      call valist(n,x,v)
c
      obj(1)=1000.0-q8-q11-q12-q13
      obj(2)=1000.0-q4
      obj(3)=(1.0+0.01*c1)*q8+q11+1.5*q12+q13+500.0
      obj(4)=2*q8+q9+1.5*q10+400.0
      obj(5)=0.01*q4*c3+500.0
c
      if (nstate .eq. 2 ) return
      do 1 i=1,n
          g(i)=0.
1     continue
      if (nrun .ne. 1 ) goto 20
      do 2 i=1,nc
          if(gam(i).gt.0.0001) k=i
2     continue
c
c *** quadratic scalarizing function is used for the calculation
c *** of the decision support matrix.
c
      c=rfp(k)
      if(dabs(c).lt.1.) c=1.
      f=obj(k)/c
```

```
c
c *** computation of gradients
c
      d=2.*f/c
      call objgra(n,k,d,x,g)
c
      f=f*f
      return
c
c *** The automatic scaled achievement variables are calculated.
c
20    if (nstate.ne.1) goto 60
      do 30 i=1,nc
          if (rfp(i) .le. objmin(i)) goto 40
          dif(i)=.5*objmin(i)
30    continue
      goto 60
40    continue
      do 50 i=1,nc
          dif(i)=.5*rfp(i)
50    continue
c
c *** The achievement scalarizing function has to be inserted
c
60    s=.0
      do 70 i=1,nc
          d=gam(i)/(dif(i)-rfp(i))
          w=(dif(i)-obj(i))*d
          s=s+w**rho
          d=-rho*w**(rho-1)*d
c
c *** computation of gradients
c
          call objgra(n,i,d,x,g)
c
70    continue
      s=s/nc
      f=s**(1/rho)
      d=s**(1/rho-1)/rho/nc
      do 80 i=1,n
          g(i)=g(i)*d
80    continue
      return
      end
c
c *************
c
      subroutine valist(n,x,v)
c
c *** association of actual variables
c
      real*8 x(n),v(n)
      do 1 i=1,n
          v(i)=x(i)
1     continue
      return
      end
```

```
      subroutine objgra(n,k,d,x,g)
c
c *** calculation of objective gradients
c
c *** n       - number of nonlinear variables
c *** k       - index of actual objective
c *** d       - factor
c *** x       - values of nonlinear variables
c *** g       - gradient vector
c
      implicit real*8 (a-h,o-z)
      dimension g(n),x(n),v(15)
      equivalence (v(1),q4),(v(2),q5),(v(3),q6),(v(4),q7),
     *            (v(5),q8),(v(6),q9),(v(7),q10),(v(8),q13),
     *            (v(9),c2),(v(10),c3),(v(11),q3),(v(12),q2),
     *            (v(13),q11),(v(14),c1),(v(15),q12)
c
      call valist(n,x,v)
c
c *** programing of gradients in the following form:
c *** g(i)=g(i)+(partial object. funct. over partial v(i))*d
c
      goto (1,2,3,4,5),k
c
c *** obj1
c
1           g(5)=g(5)-1.*d
            g(8)=g(8)-1.*d
            g(13)=g(13)-1.*d
            g(15)=g(15)-1.*d
      goto 6
c
c *** obj2
c
2           g(1)=g(1)-1.*d
      goto 6
c
c *** obj3
c
3           g(5)=g(5)+(1.0+0.01*c1)*d
            g(8)=g(8)+1.*d
            g(13)=g(13)+1.*d
            g(14)=g(14)+0.01*q8*d
            g(15)=g(15)+1.5*d
      goto 6
c
c *** obj4
c
4           g(5)=g(5)+2.*d
            g(6)=g(6)+1.*d
            g(7)=g(7)+1.5*d
      goto 6
c
c *** obj5
c
5           g(1)=g(1)+0.01*c3*d
            g(10)=g(10)+0.01*q4*d
6     return
      end
```

APPENDIX 3: INPUT AND OUTPUT FOR TEXT EXAMPLES

3.1: TEST1 (Section 4.1)

3.2: TEST2 (Section 4.2)

APPENDIX 3.1: TEST1

```
% more specs.tl
begin      testl

   minimize
   nonlinear constraints     0
   nonlinear jacobian vars  12
   nonlinear objectiv vars  12

   bou                bnd
   rhs                rhs
   rows                    20
   columns                 20
   elements               100

   objective = object
   problem no.              1
   mps file                 9
   solution               yes

   aijtol             0.000001
   difference intervall   1.0e-06
   dj tolerance        1.0e-6
   feasibility tol     1.0e-5
   linesearch toler      0.1
   lower bound            0.
   iterations           1000
   major iterations       10
   minor iterations       20
   penalty parameter     0.1
   radius of conver     0.01
   row tolerance       1.0e-6
   superbasics            12
   hessian dimension      12
   jacobian            dense

   print level (jflxi)     1

   derivative level        2

   call function routines when optimal

end
% more rfp.tl
  objl          25.0        1.000          24.0
  obj2          50.0        1.000
  obj3          45.0        1.000
  ....
```

```
% more model.tl
name           testl
rows
 e  gl1
 e  gl2
 e  gl3
 e  gl4
 e  gl5
 e  gl6
 e  gl7
 l  ugl1
 l  ugl2
 l  ugl3
 l  ugl4
 l  ugl5
columns
    x1         gl1          2.0
    x1         gl2          1.0
    x1         ugl1         2.0
    x1         ugl5         3.0
    x2         gl1          0.5
    x2         gl2          2.0
    x2         ugl1         1.0
    x2         ugl5         2.0
    x3         gl3          1.0
    x3         gl4          0.5
    x4         gl5          1.0
    x4         gl6          2.0
    x4         gl7          3.0
    x4         ugl3         5.0
    x4         ugl4         3.0
    x5         gl5          0.5
    x5         gl6          1.0
    x5         gl7         -1.0
    x5         ugl3         3.0
    x5         ugl4         2.0
    x6         gl1         -1.0
    x6         gl3          0.5
    x6         gl4          1.0
    x6         ugl2         2.0
    x7         gl2         -1.0
    x8         gl3         -1.0
    x8         gl5          0.5
    x8         gl6         -1.0
    x8         gl7          1.0
    x8         ugl4         3.0
    x9         gl4         -1.0
    x10        gl5         -1.0
    x11        gl1          1.0
    x11        gl2          1.0
    x11        gl6         -1.0
    x11        ugl1         2.0
    x12        gl3         -1.0
    x12        gl4          1.0
    x12        gl7         -1.0
    x12        ugl2         3.0
rhs
    rhs1       gl1          5.0
    rhs1       ugl1         8.0
    rhs1       ugl2        12.0
    rhs1       ugl3        15.0
    rhs1       ugl4        12.0
    rhs1       ugl5        13.0
    rhs        gl1          6.0
    rhs        ugl1         9.0
    rhs        ugl2        13.0
    rhs        ugl3        16.0
    rhs        ugl4        13.0
    rhs        ugl5        14.0
bounds
 up bndx       x1           2.0
 up bndx       x2           6.0
 up bndx       x3           3.0
 up bndx       x4           2.0
 up bndx       x5           4.0
 up bndx       x6           4.0
 up bndx       x8           3.0
 up bndx       x11          3.0
 up bndx       x12          2.0
 up bnd        x1           3.0
 up bnd        x2           7.0
 up bnd        x3           4.0
 up bnd        x4           3.0
 up bnd        x5           5.0
 up bnd        x6           5.0
 up bnd        x8           4.0
 up bnd        x11          4.0
 up bnd        x12          3.0
 fx initial    x1           1.9
 fx initial    x2           1.6
 fx initial    x3           1.7
 fx initial    x4           0.4
 fx initial    x5           1.2
 fx initial    x6           0.6
 fx initial    x7           6.1
 fx initial    x8           1.0
 fx initial    x9           2.45
 fx initial    x10          1.5
 fx initial    x11          1.0
 fx initial    x12          1.0
endata
```

```
% more sol.tl
testl, linear constrained quadratic programming problem
Fri Jan  6 10:25:23 1984

   calculation of efficient points
   -------------------------------
 objective  scale     reference    efficient     utopia       nadir
   names                point        point        point        point
-------------------------------------------------------------------------
  objl        1.0      25.000       31.642       24.019       42.500
  obj2        1.0      50.000       63.693       38.312       128.00
  obj3        1.0      45.000       59.463       48.863       108.99

   name         value              lower limit     upper limit
end constraints


   name         value              lower limit     upper limit
 x1            2.3117                0.             3.00000
 x2                0.                0.             7.00000
 x3            3.2332                0.             4.00000
 x4           0.78898                0.             3.00000
 x5            2.4928                0.             5.00000
 x6           0.81169                0.             5.00000
 x7            4.5000                0.               none
 x8            1.8825                0.             4.00000
 x9            4.1849                0.               none
 x10           2.9766                0.               none
 x11           2.1883                0.             4.00000
 x12           1.7566                0.             3.00000
 rhs         -1.00000            -1.00000          -1.00000
end variables

%
%
%
% more range.tl
  objl       24.019       42.500
  obj2       38.312       128.00
  obj3       48.863       108.99
%
% dida tl.run
                       d i d a s s
       a dynamic and interactive
  decision analysis and support system
      nonlinear version jan. 1984
  ************************************

   enter the problem name ( < 1 line, 80 char.)
testl, linear constrained quadratic programming problem
   calculation of extreme points ?
   enter ( yes or no ) !
            *      *
y
```

```
  selfish-optimization for all objectives
        * decision support matrix *
 -the diagonal represents the utopia point-
 -------------------------------------------
   i objnam(i)      obj(1)   ...
   1  obj1     |  24.019    |  89.699     |  92.138     |
   2  obj2     |  38.563    |  38.312     |  108.99     |
   3  obj3     |  42.500    |  128.00     |  48.863     |

   generation of efficient points
   ------------------------------
 You can now:
   list the reference point file, enter ( list )
                                           *
   exit from editing status,      enter ( exit )
                                           *
   change the reference point,    enter ( rfp  )
                                           *
   copy last efficient points,    enter ( copy )
                                           *
   change the scale coefficients, enter ( scal )
                                           *
   change the arbitrary coeff.,   enter ( arbi )
                                           *
l
 objective  reference   scale  rho
   names      points
--------------------------------
  obj1        25.000      1.0  24.0
  obj2        50.000      1.0
  obj3        45.000      1.0

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
e
   change the rhs and/or bounds set?
   enter ( yes or no ) !
           *      *
n
   calculation of efficient points
   ------------------------------
 objective  scale     reference   efficient     utopia        nadir
   names                point       point        point         point
------------------------------------------------------------------
  obj1        1.0      25.000      31.642       24.019        42.500
  obj2        1.0      50.000      63.693       38.312        128.00
  obj3        1.0      45.000      59.463       48.863        108.99

   output of variables and constraints?
   enter ( yes or no ) !
           *      *
y
 output of all constraints?
   enter ( yes or no ) !
           *      *
n
```

```
  name        value            lower limit     upper limit

 enter name of constraint or ( * ), if exit
*

 output of all variables?
   enter ( yes or no ) !
           *      *
y

   name        value            lower limit     upper limit
 x1           2.3117              0.              3.00000
 x2               0.              0.              7.00000
 x3           3.2332              0.              4.00000
 x4          0.78898              0.              3.00000
 x5           2.4928              0.              5.00000
 x6          0.81169              0.              5.00000
 x7           4.5000              0.                 none
 x8           1.8825              0.              4.00000
 x9           4.1849              0.                 none
 x10          2.9766              0.                 none
 x11          2.1883              0.              4.00000
 x12          1.7566              0.              3.00000
 rhs        -1.00000          -1.00000          -1.00000

 output of variables/constraints finished?
   enter ( yes or no ) !
           *      *
y
   calculation of efficient points with edited data?
   enter ( yes or no ) !
           *      *
n
21.4u 11.8s 19:45 2% 102+52k 247+217io 239pf+0w
```

APPENDIX 3.2: TEST2

```
% more specs.tc2
begin      test2

   minimize
   nonlinear constraints      5
   nonlinear jacobian vars  10
   nonlinear objectiv vars  15

   bounds                    bnd
   rhs                       rhs
   rows                       20
   columns                    20
   elements                  100

   objective = object
   problem no.                 2
   mps file                    9
   solution                  yes
   verify                    yes

   aijtol               0.000001
   difference intervall   1.0e-06
   dj tolerance         1.0e-6
   feasibility tol      1.0e-6
   linesearch toler        0.1
   lower bound             0.
   iterations            1000
   major iterations        19
   minor iterations        29
   penalty parameter      0.1
   radius of conver      0.01
   row tolerance        1.0e-6
   superbasics             12
   hessian dimension       12
   jacobian             dense

   print level (jflxi)      1

   derivative level         3

   call function routines when optimal

end

% more rfp.t2
  obj1          900.        1.000          2.0
  obj2          900.        1.000
  obj3          600.        1.000
  obj4          600.        1.000
  obj5          600.        1.000
  ....
```

```
%
% more model.t2
name          test2
rows
 g  gwtab1
 e  bafil1
 e  bafil2
 e  bafil3
 g  quali1
 l  fluba1
 g  fluba2
 g  fluba3
 l  gwtab2
 g  quali2
 g  quali3
columns
    q4          quali1
    q4          fluba3          -1.0
    q5          gwtab1
    q5          bafil1
    q5          bafil2
    q5          bafil3
    q5          fluba1           1.0
    q5          gwtab2          -0.01
    q6          gwtab1
    q6          bafil1
    q6          bafil2
    q6          bafil3
    q6          fluba2          -1.0
    q6          gwtab2          -0.02
    q7          gwtab1
    q7          bafil1
    q7          bafil2
    q7          bafil3
    q7          fluba3          -1.0
    q7          gwtab2          -0.03
    q8          gwtab1
    q8          bafil1
    q8          bafil2
    q8          bafil3
    q8          gwtab2           0.1
    q8          quali2          -0.1
    q9          gwtab1
    q9          bafil1
    q9          bafil2
    q9          bafil3
    q9          quali1
    q9          fluba3           1.0
    q9          gwtab2           0.1
    q9          quali3          -0.2
    q10         gwtab1
    q10         bafil1
    q10         bafil2
    q10         bafil3
    q10         gwtab2           0.1
    q10         quali3          -0.2
    q13         gwtab1
    q13         bafil1
    q13         bafil2
    q13         bafil3
    q13         gwtab2           0.2
    c2          quali1
    c2          quali3           1.0
    c3          quali1
    q3          fluba3           1.0
    q3          fluba2          -1.0
    q2          fluba1           1.0
    q2          fluba2           1.0
    q11         fluba2          -1.0
    c1          quali2           1.0
    q12
rhs
    rhs         gwtab1          20.0
    rhs         bafil1          27.0
    rhs         bafil2          22.2
    rhs         bafil3          44.05
    rhs         quali1           0.0
    rhs         fluba1         150.0
    rhs         fluba2           0.0
    rhs         fluba3           0.0
    rhs         gwtab2          20.0
    rhs         quali2         100.0
    rhs         quali3         200.0
bounds
 up bnd         q4             200.0
 up bnd         q5             200.0
 up bnd         q6             200.0
 up bnd         q7             200.0
 up bnd         q8             200.0
 up bnd         q9             200.0
 up bnd         q10            200.0
 up bnd         q13            200.0
 up bnd         c2            1000.0
 up bnd         c3             200.0
 up bnd         q3             200.0
 up bnd         q2             200.0
 up bnd         q11            200.0
 up bnd         c1             500.0
 up bnd         q12            200.0
 lo bnd         q4               0.0
 lo bnd         q5               0.0
 lo bnd         q6               0.0
 lo bnd         q7               0.0
 lo bnd         q8               0.0
 lo bnd         q9               0.0
 lo bnd         q10              0.0
 lo bnd         q13              0.0
 lo bnd         c2               0.0
 lo bnd         c3               0.0
 lo bnd         q3               0.0
 lo bnd         q2               0.0
```

```
 lo bnd        q11          0.0
 lo bnd        c1           0.0
 lo bnd        q12          0.0
 fx initial    q4         100.0
 fx initial    q5          20.0
 fx initial    q6          20.0
 fx initial    q7          40.0
 fx initial    q8          30.0
 fx initial    q9          60.0
 fx initial    q10         20.0
 fx initial    q13         30.0
 fx initial    c2         216.0
 fx initial    c3         130.0
 fx initial    q3          80.0
 fx initial    q2         130.0
 fx initial    q11         30.0
 fx initial    c1         103.0
 fx initial    q12          0.0
endata

% dida_tc2.run
                d i d a s s
     a dynamic and interactive
  decision analysis and support system
     nonlinear version jan. 1984
  ************************************

   enter the problem name ( < 1 line, 80 char.)
test2, nonlinear constraints and nonlinear objective functions
   calculation of extreme points ?
   enter ( yes or no ) !
           *      *
y
  selfish-optimization for all objectives
      * decision support matrix *
 -the diagonal represents the utopia point-
 -----------------------------------------
```

i	objnam(i)	obj(1)	...			
1	obj1	578.31	975.78	1211.3	799.91	548.43
2	obj2	920.14	800.00	631.35	668.98	900.00
3	obj3	1000.00	879.63	500.00	699.65	647.38
4	obj4	931.70	847.88	568.30	482.15	804.24
5	obj5	807.68	1000.00	810.35	712.69	500.00

```
   generation of efficient points
   -----------------------------
 You can now:
   list the reference point file, enter ( list )
                                          *
   exit from editing status,      enter ( exit )
                                          *
   change the reference point,    enter ( rfp  )
                                          *
   copy last efficient points,    enter ( copy )
                                          *
   change the scale coefficients, enter ( scal )
                                          *
   change the arbitrary coeff.,   enter ( arbi )
                                          *
```

```
l
 objective  reference   scale  rho
   names      points
--------------------------------
  obj1       900.00       1.0   2.0
  obj2       900.00       1.0
  obj3       600.00       1.0
  obj4       600.00       1.0
  obj5       600.00       1.0

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
a
   enter new value for rho
24
 old val.   2.0000      new val.   24.000

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
r
   enter obj.name or ( * ),if exit
obj1
   enter new rfp. value
850
 old val.   900.00      new val.   850.00
   enter obj.name or ( * ),if exit
*

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
l
 objective  reference   scale  rho
   names      points
--------------------------------
  obj1       850.00       1.0  24.0
  obj2       900.00       1.0
  obj3       600.00       1.0
  obj4       600.00       1.0
  obj5       600.00       1.0

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
e
   change the rhs and/or bounds set?
   enter ( yes or no ) !
           *      *
n
   calculation of efficient points
   -------------------------------
 objective  scale     reference  efficient     utopia        nadir
   names                point      point        point        point
-------------------------------------------------------------------
  obj1        1.0      850.00     891.85       578.31      1000.00
  obj2        1.0      900.00     909.70       800.00      1000.00
  obj3        1.0      600.00     612.04       500.00       1211.3
  obj4        1.0      600.00     488.96       482.15       799.91
  obj5        1.0      600.00     589.19       500.00       900.00
```

```
   output of variables and constraints?
   enter ( yes or no ) !
           *       *
y
 output of all constraints?
   enter ( yes or no ) !
           *       *
y
```

name	value	lower limit	upper limit
gwtabl	20.000	20.00000	none
bafil1	27.000	27.00000	27.00000
bafil2	22.200	22.20000	22.20000
bafil3	44.050	44.05000	44.05000
quali1	0.	0.	none
fluba1	150.00	none	150.00000
fluba2	0.	0.	none
fluba3	0.	0.	none
gwtab2	20.000	none	20.00000
quali2	100.000	100.00000	none
quali3	200.00	200.00000	none

```
 output of all variables?
   enter ( yes or no ) !
           *       *
y
```

name	value	lower limit	upper limit
q4	90.305	0.	200.00000
q5	17.363	0.	200.00000
q6	17.963	0.	200.00000
q7	37.225	0.	200.00000
q8	0.	0.	200.00000
q9	41.554	0.	200.00000
q10	31.604	0.	200.00000
q13	71.669	0.	200.00000
c2	214.63	0.	1000.00000
c3	98.764	0.	200.00000
q3	85.975	0.	200.00000
q2	132.64	0.	200.00000
q11	28.699	0.	200.00000
c1	100.000	0.	500.00000
q12	7.7843	0.	200.00000
rhs	-1.00000	-1.00000	-1.00000

```
 output of variables/constraints finished?
   enter ( yes or no ) !
           *       *
y
   calculation of efficient points with edited data?
   enter ( yes or no ) !
           *       *
y
```

```
   generation of efficient points
   ------------------------------
 You can now:
   list the reference point file, enter ( list )
                                          *
   exit from editing status,      enter ( exit )
                                          *
   change the reference point,    enter ( rfp  )
                                          *
   copy last efficient points,    enter ( copy )
                                          *
   change the scale coefficients, enter ( scal )
                                          *
   change the arbitrary coeff.,   enter ( arbi )
                                          *
c
 previous solution copied as rfp

 objective  reference   scale  rho
   names      points
--------------------------------
  obj1       891.85       1.0  24.0
  obj2       909.70       1.0
  obj3       612.04       1.0
  obj4       488.96       1.0
  obj5       589.19       1.0

   enter ( list | copy | scal | arbi | rfp | exit ) !
           *      *      *      *      *     *
e
   change the rhs and/or bounds set?
   enter ( yes or no ) !
           *      *
n
   calculation of efficient points
   -------------------------------
 objective  scale     reference   efficient    utopia      nadir
   names                point       point       point      point
-------------------------------------------------------------------
  obj1        1.0       891.85      903.94      578.31    1000.00
  obj2        1.0       909.70      892.08      800.00    1000.00
  obj3        1.0       612.04      604.90      500.00     1211.3
  obj4        1.0       488.96      491.00      482.15     799.91
  obj5        1.0       589.19      579.55      500.00     900.00

   output of variables and constraints?
   enter ( yes or no ) !
           *      *

 n
 82.9u 14.0s 8:47 18% 142+116k 215+535io 194pf+0w
```