IAC – DIDAS – N
A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models, v. 4.0

Tomasz Kręglewski, Janusz Granat, Andrzej P. Wierzbicki

CP-91-010
June 1991

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

This Collaborative Paper is one of a series which presents the different software packages designed and implemented for interactive decision support. These packages constitute the outcome of the contracted study agreement between the System and Decision Sciences Program at IIASA and several Polish scientific institutions. The theoretical part of these results is presented in the IIASA Collaborative Paper CP-90-008 entitled *Contributions to Methodology and Techniques of Decision Analysis (First Stage)*, edited by Andrzej Ruszczyński, Tadeusz Rogowski and Andrzej P. Wierzbicki.

The distributable versions of the software are usually tailored for the illustration of methodology and possible applications. However, for most of these software packages there exists a version made for a specific application and it is possible to modify each software package for a specific real-life application (if the corresponding mathematical programming model is of the type for which a particular package has been designed).

All software developed within the scientific cooperation mentioned above is available either at distribution cost or free of charge for scientific non-commercial usage by institutions and individuals from the countries which are members of IIASA. Inquiries about more detailed information and requests for the software should be addressed to the Leader of the MDA Project.

This volume contains the theoretical and methodological backgrounds as well as the User's Guide for a version of decision analysis and support systems of DIDAS family that is designed for multicriteria analysis of nonlinear models, implemented for IBM compatible personal computer.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

Abstract

This paper presents introductive and user documentation — including extended summary, theoretical manual, short user manual and description of illustrative examples — for a version of decision analysis and support systems of DIDAS family that is designed for multicriteria analysis of nonlinear models on professional microcomputers. This version has been developed in the years 1986–1990 in the Institute of Automatic Control, Warsaw University of Technology, under a joint research program with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis. It can be run on professional microcomputers compatible with IBM--PC--XT or AT (with Hercules Graphics Card, Color Graphics Adapter or Enhanced Graphics Adapter and, preferably, with a numeric coprocessor and a hard disk) and supports graphical representation of results of interactive multicriteria analysis. Moreover, this version called IAC-DIDAS-N is provided with a new nonlinear model generator and editor that support, in an easy standard of a spreadsheet, the definition, edition and symbolic differentiation of nonlinear substantive models for multiobjective decision analysis. A specially introduced standard of defining nonlinear programming models for multiobjective optimization helps to connect the model generator with other parts of the system. Optimization runs involved in interactive, multiobjective decision analysis are performed by a solver, that is, a version of nonlinear programming algorithm specially adapted for multiobjective problems. This algorithm is based on shifted penalty functions and projected conjugate directions techniques similarly as in former nonlinear versions of DIDAS, but it was further developed and several improvements were added. The system is permanently updated and developed. Currently (starting from October 1990) the version 4.0 of the system is released. Most of enhancements added in this version are not directly visible to the user. They influence the efficiency of the system.

Contents

IAC – DIDAS – N
A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models, v. 4.0

*Tomasz Kręglewski, Janusz Granat, Andrzej P. Wierzbicki**

1 Extended summary

In many complex decision problems involving economic, environmental and technological decisions as well as in complex engineering design, decision maker needs some help of an analyst or a team of them to learn about possible decision options and their predicted results. The team of analysts frequently summarizes its knowledge in the form of a *substantive model* of the decision situation that can be formalized mathematically and computerized. Although such a model can never be perfect and cannot encompass all aspects of the problem, it is often a great help to the decision maker in the process of learning about novel aspects of the decision situation and thus gaining expertise in handling problems of a given class. Even if the final decisions are typically made judgementally — that is, are based on holistic, deliberative assessments of all available information without performing a calculative analysis of this information, see (Dreyfus, 1984) — the interaction of the decision maker and the team of analysts with substantive models prepared by them can be of great value when preparing such decisions.

In organizing such interaction, many techniques of optimization, multicriteria decision analysis and other tools of mathematical programming can be used. To be of value for a holistically thinking decision maker, however, all such techniques must be used as supporting tools of interactive analysis rather than as means for proposing unique optimal decisions and thus replacing the decision maker. The decision analysis and support systems of DIDAS family — that is, Dynamic Interactive Decision Analysis and Support systems, see e.g. (Lewandowski et al., 1983, 1987) — are specially designed to support interactive work with a substantive model while using multicriteria optimization tools, but they stress the learning aspects of the work, such as the right of a decision maker to change his priorities and preferences after learning new facts. DIDAS systems can be used either by analysts who want to analyze their substantive models, or by teams of analysts and decision makers, or even by decision makers working alone with a previously defined substantive model; in any case, we shall speak further about *the user* of the system.

There are several classes of substantive models that all require special technical means of support — see (Lewandowski et al., 1987). The IAC-DIDAS-N version is designed to support models of multiobjective nonlinear programming type. Although some nonlinear

*Institute of Automatic Control, Warsaw University of Technology, Poland

DIDAS versions were developed before, they did not follow any standards of defining such models, since such standards did not exist. In order to support the work with a user that is not a specialist in computer programming and nonlinear optimization programming, it has become necessary to introduce such standards.

Models of multiobjective nonlinear programming type specify, firstly, the following classes of variables: *input variables* that can be subdivided into *decision variables* (that is, means of multiobjective optimization) and *parametric variables* (that is, model parameters that are kept constant during multiobjective analysis but may be changed during parametric or sensitivity analysis) — and *outcome variables* that can be subdivided into *floating outcomes* (used either as model constraints or only for the easiness of definition of the nonlinear model or even only as additional information for the user) and *optimized outcomes* or *objectives* (the ends of multiobjective optimization that can be either maximized or minimized or stabilized, that is, kept close to a desired level). Actually, the distinction between various types of outcome variables is not necessarily sharp as the user may change their classification and select his objectives among various outcome variables when defining the multiobjective analysis problem.

For all input and outcome variables, a reasonably defined nonlinear model should include *lower* and *upper bounds*, that is, reasonable ranges of admissible changes of these variables. Moreover, an essential part of a nonlinear model definition are *model equations*, that is, nonlinear functions that define the dependence of all outcome variables on input variables. To make the model definition easier for the user, it is assumed that outcome variables are defined consecutively and that they can depend not only on input variables, but also on previously defined outcome variables. However, all outcome variables must be defined explicitly.

There are many examples of decision problems that can be analyzed by the use of a substantive model of multiobjective nonlinear programming type; for example, DIDAS-type systems with multiobjective nonlinear programming models were used in analyzing various environmental or technological problems (see Kaden, 1985, Grauer et al., 1983). As a demonstrative or tutorial example, IAC-DIDAS-N uses a multiobjective nonlinear programming model of acid deposition in forest soil (see Hettelingh and Hordijk, 1987). The user can also define substantive models of multiobjective nonlinear programming type for his own problems and analyze them with the help of IAC-DIDAS-N.

A typical procedure of working with the IAC-DIDAS-N system consists of several phases. In the first phase a user, mostly an analyst, defines the substantive model and edits it on the computer. In earlier versions of nonlinear DIDAS-type systems (which were mostly implemented on bigger mainframe computers) this phase was not explicitly supported in the system and the user had to separately prepare (define and edit) his nonlinear model, typically in the form of a FORTRAN procedure that contained also user-supplied formulae for the derivatives of all outcome functions with respect to decision variables. It is a known fact that most mistakes in applying nonlinear programming methods are made when determining derivatives analytically; thus, this way of substantive model preparation required rather much experience in applications of nonlinear programming.

The new features of IAC-DIDAS-N are, firstly, the definition and edition of substantive models in an easy but flexible standard format of a spreadsheet, where the input variables correspond to spreadsheet columns and the outcome variables — to spreadsheet rows; special cells are reserved for types of variables, lower and upper bounds on all variables, as well as reference levels (reservation levels for stabilized outcomes, aspiration and reservation levels for maximized and minimized outcomes) and results of various optimization computations, etc. However, another unique new feature of IAC-DIDAS-N is an

automatic support of calculations of all needed derivatives by a symbolic differentiation program. The user needn't laboriously calculate many derivatives and check whether he has not made any mistake; he must only define model equations or outcome functions (whereas a recursive, but explicit form of such functions is allowed) and make sure that these functions are differentiable and admissible for the symbolic differentiation program — that admits functions from a rather wide class. Moreover, the spreadsheet format allows also to display the automatically determined formulae for derivatives. The size of substantive models that can be defined in the spreadsheet is limited only by the size of microcomputer memory, but reasonable models of nonlinear programming type that can be usefully analyzed on microcomputers should not be too large anyway. The user of IAC-DIDAS-N can also have several substantive models recorded in special model directories, use old models to speed up the definition of a new model, etc., while the system supports automatically the recording of all new or modified models in the appropriate directory.

In further phases of the work with DIDAS-type systems, the user — here typically an analyst working together with the decision maker — specifies a multiobjective analysis problem related to his substantive model and participates in an initial analysis of this problem. There may be many multiobjective analysis problems related to the same substantive model. The specification of a multiobjective problem consists in designating optimized outcomes (objectives) among outcome variables, defining whether an objective should be minimized, or maximized, or stabilized — kept close to a given level. Moreover, the user can also shift bounds on any outcome when specifying a multiobjective analysis problem.

For a given definition of the multiobjective analysis problem, the decisions and outcomes in the model are subdivided into two categories: these that are *efficient* with respect to the multiobjective problem (that is, such that no objective can be improved without deteriorating some other objective) and those that are inefficient. It is assumed that the user is interested only in efficient decisions and outcomes (this assumption is reasonable provided he has listed all objectives of his concern; if he has not, or if some objectives of his concern are not represented in the model, he can still modify the sense of efficiency by adding new objectives, or by requiring some objectives to be kept close to given levels, or by returning to the model definition phase and modifying the model).

One of the main functions of DIDAS-type systems is computation of efficient decisions and outcomes — interactively following various instructions of the user — and their presentation for analysis. This is done by solving a special parametric nonlinear programming problem resulting from the specification of the multiobjective analysis problem; for this purpose, IAC-DIDAS-N contains a specialized nonlinear programming algorithm called *solver*. Following the experience with previous versions of nonlinear DIDAS systems, a robust nonlinear programming algorithm, based on shifted penalty functions and projected conjugate directions techniques, was further developed for IAC-DIDAS-N.

A multiobjective problem definition usually admits many efficient decisions and outcomes; the user should first learn about *ranges* of changes of outcomes and *bounds* on *efficient outcomes*. Calculations of these bounds is the main function of IAC-DIDAS-N in the initial analysis phase. The user can request the system to optimize any objective separately; however, there is also special command that automatically performs all necessary calculations.

The command "utopia" results in subsequent computations of the best possible outcomes for all objectives treated separately (such outcomes are practically never attainable jointly, hence the name *utopia point* for the point in outcome space composed of such

outcomes). During "utopia" calculations some approximations of worst possible efficient values are also obtained. The point in outcome space composed of the worst efficient values is called *nadir point*, however its exact calculation is a very difficult computational task — for nonlinear models there is even no constructive method for such calculation. The approximation of nadir point components obtained during utopia point calculations is rather too optimistic. The decision maker or the analyst can change, according to their knowledge, obtained nadir values.

The utopia and nadir points give important information to the user about reasonable ranges of (efficient) decision outcomes; in order to give him also information about a reasonable compromise efficient solution, a *neutral efficient solution* can also be computed in the initial analysis phase due to a special command. The neutral solution is an efficient solution situated 'in the middle' of the range of efficient outcomes; the precise meaning of being 'in the middle' is defined by the distances between the utopia and (the approximation of) the nadir point components. After analyzing the utopia point, the nadir point and the neutral solution (which all can be represented graphically for the user), the initial analysis is completed and the user has already learned much about ranges of attainable efficient objectives and the possible trade-off between these objectives. Each change of the definition of the substantive model or of the multiobjective analysis problem, however, actually necessitates a repetition of the initial analysis phase.

The third phase of the work with the IAC-DIDAS-N system consists in interactive scanning of efficient outcomes and decisions, guided by the user who specifies two *reference points* called *reservation point* and *aspiration point* in the objective space, i.e. *reservation levels* and *aspiration levels* for each objective; the system admits also for a more simple option of specifying only one reference (aspiration or reservation) level for some or even for all objectives. The user already has reasonable knowledge about the range of possible outcomes and thus, he can specify his reference levels: aspiration levels that he would like to attain and reservation levels that he would like to satisfy in any case. The utopia and the nadir points could be used as initial values for the aspiration point and the reservation point, respectively. However, because the neutral solution has also been calculated, the system suggests to the user another, more adequate initial aspiration point: an unattainable outcome point closer to the efficient solutions than the utopia point, and more adequate initial reservation point: an attainable outcome closer to the efficient solutions than the nadir point.

IAC-DIDAS-N utilizes the aspiration and the reservation levels as parameters in a special achievement function coded in the system, uses its solver to compute the solution of a nonlinear programming problem equivalent to maximizing this achievement function, and responds to the user with an attainable, efficient solution and outcomes that strictly correspond to the user specified references.

If the aspirations are not attainable and the reservations are attainable (that is a typical and recommended case), then the response of the system is a solution with attainable, efficient outcomes that are either between the aspiration and reservation points or uniformly as close as possible to the former one. If the aspirations are 'too low' (if they correspond to attainable but inefficient outcomes that can be improved), then the response of the system is a solution with outcomes that are uniformly better than the aspirations. If the reservations are 'too high' (if they correspond to outcomes that are not attainable), then the response of the system is an efficient solution with outcomes that are uniformly worse than the reservations. The precise meaning of the uniform approximation or improvement depends on *scaling units* for each objective that are defined automatically in the system basing on the differences between the utopia point, the current aspiration

point and the current reservation point, therefore, implicitly defined by the user. This automatic definition of scaling units has many advantages to the user who is not only free from specifying scaling units but also has a better control over the selection of efficient outcomes by changing reference levels.

After scanning several representative efficient solutions and outcomes controlled by changing references, the user typically learns enough either to subjectively select an actual decision (which need not correspond to the decisions proposed in the system, since even the best substantive model can differ from real decision situation) or to select an efficient decision proposed by the system as a basis for actual decisions. Rarely, the user can still be uncertain what decision is to be chosen; for this case, several additional options might have been included in a system of **DIDAS**-type. Such options should consist of two more sophisticated scanning rules: a *multidimensional scanning*, resulting from perturbing current aspiration levels along each coordinate of objective space, and a *directional scanning*, resulting from perturbing current aspiration levels along a direction specified by the user (see Korhonen, 1985). Another option is a *forced convergence*, that is, such changes of aspiration levels along subsequent directions specified by the user that the corresponding efficient decisions and outcomes converge to a final point that may represent the best solution for the preferences of the user. However, these additional options have not been implemented in IAC-DIDAS-N, because the experience with **DIDAS**-type systems shows that these options are rarely useful.

2 Theoretical manual

The standard form of a multiobjective nonlinear programming problem is defined as follows:

$$\underset{x \in X}{\text{maximize}} \quad [q = f(x)]; \quad X = \{ x \in R^n : g'(x) = 0, g''(x) \leq 0 \} \tag{1}$$

where $x \in R^n$, $q \in R^p$, $f : R^n \to R^p$ is a given function (assumed to be differentiable), $g' : R^n \to R^{m'}$ and $g'' : R^n \to R^{m''}$ are also given functions (of the same class as f) and the maximization of the vector q of p objectives is understood in the Pareto sense: $\hat{x}, \hat{q}$ are solutions of (1) iff $\hat{q} = f(\hat{x})$, $\hat{x} \in X$ and there are no such x, q with $q = f(x)$, $x \in X$ that $q \geq \hat{q}$, $q \neq \hat{q}$. Such solutions $\hat{x}, \hat{q}$ of (1) are called, respectively, an *efficient decision* $\hat{x}$ and the corresponding *efficient outcome* $\hat{q}$. If, in this definition, it was only required that there were no such x, q with $q = f(x)$, $x \in X$ that $q > \hat{q}$, then the solutions $\hat{x}, \hat{q}$ would be called *weakly efficient*. Equivalently, if the set of all attainable outcomes is denoted by

$$Q = \{ q \in R^p : q = f(x), x \in X \} \tag{2}$$

and so called *positive cones*

$$D = R^p_+ = \{ q \in R^p : q_i \geq 0, i = 1, \ldots, p \}, \quad \widetilde{D} = R^p_+ \setminus \{ 0 \}, \quad \widetilde{\widetilde{D}} = \text{int} R^p_+ \tag{3}$$

are introduced (thus, $q \geq \hat{q}$ can be written as $q - \hat{q} \in D$, $q \geq \hat{q}$, $q \neq \hat{q}$ as $q - \hat{q} \in \widetilde{D}$ and $q > \hat{q}$ as $q - \hat{q} \in \widetilde{\widetilde{D}}$), then the sets of efficient outcomes $\hat{Q}$ and of weakly efficient outcomes $\hat{Q}^w$ can be written as:

$$\hat{Q} = \{ \hat{q} \in Q : (\hat{q} + \widetilde{D}) \cap Q = \emptyset \} \tag{4}$$

$$\hat{Q}^w = \{ \hat{q} \in Q : (\hat{q} + \widetilde{\widetilde{D}}) \cap Q = \emptyset \} \tag{5}$$

where $\emptyset$ denotes an empty set.

The set of weakly efficient outcomes is larger and contains the set of efficient outcomes; in many practical applications, however, the set of weakly efficient outcomes is decisively too large. Some efficient outcomes for multiobjective nonlinear programming problems may have unbounded *trade-off coefficients* that indicate how much an objective outcome should be deteriorated in order to improve another objective outcome by a unit; therefore, it is important to distinguish also a subset $\hat{Q}^p \subset \hat{Q}$ called the set of *properly efficient* outcomes, such that the corresponding trade-off coefficients are bounded.

The *abstract problem of multiobjective nonlinear programming* consists in determining the entire sets $\hat{Q}^p$ or $\hat{Q}$ or $\hat{Q}^w$. The *practical problem of multiobjective decision support* using nonlinear programming models is different and consists in computing and displaying for the decision maker (or, generally, for the user of the decision support system) some selected properly efficient decisions and outcomes. However, a properly efficient outcome with trade-off coefficients that are extremely high or extremely low does not practically differ from a weakly efficient outcome. Thus, some a priori bound on trade-off coefficients should be defined and properly efficient outcomes that do not satisfy this bound should be excluded. This can be done by defining a slightly broader positive cone:

$$D_\varepsilon = \{\, q \in R^p : \mathrm{dist}\,(q, D) \leq \varepsilon \|q\| \,\} \tag{6}$$

where any norm in R^p is used, also for the definition of the distance between q and D. The corresponding, modified definition of D_ε-efficiency:

$$\hat{Q}^{p\varepsilon} = \{\, \hat{q} \in Q : (\hat{q} + \widetilde{D}_\varepsilon) \cap Q = \emptyset \,\}; \quad \widetilde{D}_\varepsilon = D_\varepsilon \setminus \{\, 0 \,\} \tag{7}$$

applies to properly efficient outcomes that have trade-off coefficients a priori bounded by approximately ε and $1/\varepsilon$; such outcomes are also called *properly efficient with* (a priori) *bound* (see Wierzbicki, 1986).

The selection of properly efficient outcomes with bound and the corresponding decisions should be easily controlled by the user and should result in any outcome in the set $\hat{Q}^{p\varepsilon}$ he may wish to attain. Before turning to some further theoretical problems resulting from these practical requirements, observe first that the standard formulation of multiobjective nonlinear programming is not the most convenient for the user. Although many other formulations can be rewritten to the standard form by shifting scales or introducing proxy variables, such reformulations should not bother the user and should be automatically performed in the decision support system. Therefore, we present here another basic formulation of the multiobjective nonlinear programming problem, more convenient for typical applications.

A *substantive model* of multiobjective nonlinear programming type consists of the specification of vectors of n decision variables $x \in R^n$ and of m outcome variables $y \in R^m$ together with nonlinear *model equations* defining the relations between the decision variables and the outcome variables and with *model bounds* defining the lower and upper bounds on all decision and outcome variables:

$$y = g(x); \quad x^{\mathrm{lo}} \leq x \leq x^{\mathrm{up}}; \quad y^{\mathrm{lo}} \leq y \leq y^{\mathrm{up}} \tag{8}$$

where $g : R^n \to R^m$ is a (differentiable) function that combines the functions f, g' and g'' from the standard formulation. Thus, $m = m' + m'' + p$; but the choice, which of the components of the outcome variable y correspond only to constraints and which correspond to objectives, is flexible and can be modified by the user. There are only inequality constraints in the definition of substantive model (8), but equality constraints for some outcomes can be easily written as

$$y_i^{\mathrm{lo}} \leq y_i \leq y_i^{\mathrm{up}} \quad \text{with} \quad y_i^{\mathrm{lo}} = y_i^{\mathrm{up}} \quad \text{for some } i \tag{9}$$

Denote the vector of p objective outcomes by $q \in R^p \subset R^m$ (some of the objective variables may be originally not represented as outcomes of the model, but we can always add them by modifying this model) to write the corresponding objective equations in the form:

$$q = f(x) \tag{10}$$

where f is also composed of corresponding components of g. Thus, the set of attainable objective outcomes is again $Q = f(X)$, but the set of admissible decisions X is defined by:

$$X = \{ x \in R^n : x^{\mathrm{lo}} \leq x \leq x^{\mathrm{up}};\ y^{\mathrm{lo}} \leq g(x) \leq y^{\mathrm{up}} \} \tag{11}$$

Moreover, the objective outcomes are not necessarily maximized; some of them can be minimized, some maximized, some stabilized or kept close to given *stabilization levels* (that is, minimized if their value is above stabilization level and maximized if their value is below stabilization level). All these possibilities can be summarized by introducing a different definition of positive cone D:

$$\begin{aligned} D = \{ q \in R^p :\ & q_i \geq 0, && 1 \leq i \leq p' \ ; \\ & q_i \leq 0, && p'+1 \leq i \leq p'' \ ; \\ & q_i = 0, && p''+1 \leq i \leq p \ \} \end{aligned} \tag{12}$$

where the first p' objectives are to be maximized, the next from $p'+1$ until p'' — minimized, and the last from $p''+1$ until p — stabilized. The definition of the cone D_ϵ does not change its analytical form (6), although the cone itself changes appropriately. Actually, the user has only to define what to do with subsequent objectives; the concept of the positive cones D and D_ϵ is used here only in order to define comprehensively efficient and properly efficient outcomes for the multiobjective problem.

For given some stabilization levels q_i^s for stabilized objectives and the requirement that these objectives should be minimized above and maximized below stabilization levels, the set of efficient outcomes can be defined only relative to the stabilization levels. However, since the user can define stabilization levels arbitrarily, of interest here is the union of such relative sets of efficient outcomes. Let $\widetilde{D} = D \setminus \{\emptyset\}$ and $\widetilde{D_\epsilon} = D_\epsilon \setminus \{\emptyset\}$; then, for arbitrary stabilization levels, the outcomes efficient or properly efficient with bound can be defined, as before, by the relations (4) or (7). The weakly efficient outcomes are of no practical interest in this case, since the cone D typically has empty interior which implies that weakly efficient outcomes coincide with all attainable outcomes.

The stabilized outcomes in the above definition of efficiency are, in a sense, similar to the outcomes with equality constraints (9); however, there is an important distinction between these two concepts. Equality constraints must be satisfied; if not, then there are no admissible solutions for the model. Stabilized objective outcomes should be kept close to stabilization levels, but they can differ from these levels if, through this difference, other objectives can be improved. The user of a decision support system should keep this distinction in mind and can, for example, modify the definition of the multiobjective analysis problem by removing equality constraints for some outcomes and putting these outcomes into the stabilized objective category. Outcomes with inequality constraints can be converted in the same way to either minimized or maximized outcomes.

By adding shifting scales, adding a number of proxy variables and changing the interpretation of the function g, the substantive model formulation (8), (9), (10), (11) together with its positive cone (12) and the related concept of efficiency can be equivalently rewritten to the standard form of multiobjective nonlinear programming (1); this, however, does

not concern the user. More important is the way of the user-controlled selection of an efficient decision and outcome from the set (4) or (7). For stabilized objective outcomes, the user can change the related stabilization levels in order to influence this selection; *it is assumed here that he will do so for all objective outcomes*, that is, *he will use the corresponding reference levels in order to influence the selection of efficient decisions.*

For minimized and maximized objectives the user can specify two kinds of reference levels: *aspiration levels* denoted here $\bar{q}_i$ or $\bar{q}$ as a vector called *aspiration point* and *reservation levels* denoted $\bar{\bar{q}}_i$ or $\bar{\bar{q}}$ as a vector called *reservation point*. The aspiration levels represent the levels that the user would like to attain (although the aspiration point as whole is not attainable in most cases), whereas the reservation levels could be interpreted as 'soft' lower limits for objectives (for maximized objectives; upper limits for minimized objectives). Reservation levels $\bar{\bar{q}}_i$ for maximized objectives should be 'below' the aspiration levels $\bar{q}_i$ ($\bar{\bar{q}}_i < \bar{q}_i$, $i = 1, \ldots, p'$), whereas reservation levels $\bar{\bar{q}}_i$ for minimized objectives should be 'above' the aspiration levels $\bar{q}_i$ ($\bar{\bar{q}}_i > \bar{q}_i$, $i = p' + 1, \ldots, p''$). If these conditions are not satisfied for some objectives, system automatically changes $\bar{q}_i$ or $\bar{\bar{q}}_i$.

For each stabilized objective q_i the user can specify the *lower reservation level* denoted $\bar{\bar{q}}_i^l$ and the *upper reservation level* denoted $\bar{\bar{q}}_i^u$. It is assumed that the stabilization level q_i^s is given implicitly as the mean value of two reservation levels $q_i^s = (\bar{\bar{q}}_i^l + \bar{\bar{q}}_i^u)/2$, thus, the user defines the *reservation range* around the stabilization level. Moreover, the system defines internally the *lower aspiration level* $\bar{q}_i^l = q_i^s - \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2$ and the *upper aspiration level* $\bar{q}_i^u = q_i^s + \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2$, thus, the *aspiration range* is δ times narrower than the reservation range with q_i^s being the center of both ranges. The coefficient δ has the default value 0.1 and can be changed by the user during the interactive process.

The aspiration and reservation points, called jointly *reference points*, are both user-selectable parameters (for minimized and maximized objectives; for stabilized objectives two reservation levels are user-selectable). A special way of parametric scalarization of the multiobjective analysis problem is utilized for the purpose of influencing the selection of efficient outcomes by changing reference points. This parametric scalarization is obtained by maximizing an *order-approximating achievement function* (see Wierzbicki, 1983, 1986). There are several forms of such functions; properly efficient outcomes with approximate bound $\varepsilon, 1/\varepsilon$ are obtained when maximizing a function of the following form:

$$\begin{aligned} s(q, \bar{q}, \bar{\bar{q}}) \;=\; & \min \left(\min_{1 \le i \le p''} z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i), \; \min_{p''+1 \le i \le p} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) + \\ & + \varepsilon \left(\sum_{i=1}^{p''} z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) + \sum_{i=p''+1}^{p} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) , \end{aligned} \tag{13}$$

where the parameter ε should be positive, even if very small; if this parameter is equal to zero, then the above function is not order-approximating any more, but *order-representing*, and its maximal points can correspond to weakly efficient outcomes.

The functions $z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$ for maximized objectives ($i = 1, \ldots, p'$) are defined by:

$$z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) = \min \left((q_i - \bar{\bar{q}}_i)/s_i', 1 + (q_i - \bar{q}_i)/s_i'' \right) \tag{14}$$

and the functions $z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$ for minimized objectives ($i = p' + 1, \ldots, p''$) are defined by:

$$z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) = \min \left((\bar{\bar{q}}_i - q_i)/s_i', 1 + (\bar{q}_i - q_i)/s_i'' \right) \tag{15}$$

while the functions $z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u)$ for stabilized objectives $(i = p''+1, \ldots, p)$ are defined by:

$$\begin{aligned} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) &= \min\left(z_i^l, z_i^u\right) \\ z_i^l &= \min\left((q_i - \bar{\bar{q}}_i^l)/s_i', 1 + (q_i - \bar{q}_i^l)/s_i''\right) \\ z_i^u &= \min\left((\bar{\bar{q}}_i^u - q_i)/s_i', 1 + (\bar{q}_i^u - q_i)/s_i''\right) \end{aligned} \tag{16}$$

where

$$\begin{aligned} \bar{q}_i^l &= q_i^s - \delta(q_i^s - \bar{\bar{q}}_i^l), \\ \bar{q}_i^u &= q_i^s + \delta(\bar{\bar{q}}_i^u - q_i^s), \\ q_i^s &= (\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2. \end{aligned} \tag{17}$$

The coefficients $s_i' > 0$, $s_i'' > 0$ in (14), (15) and (16) are scaling units for all objectives and are determined automatically in the IAC-DIDAS-N system to obtain the following common *absolute achievement measure* for all *individual criterion achievement functions* $z_i(q_i, \cdot, \cdot)$:

$$z_i = \begin{cases} 1+\eta & \text{if } q_i = q_i^{\text{best}} \quad (q_i^s \text{ for stabilized objectives}) \\ 1 & \text{if } q_i = \bar{q}_i \quad (\bar{q}_i^l \text{ or } \bar{q}_i^u \text{ for stabilized objectives}) \\ 0 & \text{if } q_i = \bar{\bar{q}}_i \quad (\bar{\bar{q}}_i^l \text{ or } \bar{\bar{q}}_i^u \text{ for stabilized objectives}) \end{cases} \tag{18}$$

where q_i^{best} is the upper limit (for maximized objectives; lower limit for minimized objectives) of all attainable efficient values of objective q_i and $\eta > 0$ is an arbitrary coefficient.

For minimized or maximized objectives $(i = 1, \ldots, p'')$, scaling coefficients s_i' and s_i'' depend on relations between aspiration level $\bar{q}_i$, reservation level $\bar{\bar{q}}_i$ and upper limit $q_i^{\max}$ (for maximized objectives; lower limit $q_i^{\min}$ for minimized objectives) of all attainable efficient values of objective q_i :

$$\begin{aligned} s_i' &= \bar{q}_i - \bar{\bar{q}}_i, \quad & s_i'' &= (q_i^{\max} - \bar{q}_i)/\eta, \quad & \text{if} \quad & 1 \le i \le p', \\ s_i' &= \bar{\bar{q}}_i - \bar{q}_i, \quad & s_i'' &= (\bar{q}_i - q_i^{\min})/\eta, \quad & \text{if} \quad & p'+1 \le i \le p''. \end{aligned} \tag{19}$$

For stabilized objectives $(i = p''+1, \ldots, p)$, scaling coefficients s_i' and s_i'' depend on the distance between $\bar{\bar{q}}_i^l$ and $\bar{\bar{q}}_i^u$ (i.e. reservation range) and on the user-defined coefficient δ (i.e. relations between aspiration and reservation ranges):

$$\left.\begin{aligned} s_i' &= (1-\delta)(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2 \\ s_i'' &= \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/(2\eta) \end{aligned}\right\} \quad \text{if } p''+1 \le i \le p. \tag{20}$$

Parameter η in (18), (19) and (20) is selected according to current relations between $\bar{q}_i$, $\bar{\bar{q}}_i$, $q_i^{\max}$, $q_i^{\min}$ and the value of coefficient δ :

$$\eta = \min\left(\min_{1\le i\le p'} \frac{q_i^{\max} - \bar{q}_i}{\bar{q}_i - \bar{\bar{q}}_i}, \min_{p'+1\le i\le p''} \frac{\bar{q}_i - q_i^{\min}}{\bar{\bar{q}}_i - \bar{q}_i}, \frac{\delta}{1-\delta}\right) \tag{21}$$

The system checks and does necessary projections for three sets of conditions that must hold for this selection of s_i' and s_i'' :

$$\begin{aligned} \bar{\bar{q}}_i < \bar{q}_i < q_i^{\max} \quad &, \text{if} \quad 1 \le i \le p', \\ q_i^{\min} < \bar{q}_i < \bar{\bar{q}}_i \quad &, \text{if} \quad p'+1 \le i \le p'', \\ \bar{\bar{q}}_i^l < \bar{\bar{q}}_i^u \quad &, \text{if} \quad p''+1 \le i \le p. \end{aligned} \tag{22}$$

The achievement function $s(q, \bar{q}, \bar{\bar{q}})$ can be maximized with $q = f(x)$ over $x \in X$; however, the function (13) is nondifferentiable (for example, if $q = \bar{q}$). On the other hand, if the function $g(x)$ (and thus also $f(x)$) is differentiable, then the maximization of function (13) in the system can be converted automatically to an equivalent differentiable nonlinear programming problem by introducing proxy variables and substituting the min operation in (13) by a number of additional inequalities. If the coefficient ε is positive ($\varepsilon > 0$), then the achievement function has the following properties (see Wierzbicki, 1986):

a) For any arbitrary aspiration and reservation points satisfying conditions (22), not necessarily restricted to be attainable ($\bar{q} \in Q$, $\bar{\bar{q}} \in Q$) or not attainable ($\bar{q} \notin Q$, $\bar{\bar{q}} \notin Q$), each maximal point $\hat{q}$ of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$ with $q = f(x)$ over $x \in X$ is a D_ε-efficient solution, that is, a properly efficient solution with trade-off coefficients bounded approximately by ε and $1/\varepsilon$.

b) For any properly efficient outcome $\hat{q}$ with trade-off coefficients bounded by ε and $1/\varepsilon$, there exist such aspiration $\bar{q}$ and reservation $\bar{\bar{q}}$ points that the maximum of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$ is attained at the properly efficient outcome $\hat{q}$. In particular, if the user (either by chance or as a result of a learning process) specifies some attainable but not efficient reservation point $\bar{\bar{q}}$ and an aspiration point $\bar{q}$ that in itself is such properly efficient outcome, $\bar{q} = \hat{q}$, and if conditions (22) are satisfied, then the maximum of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$, equal to one, is attained precisely at this point.

c) If the aspiration point $\bar{q}$ is 'too high' (for maximized outcomes; 'too low' for minimized outcomes), then the maximum of the achievement function, smaller than one, is attained at an efficient outcome $\hat{q}$ that best approximates uniformly, in the sense of scaling units s_i', the aspiration point. If the aspiration point $\bar{q}$ is 'too low' (for maximized outcomes; 'too high' for minimized outcomes), then the maximum of the achievement function, larger than one, is attained at an efficient outcome $\hat{q}$ that is uniformly, in the sense of scaling units s_i'', 'higher' than the aspiration point.

d) By changing his aspiration $\bar{q}$ and reservation $\bar{\bar{q}}$ points, the user can continuously influence the selection of the corresponding efficient outcomes $\hat{q}$ that maximize the achievement function, provided the maximum is unique and the set $\hat{Q}^{p\varepsilon}$ is connected.

The parameter ε in the achievement function determines bounds on trade-off coefficients: if an efficient solution has trade-off coefficients that are too large or too small (say, lower than 10^{-6} or higher than 10^{6}) then for the decision maker it does not differ from weakly efficient outcomes — some of its components can be improved without practically deteriorating other components. Another interpretation of this parameter is that it indicates how much an average overachievement (or underachievement) of aspiration levels should correct the minimal overachievement (or maximal underachievement) in the function (13).

The achievement function (13) can be transformed to an equivalent form when taking into account the scaling coefficients determined by (19) and (20) and assuming, for simplicity, that the parameter $\varepsilon = 0$:

$$s(q, \bar{q}, \bar{\bar{q}}) = 1 + \eta - \max \left(\max_{1 \le i \le p''} \tilde{z}_i(q_i, \bar{q}_i, \bar{\bar{q}}_i), \max_{p''+1 \le i \le p} \tilde{z}_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) \qquad (23)$$

with

$$\begin{aligned} \tilde{z}_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) &= \max\left(w_i', w_i''\right), \qquad 1 \le i \le p'', \\ \tilde{z}_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) &= \max\left(w_i^{-\prime}, w_i^{-\prime\prime}, w_i^{+\prime\prime}, w_i^{+\prime}\right), \qquad p''+1 \le i \le p, \end{aligned} \tag{24}$$

where

$$w_i' = \begin{cases} \eta + \dfrac{\bar{q}_i - q_i}{\bar{q}_i - \bar{\bar{q}}_i}, & \text{if} \quad 1 \le i \le p', \\ \eta + \dfrac{q_i - \bar{q}_i}{\bar{\bar{q}}_i - \bar{q}_i}, & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{25}$$

$$w_i'' = \begin{cases} \eta \dfrac{q_i^{\max} - q_i}{q_i^{\max} - \bar{q}_i}, & \text{if} \quad 1 \le i \le p', \\ \eta \dfrac{q_i - q_i^{\min}}{\bar{q}_i - q_i^{\min}}, & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{26}$$

$$\left.\begin{aligned} w_i^{-\prime} &= \eta + \frac{\bar{q}_i^l - q_i}{\bar{q}_i^l - \bar{\bar{q}}_i^l} \\ w_i^{-\prime\prime} &= 2\eta \frac{q_i^s - q_i}{\bar{q}_i^u - \bar{q}_i^l} \\ w_i^{+\prime\prime} &= 2\eta \frac{q_i - q_i^s}{\bar{q}_i^u - \bar{q}_i^l} \\ w_i^{+\prime} &= \eta + \frac{q_i - \bar{q}_i^u}{\bar{\bar{q}}_i^u - \bar{q}_i^u} \end{aligned}\right\} \quad \text{if} \quad p''+1 \le i \le p, \tag{27}$$

with q_i^s, $\bar{q}_i^l$ and $\bar{q}_i^u$ given by (17).

The maximization of an achievement function in IAC-DIDAS-N is performed by a specially developed nonlinear optimization algorithm, called *solver*. Since this maximization is performed repetitively, at least once for each interaction with the user that changes the parameters $\bar{q}$ or $\bar{\bar{q}}$, there are special requirements for the solver that distinguish this algorithm from typical nonlinear optimization algorithms: it should be robust, adaptable and efficient, that is, it should compute reasonably fast an optimal solution for optimization problems of a broad class (for various differentiable functions $g(x)$ and $f(x)$) without requiring of the user to adjust special parameters of the algorithm in order to obtain a solution. The experience in applying nonlinear optimization algorithms in decision support systems (see Kreglewski and Lewandowski, 1983, Kaden and Kreglewski, 1986) has led to the choice of an algorithm based on penalty shifting technique and projected conjugate gradient method. Since a penalty shifting technique anyway approximates nonlinear constraints by penalty terms, an appropriate form of an achievement function that differentiably approximates function (23) has been also developed and is actually used in IAC-DIDAS-N. This *smooth order-approximating achievement function* has the form:

$$\begin{aligned} s(q, \bar{q}, \bar{\bar{q}}) = 1 + \eta - \Bigg\{ &\sum_{i=1}^{p''} \left[\left(\max(0, w_i')\right)^\alpha + (w_i'')^\alpha \right] + \\ &+ \sum_{i=p''+1}^{p} \left[\left(\max(0, w_i^{-\prime})\right)^\alpha + \left(\max(w_i^{-\prime\prime}, w_i^{+\prime\prime})\right)^\alpha + \left(\max(0, w_i^{+\prime})\right)^\alpha \right] \Bigg\}^{1/\alpha} \end{aligned} \tag{28}$$

where w_i', w_i'', $w_i^{-\prime}$, $w_i^{-\prime\prime}$, $w_i^{+\prime\prime}$ and $w_i^{+\prime}$ are given by (25), (26) and (27).

The parameter $\alpha \geq 2$ is responsible for the approximation of the function (13) or (23) by the function (28): if $\alpha \to \infty$ and $\varepsilon \to 0$, then these functions converge to each other (if taking into account the specific definition of scaling coefficients in (13)). However, the use of too large parameter α results in badly conditioned problems when maximizing function (28), hence $\alpha = 4 \div 10$ are suggested to be used, the default value is $\alpha = 10$. During numerical computations a slightly simpler *scalarizing function* is used and minimized:

$$\tilde{s}(q, \bar{q}, \bar{\bar{q}}) = \sum_{i=1}^{p''} \left[(\max(0, w_i'))^\alpha + (w_i'')^\alpha \right] + \\ + \sum_{i=p''+1}^{p} \left[\left(\max(0, w_i^{-\prime}) \right)^\alpha + \left(\max(w_i^{-\prime\prime}, w_i^{+\prime\prime}) \right)^\alpha + \left(\max(0, w_i^{+\prime}) \right)^\alpha \right] \tag{29}$$

The function (29) must be minimized with $q = f(x)$ over $x \in X$, while X is determined by simple bounds $x^{\text{lo}} \leq x \leq x^{\text{up}}$ as well as by inequality constraints $y^{\text{lo}} \leq g(x) \leq y^{\text{up}}$ (or equality constraints for some i such that $y_i^{\text{lo}} = y_i^{\text{up}}$). In the shifted penalty technique, the following function is minimized instead:

$$p(x, \xi', \xi'', u', u'') = \tilde{s}(f(x), \bar{q}, \bar{\bar{q}}) + \\ + \frac{1}{2} \sum_{i=1}^{p} \xi_i' \left(\max(0, g_i(x) - y_i^{\text{up}} + u_i') \right)^2 + \\ + \frac{1}{2} \sum_{i=1}^{p} \xi_i'' \left(\max(0, y_i^{\text{lo}} - g_i(x) + u_i'') \right)^2 \tag{30}$$

where ξ', ξ'' are penalty coefficients and u', u'' are penalty shifts. This function is minimized with respect to x such that $x^{\text{lo}} \leq x \leq x^{\text{up}}$ while applying conjugate gradient directions, projected on these simple bounds if some of them become active. When a minimum of this penalty function with given penalty coefficients and given penalty shifts (the latter are initially equal to zero) is found, the violations of all outcome constraints are computed, the penalty shifts and coefficients are modified according to the shifted-increased penalty technique (see, e.g., Wierzbicki, 1984), and the penalty function is minimized again until the violations of outcome constraints are admissibly small. The results are then equivalent to the outcomes obtained by minimizing the scalarizing function (29) under all constraints. This technique, though it might seem cumbersome, is according to our experience one of the most robust nonlinear optimization methods; the user of the system is not bothered with its details, since the adjustment of penalty shifts and coefficients is done automatically.

Another advantage for the user is that he is bothered neither with the definition of derivatives of penalty function (30), needed in the conjugate gradient method, nor even with the definition of the derivatives of constraint functions $g_i(x)$ and outcome functions $f(x)$. This is the unique feature of IAC-DIDAS-N system: all needed derivatives are automatically (symbolically) determined and computed either in the nonlinear model generator that supports the model definition phase or in the solver algorithm that uses shifted penalty technique.

The only parameter that may influence the interaction of the system with the user is the parameter α in the smooth scalarizing function (29). Thus, the user can select this parameter; if this parameter is very large, his control of efficient outcomes obtained by

minimizing (29) is somewhat easier, but the solver may take long time or produce not quite robust results in this case. The user has also access to some other parameters of the optimization procedures; it is needed in cases of especially difficult optimization problems.

The minimization of a scalarizing function is a convenient way of organizing the interaction between the model and the user. Before the interactive analysis phase, however, the user must firstly define the substantive model, then define the multiobjective analysis problem by specifying outcome variables that should be maximized, minimized, stabilized, or *floating* (that is, displayed for user's information only, but not included as optimized objectives; such outcome should be defined as minimized or maximized but with neither aspiration level nor reservation level defined).

The scalarizing function of the form (29) uses two kinds of additional information:

- bounds for efficient outcomes: 'upper' bounds for maximized outcomes, 'lower' bounds for minimized outcomes. These bounds must be determined once for the given multiobjective analysis problem.

- user-supplied reference levels: aspiration level and reservation level for each minimized or maximized outcome, two reservation levels for each stabilized outcome. The user changes reference levels (aspiration, reservation or both) several times during the interactive analysis of the multiobjective problem, however some initial values should be determined in the system.

In the initial analysis phase of the work with the IAC-DIDAS-N system the bounds for efficient outcomes are calculated: the 'upper' (in the meaning of the 'best' attainable) and the 'lower' (in the meaning of the 'worst' attainable and efficient). The former is determined exactly (with given numerical accuracy), whereas the latter is only approximated, because there is no constructive way to determine it exactly for nonlinear multicriteria problems.

The 'upper' bound for efficient solutions is obtained through maximizing each objective separately (or minimizing, in case of minimized objectives; in the case of stabilized objectives, the user should know their entire attainable range, hence they should be both maximized and minimized), while all other objectives (including stabilized ones) should be considered as floating or free. The scalarizing function (29) is not used during these calculations, objective functions $q_i = f_i(x)$ are used in the penalty function instead of $\tilde{s}$ (with the plus sign if the objective under consideration should be minimized or with the minus sign if it should be maximized). If there are no stabilized outcomes, the results of such optimizations form a point that limits from 'above' (for maximized outcomes; from 'below' for minimized outcomes) the set of efficient outcomes $\widehat{Q}$, but this point almost never (except in degenerate cases) is in itself an attainable outcome; therefore, it is called the *utopia point*. The total number of optimization runs in utopia point computations is $p'' + 2(p - p'')$.

During all these computations, the 'lower' bound for efficient outcomes can be also estimated, just by recording the lowest (for maximized objectives; highest for minimized objectives) efficient outcomes that occur in subsequent optimizations (there is no need to record them for stabilized objectives, where the entire attainable range is anyway estimated). However, such a procedure results in the accurate, strict 'lower' bound for efficient outcomes — called *nadir point* $\hat{q}^{\mathrm{nad}}$ — only if $p'' = 2$; for larger number of maximized and minimized objectives, particularly for nonlinear models, this procedure can give misleading results. In further computations appropriate components of $\hat{q}^{\mathrm{uto}}$ and $\hat{q}^{\mathrm{nad}}$ are used as components of q^{max} and q^{min} in the scalarizing function (29).

In very rare and rather degenerate cases some components $\hat{q}_i^{\text{nad}}$ of the nadir point estimation and corresponding components $\hat{q}_i^{\text{uto}}$ of the utopia point can have the same value — it may happen if, for example, the structure of the substantive model results in the set (2) with empty interior. In such case the user can update manually these nadir point components according to his knowledge, otherwise the IAC-DIDAS-N system assumes such outcomes to be floating (they are not included in the scalarizing function (29) regardless of its type — maximized, minimized or stabilized) but checks their values at each efficient solution whether they are still equal to the values $\hat{q}_i^{\text{nad}} = \hat{q}_i^{\text{uto}}$.

The approximate bounds $\hat{q}^{\text{uto}}$ and $\hat{q}^{\text{nad}}$ once computed and presented to the user can be utilized in various ways. First, their appropriate components are used as components of q^{max} and q^{min} in the scalarizing function (29). Second way consists in computing a *neutral efficient solution*, with objectives situated approximately 'in the middle' of the efficient set. For this purpose, the aspiration point $\bar{q}$ is set very close to the utopia point $\hat{q}^{\text{uto}}$ (only for maximized or minimized outcomes; for stabilized outcomes upper and lower limits of efficient outcomes are used as appropriate reservation levels $\bar{\bar{q}}_i^l = \hat{q}_i^{\text{min}}$ and $\bar{\bar{q}}_i^u = \hat{q}_i^{\text{max}}$) and the reservation point $\bar{\bar{q}}$ is set very close to the nadir point $\hat{q}^{\text{nad}}$ (only for maximized and minimized objectives). By minimizing the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$ with such data, the neutral efficient solution is obtained and can be utilized by the user as a starting point for further interactive analysis of efficient solutions. Basing on the neutral efficient solution $\hat{q}^{\text{neu}}$ and bounds on efficient objectives $\hat{q}^{\text{uto}}$ and $\hat{q}^{\text{nad}}$, system proposes to the user the following initial values for aspiration levels $\bar{q}_i$ and reservation levels $\bar{\bar{q}}_i$ for maximized and minimized objectives:

$$\left.\begin{aligned} \bar{q}_i &= \hat{q}_i^{\text{neu}} - (\hat{q}_i^{\text{neu}} - \hat{q}_i^{\text{uto}})/3 \\ \bar{\bar{q}}_i &= \hat{q}_i^{\text{neu}} + (\hat{q}_i^{\text{neu}} - \hat{q}_i^{\text{uto}})/3 \end{aligned}\right\} \quad \text{if} \quad 1 \leq i \leq p'' \tag{31}$$

and the following initial values for lower $\bar{\bar{q}}_i^l$ and upper $\bar{\bar{q}}_i^u$ reservation levels for stabilized objectives:

$$\left.\begin{aligned} \bar{\bar{q}}_i^l &= \hat{q}_i^{\text{neu}} - \Delta_i \\ \bar{\bar{q}}_i^u &= \hat{q}_i^{\text{neu}} + \Delta_i \end{aligned}\right\} \quad \text{if} \quad p'' + 1 \leq i \leq p \tag{32}$$

where

$$\Delta_i = 0.5 \min \left(\hat{q}_i^{\text{neu}} - q_i^{\text{min}}, q_i^{\text{max}} - \hat{q}_i^{\text{neu}}\right)$$

These values, although rather arbitrary, constitute a good starting point for further interaction.

In further interactive analysis, an important consideration is that the user should be able to easily influence the selection of the efficient outcomes $\hat{q}$ by changing the aspiration point $\bar{q}$ (and, optionally, the reservation point $\bar{\bar{q}}$) for maximized and minimized objectives and reservation levels $\bar{\bar{q}}^l$ and $\bar{\bar{q}}^u$ for stabilized objectives in the minimized scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$. It can be shown (see Wierzbicki, 1986) that the best suited choice for this purpose is the choice of scaling units s_i' and s_i'' that are not constant, but are changed implicitly by the user and depend on differences between current values of aspiration and reservation levels and utopia point components either according to (19) and (20) or, equivalently, as a result of using the scalarizing function (29) with (25), (26) and (27) provided that conditions (22) hold. The interpretation of such way of setting scaling units is that the user attaches implicitly more importance to reaching an aspiration component $\bar{q}_i$ if he places it close to the known utopia component; in such case, the corresponding scaling unit becomes smaller and the corresponding objective component weighs stronger in

the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$. Thus, this way of *scaling relative to utopia-reference difference* takes into account the implicit information given by the user in the relative position of the aspiration point. The only drawback of the described choice of scaling units are strong inequalities in conditions (22), not convenient for the user and for the numerical application. Therefore, $q_i^{\max}$ and $q_i^{\min}$ in (25) and (26) are not taken directly as appropriate components of $\hat{q}^{\text{uto}}$ and $\hat{q}^{\text{nad}}$, but slightly displaced utopia and nadir points are used instead in the current system implementation:

$$\begin{aligned} q_i^{\max} &= \hat{q}_i^{\text{uto}} + 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}), \quad \text{if} \quad 1 \le i \le p', \\ q_i^{\min} &= \hat{q}_i^{\text{uto}} - 0.01(\hat{q}_i^{\text{nad}} - \hat{q}_i^{\text{uto}}), \quad \text{if} \quad p'+1 \le i \le p''. \end{aligned} \tag{33}$$

It is assumed now that the user selects the aspiration and reservation components satisfying $\hat{q}_i^{\text{nad}} \le \bar{\bar{q}}_i < \bar{q}_i \le \hat{q}_i^{\text{uto}}$ for maximized outcomes and $\hat{q}_i^{\text{uto}} \le \bar{q}_i < \bar{\bar{q}}_i \le \hat{q}_i^{\text{nad}}$ for minimized outcomes and $\bar{\bar{q}}_i^l < \bar{\bar{q}}_i^u$ for stabilized outcomes (if he does not, the system automatically does necessary projections). If the user specifies only one reference value for some objective, then the system determines the second value internally, thus the same two reference level scalarizing function can be used. For maximized and minimized objectives missing reservation levels are calculated using formulae:

$$\bar{\bar{q}}_i = \begin{cases} \bar{q}_i - (q_i^{\max} - \bar{q}_i), & \text{if} \quad 1 \le i \le p', \\ \bar{q}_i + (\bar{q}_i - q_i^{\min}), & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{34}$$

whereas missing aspiration levels are calculated using formulae:

$$\bar{q}_i = \begin{cases} 0.5(q_i^{\max} + \bar{\bar{q}}_i), & \text{if} \quad 1 \le i \le p', \\ 0.5(\bar{\bar{q}}_i + q_i^{\min}), & \text{if} \quad p'+1 \le i \le p''. \end{cases} \tag{35}$$

When the relative scaling is applied, the user can easily obtain — by suitably moving reference points — efficient outcomes that are situated either close to the neutral solution, in the middle of efficient outcome set $\hat{Q}$, or in some remote parts of the set $\hat{Q}$, say, close to various extreme solutions. Typically, several experiments of computing such efficient outcomes give enough information to the user to select an actual decision — either some efficient decision suggested by the system, or rather a different one, since even the best substantive model cannot encompass all aspects of a decision situation. However, there may be some cases in which the user would like to receive further support — either in analyzing the sensitivity of a selected efficient outcome or in converging to some best preferred solution.

For analyzing the sensitivity of an efficient solution to changes in the proportions of outcomes, a *multidimensional scan* of efficient outcomes can be applied in IAC-DIDAS-N. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for aspiration points, and performing p (or p'') additional optimization runs with the aspiration points determined by:

$$\bar{q}_j = \bar{q}_j^{\text{bas}} + \beta\left(\hat{q}_j^{\text{uto}} - \hat{q}_j^{\text{nad}}\right), \quad \bar{q}_i = \bar{q}_i^{\text{bas}}, \quad i \ne j, \quad 1 \le i \le p \tag{36}$$

where β is a coefficient determined by the user, $-1 \le \beta \le 1$; if the aspiration components determined by (36) are outside the range $\hat{q}_j^{\text{nad}}$, $\hat{q}_j^{\text{uto}}$, they are projected automatically

on this range; the reservation point is kept constant ($\bar{\bar{q}} = \hat{q}^{\text{nad}}$) during this procedure. The aspiration components for stabilized outcomes may or may not be perturbed in this operation. The efficient outcomes, resulting from the minimization of the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$ with such perturbed aspiration points, are typically also perturbed mostly along their respective components, although other their components may also change.

For analyzing the sensitivity of an efficient solution when moving along a direction in the outcome space — and also as a help in converging to a most preferred solution — a *directional scan* of efficient outcomes can be implemented in IAC-DIDAS-N. This operation consists again in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for aspiration points, selecting another aspiration point $\bar{q}$, and performing a user-specified number K of additional optimizations with aspiration points determined by:

$$\bar{q}(k) = \bar{q}^{\text{bas}} + \frac{k}{K}\left(\bar{q} - \bar{q}^{\text{bas}}\right), \quad 1 \leq k \leq K \tag{37}$$

The efficient solutions $\hat{q}(k)$ obtained through minimizing the scalarizing function $\tilde{s}(q, \bar{q}(k), \bar{\bar{q}})$ with such aspiration points (and constant reservation point $\bar{\bar{q}} = \hat{q}^{\text{nad}}$) constitute a cut through the efficient set $\tilde{Q}$ when moving approximately in the direction $\bar{q} - \bar{q}^{\text{bas}}$. If the user selects one of these efficient solutions, accepts as a new $\bar{q}^{\text{bas}}$ and performs next directional scans along some new directions of improvement, he can converge eventually to his most preferred solution (see Korhonen, 1985). Even if he does not wish the help in such convergence, directional scans can give him valuable information.

Another possible way of helping with convergence to the most preferred solution is choosing aspiration points as in (37) but using a harmonically decreasing sequence of coefficients (such as $1/j$, where j is the iteration number) instead of user-selected coefficients k/K. It results in convergence even if the user makes stochastic errors in determining next directions of improvement of aspiration points, and even if he is not sure about his preferences and learns about them during this analysis (see Michalevich, 1986). Such a convergence, called here forced convergence, is rather slow. Neither the forced convergence nor multidimensional scan nor directional scan are implemented in the current version of IAC-DIDAS-N, though they could be included in later versions.

3 Short user manual

3.1 Introduction

The IAC-DIDAS-N system (Institute of Automatic Control, Dynamic Interactive Decision Analysis and Support, Nonlinear version) is decision support system designed to help in the analysis of decision situations where a mathematical model of substantive aspects of the situation can be formulated in the form of a multiobjective nonlinear programming problem.

The system can be run on an `IBM--PC--XT`, `AT` or a compatible computer with Hercules Graphics Card, Color Graphic Adapter or Enhanced Graphics Adapter and, preferably, with a numeric coprocessor and a hard disk. If a numeric coprocessor is available, then the system takes advantage of the coprocessor computational capacity, otherwise the system uses built-in software emulator of the numeric coprocessor with less computational power. The system is recorded on one diskette. The diskette contains the compiled code of the program together with some data files with demonstrative examples of nonlinear models. When the installation of of the system in the user directory on a hard disk (or less

preferably on a working diskette) is done (using `INSTALL` batch file contained on the diskette — see the installation guide in the Appendix A), the program can be activated by the command `DIDASN` at the `DOS` prompt.

System supports the following general functions:

- definition and edition of a substantive model of the decision situation in a user-friendly way using a spreadsheet and a screen window editor.
- simulation of the model. All numerical errors can be fixed directly. This is performed by model debugger with visualisation of outcome formulae and their derivatives, automatic error tracking together with forward and backward step by step calculations option.
- specification of a multiobjective decision analysis problem related to the substantive model. This is performed by specific features of spreadsheet edition.
- initial multiobjective analysis of the problem, resulting in estimating bounds on efficient outcomes of decisions and in learning about some extreme and some neutral decisions. These functions are supported by some specific commands and the results are presented to the user in the spreadsheet and graphical form.
- interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized as system's response to user-specified aspiration and reservation levels for objective outcomes. The IAC-DIDAS-N system responds with efficient solutions and objective outcomes obtained by the maximization of an achievement function that is parameterized by the user-specified aspiration and reservation points. A nonlinear programming algorithm called solver performs the maximization. The interactive analysis is supported by entering user data into specific cells in the spreadsheet, executing commands from the menu and using graphical representation of results.

The menus of IAC-DIDAS-N are organized as pull-down tree-structured menus and they perform various functions used in several phases of the interactive analysis process. Most of the functions of the model edition phase as well as of the phase of the decision problem specification and the problem initial analysis are specific commands in the spreadsheet edition (the decision variables are defined as columns of the spreadsheet, the outcome variables are defined as rows, outcome formulae are entered in the corresponding cells; there are special rows and columns for lower and upper bounds, for defining user names of objective outcomes and their types, reference points, utopia point, for solutions corresponding to the reference points). The functions of the interactive analysis phase are executed by macrocommands involved by menus and various function keys; the user can get various help displays that suggest in an easy fashion the commands useful in a current phase of the work with the system.

IAC-DIDAS-N system has been developed in the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland which has the authorship rights, under the contracted study agreement "Theory, Software and Testing Examples for Decision Support Systems" with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis, Laxenburg near Vienna, Austria, which has the copyright for this system in international distribution. Please contact Methodology of Decision Analysis Project of SDS Program at IIASA, A–2361 Laxenburg, Austria.

3.2 Phases of the work

The work with a IAC-DIDAS-N system consists of three phases:

1. model edition phase
2. problem definition and initial analysis phase
3. interactive analysis and comparison of results phase

All these phases are supported in the system and an explicit command is required to move from one phase to another. Moreover, system checks whether requested move is possible and gives appropriate error messages or asks for additional confirmation. There are two logical spreadsheets: a model editing spreadsheet and an interactive analysis spreadsheet. The former is used mostly to perform all the system functions in the first phase of the work, whereas the latter performs all the system functions during two other phases.

System invoked without arguments always starts with phase 1 and permits of the move to phase 2 only if the model definition is complete. A complete model consists of three groups of obligatory data:

- valid formulae for all defined outcomes (rows of the spreadsheet),
- lower and upper bounds (that do not contradict each other) for all variables,
- values for all used parameters.

Optionally, model can contain some other, user-supplied data:

- lower and upper bounds (that do not contradict each other) for outcomes,
- names of variables, parameters and outcomes (user names override standard system names of the form `x1`, `x2`, ...for variables, `z1`, `z2`, ...for parameters and `y1`, `y2`, ...for outcomes). The names must be unique within the model.
- units for variables, parameters and outcomes. This information can be included to improve the understanding of the model in the spreadsheet, but it is not used by the system. The only exception is the use of outcome units for special scaling method in graphical representation.
- lower and upper bounds for parameters; not used in the current system implementation but planned to be used in parametric analysis in future system implementations.
- short (up to 30 characters) model description; it is displayed in the spreadsheet and printed together with the print-out of the model. It may be used as an extension to the model name, that is too short (up to 8 characters) to be enough meaningful.
- five parameters that are used to tune the nonlinear solver (see the next section). If some of them are not given, then current default system values are stored together with other model data.

To store the edited model on a disk, the name of the model must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt M F > (Model selection — Fix and save) commands (see section 3.6), system asks for the name (up to 8 characters; it must be a valid DOS file name). However, there is an important distinction between these

two commands. The former just saves the model as it is, whereas the latter first checks the model and either displays an error message if the model is not complete or changes the status of the model to 'fixed' and stores the complete model. Both commands check, whether the given name is not the same as the model file name already existing in the current model directory on a disk. If it is the same, then the system asks for additional confirmation. Answering 'yes' means, that the system deletes the file with the same name containing the previous model definition together with all related problem definitions and result data.

Successful 'Fix and save' command for the model moves the system automatically to the second phase of the work. Obligatory elements of the model definition and bounds for outcomes (if given) cannot be changed now. The command < Alt M N > (Model selection — New) must be used to move back to the first phase to allow any change in this part of the model definition, but it will be treated as a definition of a new model. The command < Alt M R > (Model selection — Reset) can also be used to move back to the first phase, but it deletes all model data and starts a new model definition from the scratch. Optional parts of the model definition (except the bounds for outcomes) can be changed even if the model is fixed, because they do not affect computational characteristics of the model. Such changes in the model definition can be stored on a disk using the command < F2 > (Save). In particular changes in the rows' and columns' order done with the < Alt F R M > (Format — Rows — Move) and < Alt F C M > (Format — Columns — Move) commands are admissible for a fixed model and are stored with the < F2 > (Save) command.

Model stored on a disk can be restored using the command < Alt M G > (Model selection — Get from disk). The model is restored together with the information, whether it was fixed or not. Thus, after restoring not fixed model the system is still in the phase 1, whereas after restoring a fixed model the system moves automatically to the phase 2. It is also possible to restore a model immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line (for example, to restore automatically the model `DEMO`, invoke the system with the command `DIDASN DEMO`).

Edited model (not necessarily fixed) can be printed using the command < Alt M P > (Model selection — Print). Model stored on a disk, but actually not interesting to the user can be deleted from a disk together with all related problem definitions and result data using the command < Alt M E > (Model selection — Erase).

Second phase of the work with the IAC-DIDAS-N system lies in a specification of a decision problem to be analyzed, for already defined and 'fixed' model. The complete definition of a decision problem consists of two parts: user-supplied part and system-supplied part. The user-supplied part must contain two kinds of information: a selection of outcomes to be minimized objectives, maximized objectives or stabilized ones (defined by the contents of the 'Stat' column in the spreadsheet) and values of lower and upper bounds on outcomes, if they are not given in the model definition. Moreover, the user can add some other data:

- lower and upper bounds for outcomes, even if already given in the model definition; the new values override correspondent data in the model definition,
- short (up to 30 characters) problem description; it is displayed in the spreadsheet and printed together with the print-out of the problem. It may be used as an extension to the problem name, that is too short (up to 8 characters) to be enough meaningful.

- five parameters that are used to tune the nonlinear solver (see the next section); these values override correspondent data either in the model definition or default system values.

- new bounds (updates) of the system-supplied approximation of the nadir point.

For a given set of selected objectives and bounds for objectives (and optional bounds redefinitions and some solver parameters) system performs initial analysis of the decision problem: calculation of ranges of efficient solutions (utopia and nadir points) and calculation of neutral solution being the starting point for further interaction during the third phase of the work (see the theoretical manual). Two parts of this analysis can be performed jointly using the command < F3 > (Calculate) or separately using first the command < Alt P U > (Problem selection — Utopia) and next the command < Alt P T > (Problem selection — neuTral). The system-supplied part of a complete problem consists of results of these calculations. If the calculations are performed separately, then after calculating the utopia point (but prior to the neutral solution calculation) the user can modify values of the nadir point approximation using the command < Alt P A > (Problem selection — nAdir).

To store the edited problem on a disk, the name of the problem must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt P F > (Problem selection — Fix and save) commands, system asks for the name (up to 8 characters). However, there is again an important distinction between these two commands. The former just saves the problem as it is, whereas the latter first checks the problem and either displays an error message if the problem is not complete or changes the status of the problem to 'fixed' and stores the complete problem. The file containing the model definition is automatically updated if any changes are made in its optional part. Both commands check, whether the given name is not the same as the problem name already existing for the currently used model. If it is, then system asks for additional confirmation. Answer 'yes' means, that the system deletes previous problem definition together with all related result data.

Successful 'Fix and save' command for the problem moves the system automatically to the third phase of the work. None of the problem elements can be changed now. The command < Alt P N > (Problem selection — New) must be used to move back to the second phase to allow any change in the problem definition, but it will be treated as a definition of a new problem. The command < Alt P R > (Problem selection — Reset) can also be used to move back to the second phase, but it deletes all problem data and starts new problem definition from the scratch.

Problem stored on a disk can be restored using the command < Alt P G > (Problem selection — Get from disk). The problem is restored together with the information, whether it was fixed or not. Thus, after restoring not fixed problem the system is still in the phase 2, whereas after restoring a fixed problem the system moves automatically to the phase 3. It is also possible to restore a problem immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line and the name of the problem must be given as the second argument in the DOS command line (for example, to restore automatically the problem `FIRST` defined for the model `DEMO`, invoke the system with the command `DIDASN DEMO FIRST`).

The result of neutral solution calculation is stored as a result with a system-defined name 'Neutral'. It is stored and restored each time when a fixed problem related to it is stored and restored.

Edited problem (not necessarily fixed) can be printed using the command < Alt P P > (Problem selection — Print). Moreover, if the problem is fixed, then the neutral result can also be printed using the command < Alt R P > (Result selection — Print). Problem stored on a disk, but actually not interesting to the user can be deleted together with all related result data using the command < Alt P E > (Problem selection — Erase).

Third phase of the work with the IAC-DIDAS-N system lies in an interactive analysis of the decision problem already defined and 'fixed', for defined and 'fixed' model. The only values, that are changed by the user during this phase, are aspirations and reservations for minimized or maximized objectives as well as lower and upper reservations for stabilized objectives (see the theoretical manual). The complete result consists of these user-supplied data together with the system's response — an efficient solution calculated using scalarizing function parameterized with these data. Two commands, either < F3 > (Calculate) or < Alt R C > (Result selection — Calculate), can be used to start the calculations.

Moreover, the user can add short (up to 30 characters) result description; it is displayed in the spreadsheet and printed together with the print-out of the result. It may be used as an extension to the result name, that is too short (up to 8 characters) to be enough meaningful.

To store the calculated result on a disk, the name of the result must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt R S > (Result selection — Save and new) commands, system asks for the name (up to 8 characters). There is no difference between these two commands. Both commands first check the result and either display a message if the result is not calculated or store the calculated result. The model definition is automatically updated if any changes are made in its optional part. Both commands check, whether the given name is not the same as the result name already existing for the currently used problem and model. If it is, then system asks for additional confirmation. Answer 'yes' means, that the system deletes the previous result data. Moreover, the name 'Neutral' is reserved and cannot be used.

Two kinds of data are stored and restored as results; these are efficient values of objectives together with values of other, non-objective outcomes and values of variables related to the efficient solution. Values of variables are loaded into the spreadsheet (row Values in the model editing spreadsheet) following each successful calculation of an efficient solution and while restoring a result from a disk. The obtained or restored values are used as a starting point for the calculation of next efficient solution.

If the user finds, that calculated result is not interesting, it is possible to clear it either using the command < Alt R N > (Result selection — New) that clears only the system response or using the command < Alt R R > (Result selection — Reset) that clears also user-supplied part of the problem definition. However, only results stored on a disk can be compared using graphical representation.

Result stored on a disk can be restored using the command < Alt R G > (Result selection — Get from disk). It is also possible to restore a result immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line, the name of the problem must be given as the second argument in the DOS command line and the name of the result must be given as the third argument in the DOS command line (for example, to restore automatically the result `R1` obtained for the problem `FIRST` defined for the model `DEMO`, invoke the system with the command `DIDASN DEMO FIRST R1`).

Calculated result can be printed using the command < Alt R P > (Result selection — Print). Result stored on a disk, but actually not interesting to the user can be deleted

from a disk using the command < Alt R E > (Result selection — Erase).

3.3 Editing with the spreadsheet

Two spreadsheets used in the IAC-DIDAS-N system are rather specialized. They differ from the standard ones (like Lotus 1-2-3) in two aspects. First, in the implemented spreadsheets there are predefined types of contents of all cells: there are dedicated cells for storing text, other cells for storing numbers and other ones for storing formulae. Secondly, the IAC-DIDAS-N has an integrated compiler with a symbolic differentiation facility, that compiles the formulae and produces binary codes for calculations of formula value and for calculations of all derivatives. Therefore, two kinds of operation are defined for spreadsheet cells with formulae: compilation and calculation. Version 4.0 of the IAC-DIDAS-N uses new compiler that generates directly binary code for numeric coprocessor; calculations of formula and derivative values are more than ten times faster then in previous versions of the system with interpreted internal code.

The top screen line in both spreadsheets contains pull-down menu entries and the bottom line contains the function keys meanings.

Both spreadsheets are built from two partially independent parts. First three columns from the left are common for both spreadsheets and have as many rows as outcomes in the model. If the model has more than 13 outcomes, then only 13 are displayed and the spreadsheet is scrolled up and down according to moves of the spreadsheet marker. These three columns are used to enter and display outcomes' definition: name, unit and status, from left to right, respectively. However, the status is the element of a problem definition. Therefore, it can be accessed only from the interactive analysis spreadsheet. The fourth column of the model editing spreadsheet contains outcome formulae, but only their values are displayed; to display and edit the formula one cell from this column must be selected with the spreadsheet marker and then the < Enter > key causes the display of formula in a window.

In the upper left corner of both spreadsheets the status of the model, problem and result are displayed together with the amount of free memory in the relation to the amount of free memory available after the system initialization. For example, FreeMem 78 % means that only 22% of the available memory is currently used for model, problem and result definitions. If the displayed value is below 20% then it is not recommended to use the system , because of large amount of memory required temporarily during compilation of formula, optimization and graphical representation. The behavior of the system is not completely predicable in some out of memory situations. Another status value displayed in the upper left corner of the screen is the flag Auto `ON/OFF` toggled with the function key < F8 > or with the command < Alt S A > (Switches — Auto `ON/OFF`). This flag reflects the state of the automatic spreadsheet recalculation feature. If it is `ON`, then the spreadsheet is recalculated after each change of any cell, otherwise, the recalculation is only on explicit request command < F3 > (Calculate).

The second part of the model editing spreadsheet has as many columns as variables, parameters and outcomes; at the top of each column a type designator is displayed: 'var', 'par' or 'out', respectively. Each column contains: name, unit, upper bound, value, lower bound and values of derivatives, from top to bottom, respectively. First five items are elements of model definition, except values of variables that are accessible and can be changed in any phase of the work. Cells with values of outcomes are only for display and their contents can not be edited. Only up to four columns are displayed at a time. If there are more than four columns, the spreadsheet is scrolled left and right according to

moves of the spreadsheet marker. Just above the area with derivative values the texts 'Partial derivative values' or 'Total derivative values' are alternatively displayed. They are toggled with the function key < F7 > or with the command < Alt S T > (Switches — Totals **ON/OFF**) and reflect the type of derivatives that are calculated and displayed. Values of partial or total derivatives are only displayed in the spreadsheet. To display a formula of a derivative an appropriate cell must be selected with the spreadsheet marker and then the < Enter > key causes the display of formula in a window. Formulae for partial derivatives can only be displyed.

The model debugger can be used if either a formula or a derivative of a formula is displayed in the editor window. According to the information displayed on the bottom of the window, < Alt D > keys activates step by step execution of currently displayed formula. < PgUp > key causes next step to be performed; partial results are displayed in additional window and the part of formula that is calculated in current step is highlighted. < PgDn > key causes the same steps to be performed backward. < Alt D > keys cause restart of the debugger from the beginning of the currently displayed formula. < Enter > key or < Esc > key terminate the debugger session.

The second part of the interactive analysis spreadsheet has seven columns, but only five of them are displayed at a time. Thus, it is scrolled one column left or one column right according to moves of the spreadsheet marker. Contents of these seven columns depend on the type of outcome and can be different for different rows of the spreadsheet. Descriptions of the columns change according to the type of outcome which the spreadsheet marker is currently pointing to. For outcomes not selected as objectives the spreadsheet columns contain (from left to right): upper bound, blank (column with empty cells), blank, last calculated or loaded value of solution, blank, blank, lower bound. For maximized objectives there are there: upper bound, utopia value, aspiration level, last solution value, reservation level, nadir value and lower bound. For minimized objectives there are there: lower bound, utopia value, aspiration level, last solution value, reservation level, nadir value and upper bound. At last, for stabilized objectives there are there: upper bound, upper utopia value, upper reservation level, last solution value, lower reservation level, lower utopia value and lower bound. At a first look, this arrangement seems to be very complicated, but one can easily find that all values, that should be mutually compared, are placed side by side and that in most cases all values at each row either decrease or increase while moving from left to right. If the problem is not fixed, then the columns with aspirations and reservations (for minimized and maximized objectives; with lower and upper reservations for stabilized objectives) are not displayed, thus the remaining five columns are displayed all the time and there is no need to scroll the spreadsheet horizontally. At the top of this part of the interactive analysis spreadsheet there are displayed names and descriptions of current model, problem and result.

To move the spreadsheet marker eight cursor keys can be used to perform four regular and four oblique directions of move (for example, < Home > cursor key moves the marker in the upper-left direction). Moreover, it is also possible to perform fast jumps from one part of the spreadsheet to another (only in the model editing spreadsheet). Fast jumps from the right part to the left part (namely to the formulae column) and back, without change of the row and without scrolling the right part, are performed with the key combinations < Ctrl Left > and < Ctrl Right >, whereas fast jumps from the bottom part to the top part (namely to the value row) and back, without change of the column and without scrolling the bottom part, are performed with the key combinations < Ctrl PgUp > and < Ctrl PgDn >.

The way data are entered into a particular cell depends on a type of data that the cell is

destined to store. To edit the contents of a particular cell one must move the spreadsheet marker to this cell. The edition of contents of a spreadsheet cell is performed by use of two editors: formula editor for edition of formula (only the formulae column in the model edition spreadsheet) and the cell editor for all other cells. In the latter case there are two possibilities: the modification of the previous contents or the input of new contents. The < Enter > key causes the entry into the cell editor using the previous contents of the cell. Any other key is treated as the first input character of the new contents; the previous contents are then deleted. The cell editor allows for the use of standard editing keys (< Backspace >, < Del >, < Left >, < Right >, < Home >, < End >) in two modes: insert mode and replace mode that are toggled with the < Ins > key. Current mode is indicated with the shape of the cursor — block cursor means insert mode, underline cursor means replace mode. There are two keys that ends the cell edition: < Enter > key means storing a new data, < Esc > key means restoring the previous contents.

To start the edition of a formula only the < Enter > key can be pressed, thus it is always the edition of the previous contents of the cell. The formula editor is a full featured window editor. Standard editing keys (< Backspace >, < Del >, < Left >, < Right >, < Up >, < Down >, < Home >, < End >, < PgUp >, < PgDn >) can be used in two modes exactly like in the cell editor. The functions of two terminating keys are also the same. Moreover, to facilitate the edition of several formulae that are very similar or even with some identical parts, the concept of a buffer has been implemented.

Any part of currently displayed formula can be marked (displayed in a reverse video mode): its beginning is marked with < F7 > function key and the end is marked with < F8 > function key. Marked area can be copied (duplicated) into the current cursor position by use of < F10 > function key. Marked area can be deleted by use of < F6 > function key. However, to avoid unintentional deletions the cursor must be at the position just following the end of the marked block, otherwise, an appropriate message is displayed and the block is not deleted. Marked area can be copied (duplicated) into the buffer using the function key < F5 > and then restored in the current cursor position using the function key < F10 >. The contents of the buffer are displayed in a window at the bottom of the screen. This window is closed (without deleting the contents of the buffer) either by use of the function key < F9 > or when the formula editor is left, and opened again by use of the function key < F9 >. If the buffer contents is too long to fit the size of the window, then < Up > and < Down > cursor keys are used to scroll it up and down, one line at a time.

3.4 Usage of the nonlinear solver

The nonlinear solver used in the IAC-DIDAS-N system is rather fast and robust. Its operation, however, depends on some parameters that should be sometimes adjusted to the properties of a particular model and problem. For small models (consisting of not more than ten variables and outcomes) the default system values of the parameters can be successfully used. There are four ways of changing values of these parameters: permanent, for a model, for a problem and for a result. Parameters are changed using menu entry Options in the either of spreadsheets. If any parameter is changed, then, while leaving the menu, system asks 'Do you want to make these changes permanent (Y/N) ?'. If the answer is yes (< Y > or < y >), then the program updates itself on a disk in such a way, that the new values become default values in all subsequent runs of the system, otherwise, the changes are only temporary, valid till the end of a current run of the system. Values of the parameters are stored together with the model definition and with the problem

definition, these values are restored while the model or problem are loaded.

The first three parameters are used in stop tests of the solver:

Accuracy — the norm of gradient of the penalty function at the solution point must be not greater than the Accuracy value. The default value is 10^{-4}, but for large, highly nonlinear models this value should be changed to 10^{-2} or even more.

Violation — the nonlinear constraints (bounds on outcomes) at the solution point must not be violated more than Violation value. The default value is 10^{-4}, but this value should be also increased for large models with many nonlinear constraints.

Iterations — the limit of iterations (recalculations of the spreadsheet). The default value is 1000.

The last two parameters define the shape of the scalarizing function:

Scaling exp. — It is the parameter α in the scalarizing function (see theoretical manual). The default value is 10; it should be decreased to 4 or 6 for large models (it must be an even number).

Ratio Asp/Res — It is the parameter δ in the scalarizing function (see theoretical manual). The default value is 0.1 and may be changed within a range 0.01–0.9. The selection of this value is rather a matter of taste and does not depend on the size of the problem.

Although there are two independent stop tests (Accuracy check together with Violation check and Iterations limit check) the user can interrupt the running optimization process by pressing < Ctrl Break > key combination. The best point obtained till this moment is then displayed as a solution, but the system does not treat this point as a solution.

Optimization runs are sometimes very time consuming, therefore, to give the user an information that the system has not crashed, a sequence of letters is displayed. Each time, when the solver stops the move in a current direction, a next letter is displayed. The letters are displayed in the right half of the second to the last line on the screen starting with the letter 'A'. If all 40 positions are filled with 'A', they are cleared and letters 'B' ('C', 'D' etc.) are displayed.

There are five possible messages displayed at the end of optimization run. These are:

Optimal solution found — it is the most desirable message.

Required accuracy not attainable — it means that the solution has been found with the full numerical accuracy, but due to round-off errors the required accuracy has not been obtained. The Accuracy and Violation parameters should be increased.

Iterations limit — solution not found — the Iterations parameter should be increased.

Interrupted with < Ctrl-Break > — the user pressed the < Ctrl Break > key combination.

Solver error NNN — this message should never appear[1].

[1] Please let us know if it does (NNN is the error number).

3.5 Graphical representation of results

Several objectives and several results can be displayed simultaneously in a graphical bar form. The system has some internal rules of selecting results and objectives to be displayed. These rules can be summarized as follows: display up to 10 objectives and as many results as possible. The < F9 > (Graphics) command is used to display bar representation using these rules. However, the command < Alt G O > (Graphics — Objectives selection) can be used to manually select objectives to be displayed — up to 10 objectives can be marked and next displayed. The command < Alt G R > (Graphics — Results selection) can be used to manually select results to be displayed — up to 10 results can be marked and next displayed. The user selection overrides the system rules, thus, subsequent executions of < F9 > (Graphics) command cause the display of user selected objectives and results.

Two scaling methods are implemented in the graphical representation. In the first scaling method (Normal scale `ON`) each objective is scaled independently: the maximal level (top of the bar picture) is equal to the upper bound on displayed objective and the minimal level (bottom of the bar picture) is equal to the lower bound on displayed objective. This scaling method is very simple, but has two important drawbacks. First, very often the bounds' range is much wider than the range of efficient solutions, therefore, all solutions together with utopia and nadir points can be represented as one point of he bar. Secondly, the objectives that form trajectories (in dynamic cases) are scaled independently and cannot be compared.

In the second scaling method (Normal scale `OFF`) the range of efficient changes is determined for each objective independently:

- for minimized objectives the maximal level is equal to the value of the reservation level or to the value of the solution, whichever is greater; the minimal level is equal to the value of the utopia level.
- for maximized objectives the maximal level is equal to the value of the utopia level; the minimal level is equal to the value of the reservation level or to the value of the solution, whichever is less.
- for stabilized objectives the maximal level is equal to the value of upper reservation level or to the value of the solution, whichever is greater; the minimal value is equal to the value of lower reservation level or to the value of the solution, whichever is less.

Next, for all groups of objectives with the same contents of the Units column in the spreadsheet, the common scale is determined as the distance from the lowest minimal level to the highest maximal level of all objectives within the group.

3.6 Menu and function keys description

Most commands are executed by entering into a pull-down menu, moreover some most frequently used commands are also accessible by function keys. There are two sets of pull-down menu entries and two sets of function key commands for two existing spreadsheets — model edition spreadsheet and multiobjective analysis spreadsheet; in particular, function keys are used to select one of the spreadsheets.

Pull-down menu commands can be invoked in two ways — using < Alt > key or using < Esc > key. Acting in the former way the user must press the key related to the first

(highlighted) letter of the pull-down menu entry (eg. < S > key for Switches menu entry) while holding down the < Alt > key — such action will be denoted as < Alt X > where X means the letter (eg. < Alt S >). Acting in the latter way the user must press < Esc > key and the last used pull-down menu is entered — thus this way is useful for repeated commands.

In both ways, < Right > and < Left > cursor keys can be used to move from one menu entry to another. Commands within current menu entry can be selected either by moving the menu marker with < Up > and < Down > cursor keys and pressing the < Enter > key or by pressing the key related to the first (highlighted) upper case letter of the command name (in most cases it is the first letter of the command). Selection means the execution of the command or entry to the next menu level. Pressing the < Esc > key, while in a menu, causes exit from the current level of the menu either to the previous menu level or to the spreadsheet. If the menu window is too small to display all items then up and down arrows appear in the bottom line of the menu window and the menu is scrolled vertically according to the moves of the marking bar.

In the following description menu entries are terminated by a colon whereas commands are terminated by a dot. In the former case related menu items are listed below the description of the menu entry.

3.6.1 Menu for model edition

< Alt M > **Model selection:** — commands for operations on models as whole entities

- **Get from disk:** — displays menu of currently defined models; selected model is loaded from a disk into the spreadsheet
- **Fix and save.** — checks the completeness of the model, fixes it and saves on a disk (asks for model name if not named previously)
- **Description.** — asks for brief description of the model (up to 30 characters long)
- **New.** — un-fixes the model, resets the name and the description; the model in the spreadsheet remains unchanged
- **Reset.** — un-fixes the model, resets the name and the description, deletes all data in the spreadsheet, resizes the spreadsheet to one-row and one-column
- **Erase:** — displays menu of currently defined models; selected model is deleted together with all related problems and results
- **Print.** — prints current model

< Alt F > **Format:** — operations on rows or columns of the spreadsheet

- **Rows:** — changes the number or the order of rows in the spreadsheet

Insert. — inserts blank, highlighted row in the spreadsheet; this row can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys; < Alt I > makes insertion permanent, < Esc > key cancels it

Delete. — highlights a row; highlighting horizontal bar can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys, < Alt D > deletes highlighted row (if not referenced in other rows) and related column, < Esc > key cancels the command

Move. — highlights a row; the highlighting horizontal bar can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys, < Alt S > selects highlighted row. The selected row can then be moved within a spreadsheet; < Alt P > places the highlighted row in a current place, < Esc > key cancels the command

Columns: — changes the number, order or type of columns in the spreadsheet

Insert. — inserts blank, highlighted column in the spreadsheet; this column can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt V > fixes the column as a variable, < Alt P > fixes the column as a parameter (system asks for its initial value), < Esc > key cancels it

Delete. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt D > deletes selected column (if not referenced in the spreadsheet; columns related to outcomes cannot be deleted), < Esc > key cancels the command

Move. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt S > selects highlighted column. The selected column can then be moved within a spreadsheet; < Alt P > places the highlighted column in a current place, < Esc > key cancels the command

to **Var.** — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to a variable type (if it was an outcome then related row is deleted), < Esc > key cancels the command

to **Par.** — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to a parameter type (if it was an outcome then related row is deleted), < Esc > key cancels the command

to **Out.** — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to an outcome type (related row is created), < Esc > key cancels the command

< Alt S > **S**witches: — influence numerical calculations in the spreadsheet

Totals on/off. — toggles calculation and display of values of either partial or total derivatives

Auto on/off. — switches on and off the automatic recalculation of the spreadsheet after each change of its contents

< Alt C > **C**alculate. — recalculates the whole spreadsheet

< Alt L > **L**ist: — displays the menu of models stored on a disk, for the selected model displays the menu of problems stored on a disk, for the selected problem displays

the list of results stored on a disk, < Enter > key selects, < Esc > key moves back

< Alt O > **Options:** — changes colors and some problem-dependent parameters in the nonlinear programming solver

Colors: — enables changes of foreground and background colors or attributes of all items displayed on the screen during the work of the program in an easy, interactive way; all changes are immediately visible on the screen (see Appendix B for details)

Accuracy. — changes accuracy of optimization; if this value is too high, then results may be misleading, if this value is too low, then optimization time may be too large

Violation. — changes acceptable violation of bounds on outcomes; if this value is too high, then results may be misleading, if this value is too low, then optimization time may be too large

Iterations. — changes limit of iteration number (recalculations of the model) during each optimization run

Scaling exponent. — changes coefficient α in the scalarizing function

Ratio Asp./Res. — changes ratio of the width of aspiration and reservation ranges for stabilized objectives

3.6.2 Function keys for model edition

< F1 > — context sensitive help.

< F2 > — save. — saves model (if changed), problem (if changed) and result (if changed); asks for names (if not named previously)

< F3 > — calculate. — the same as < Alt C >

< F4 > — list. — the same as < Alt L >

< F6 > — go to the interactive analysis spreadsheet.

< F7 > — totals on/off. — the same as < Alt S T >

< F8 > — auto on/off. — the same as < Alt S A >

< F10 > — exit to DOS.

3.6.3 Menu for interactive analysis

< Alt M > **M**odel selection: — the same as in the model edition spreadsheet

< Alt P > **P**roblem selection: — commands for operation on problems as whole entities

Get from disk: — displays menu of currently defined problems (for current model), selected problem is loaded from a disk into the spreadsheet

Fix and save. — checks the completeness of the problem, fixes it and saves on a disk (asks for problem name if not named previously)

Description. — asks for brief description of the problem (up to 30 characters long)

New. — un-fixes the problem, resets the name and the description; the problem in the spreadsheet remains unchanged

Reset. — un-fixes the problem, resets the name and the description, deletes all problem data in the spreadsheet

Utopia. — checks the completeness of the problem and calculates the utopia point, approximates the nadir point (see theoretical manual)

n**A**dir. — enters special spreadsheet editing mode that enables user updates of the nadir point values, < Esc > key exits this mode

neu**T**ral. — checks the completeness of the problem and calculates the neutral solution (see theoretical manual)

Erase: — displays menu of currently defined problems (for current model); selected problem is deleted together with all related results

Print. — prints current problem

< Alt R > **R**esult selection: — commands for operations on results as whole entities

Get from disk: — displays menu of currently defined results (for current model and for current problem); selected result is loaded from a disk into the spreadsheet

Save and new. — checks the completeness of the result, saves it on a disk (asks for result name if not named previously) and resets the name and comment

Description. — asks for brief description of the result (up to 30 characters long)

New. — resets the name and the description; the result data in the spreadsheet remains unchanged

Reset. — resets the name and the description, clears all result data in the spreadsheet

Calculate. — checks the completeness of the result data and calculates the efficient solution (see theoretical manual)

Variables. — displays a window with values of variables related to current efficient solution

Erase: — displays menu of currently defined results (for current model and for current problem); selected result is deleted

Print. — prints current result

< Alt G > **Graphics:** — selects the results and objectives to be displayed, switches the scaling method and starts the display

Display. — displays the graphical representation of the results

Result selection: — displays menu of currently defined results (for current model and for current problem); selected results are displayed on the screen, < Enter > key selects (up to 10 results can be selected), < Esc > key moves back to the Graphics menu

Objectives selection: — displays menu of objectives; selected objectives are displayed on the screen, < Enter > key selects, < Esc > key moves back to the Graphics menu

Normal scale — toggles the scaling methods

< Alt O > **Options:** — the same as in the model edition spreadsheet

3.6.4 Function keys for interactive analysis

< F1 > — context sensitive help.

< F2 > — save. — the same as in the model edition spreadsheet

< F3 > — calculate. — calculates utopia point (if not calculated), neutral solution (if not calculated) and efficient solution (if not calculated), performs all necessary checks the completeness of model, problem and result data

< F4 > — list. — the same as < F4 > or < Alt L > in the model edition spreadsheet

< F5 > — go to the model edition spreadsheet.

< F9 > — graphics. — displays the graphical representation of the results using either default selection rules or last user-defined selection

< F10 > — exit to DOS.

3.7 Syntax of formulae

Outcome formulae entered into the spreadsheet are standard arithmetic expressions with some possible extensions. Five binary arithmetical operators can be used: addition '+', subtraction '–', multiplication '*', division '/' and power '^' , moreover an unary minus can be used, with higher precedence than binary operators. Standard arithmetical rules are used for operator precedence and calculation order, parenthesis can be inserted to imply specific order of calculations. There is only one restriction for the use of these operators: a sequence of two power operators $x \hat{} y \hat{} z$ is not allowed — either operator together with its arguments must by enclosed in parenthesis to explicitly define the order of calculations, i.e. it must be written as $(x \hat{} y) \hat{} z$ or $x \hat{} (y \hat{} z)$.

There are several built-in functions that can be used in outcome formulae, ten functions with one argument *abs*, *arctan*, *cos*, *exp*, *ln*, *log*, *signum*, *sin*, *sqr*, *sqrt* and two functions with two arguments *min*, *max*. Moreover, there is a predefined constant *Pi*. Functions *abs*, *signum*, *min*, *max* should be used with caution because they are nondifferentiable. Logical structures of the form *if logical expression then arithmetic expression else arithmetic expression* or *if logical expression then arithmetic expression elsif logical expression then arithmetic expression else arithmetic expression* should be used also with caution for the same reason. Up to ten levels of *elsif* are allowed. Relations of the form a `<` b (where a and b are any arithmetical expressions; possible relation operators are: '<' '<=' '=' '<>' '>=' '>') and of the form a `in [` b,c `]` (where a, b and c are any arithmetical expressions; '[' and ']' mean closed interval, '(' and ')' can also be used and mean open interval; possible forms are: 'in [.,.]' 'in [.,.)' 'in (.,.]' 'in (.,.)') and logical operators 'and' 'or' 'xor' 'not' are allowed in logical expressions.

4 Illustrative examples

4.1 Testing Example

This example has been chosen because its multiobjective analysis is simple and can be performed analytically. It serves to test the correctness of the installation of the program and to check whether the hardware and software `IBM--PC` compatibility of your computer is sufficient to use the `DIDASN` program.

The model has four variables (`xa xb xc xd`), two parameters (`za zb`) and three outcomes (`obj1 obj2 wrk`).

The model is defined as follows:

Outcome equations:

```
obj1  = (xa - 1) ^ 2 + za * (xb - 1) ^ 2 + (xc - 1) ^ 2 +
        (xd - 1) ^ 2 + wrk
obj2  = xa ^ 2 + za * xb ^ 2 + xc ^ 2 + xd ^ 2 + wrk
wrk   = zb * (xa - xb) ^ 2 + (zb - za) * (xc - xd) ^ 2
```

Bounds on variables and outcomes:

$$\begin{aligned} -10 \leq\ & \texttt{xa} & \leq 10 \\ -10 \leq\ & \texttt{xb} & \leq 10 \\ -10 \leq\ & \texttt{xc} & \leq 10 \\ -10 \leq\ & \texttt{xd} & \leq 10 \\ -1 \leq\ & \texttt{obj1} & \leq 12 \\ -1 \leq\ & \texttt{obj2} & \leq 12 \\ 0 \leq\ & \texttt{wrk} & \leq 100 \end{aligned}$$

Values of parameters:

$$\texttt{za} = 7$$
$$\texttt{zb} = 100$$

Initial values of variables:

$$\texttt{xa} = 1$$
$$\texttt{xb} = 2$$
$$\texttt{xc} = 3$$
$$\texttt{xd} = 4$$

The multiobjective nonlinear programming problem is to minimize objectives `obj1` and `obj2`, while the outcome `wrk` is floating (free).

The Pareto frontier in the objective space for this example can be determined analytically and has the form:

$$\sqrt{\texttt{obj1}/10} + \sqrt{\texttt{obj2}/10} = 1$$

with the utopia point `( 0.0, 0.0 )`, nadir point `( 10.0, 10.0 )`, and the neutral solution point `( 2.5, 2.5 )`. Numerical results obtained during computation will be slightly different because of numerical errors and finite accuracy of calculations.

To go through the testing example, we will perform the following actions:

1. We activate the program `DIDASN` at the DOS prompt.
2. We get initial banner with the system name, a version number and an information about the authors (Fig. 1).
3. We press any key and get initial, smallest possible model editing spreadsheet (Fig. 2).
4. We load the model by pressing following keys:

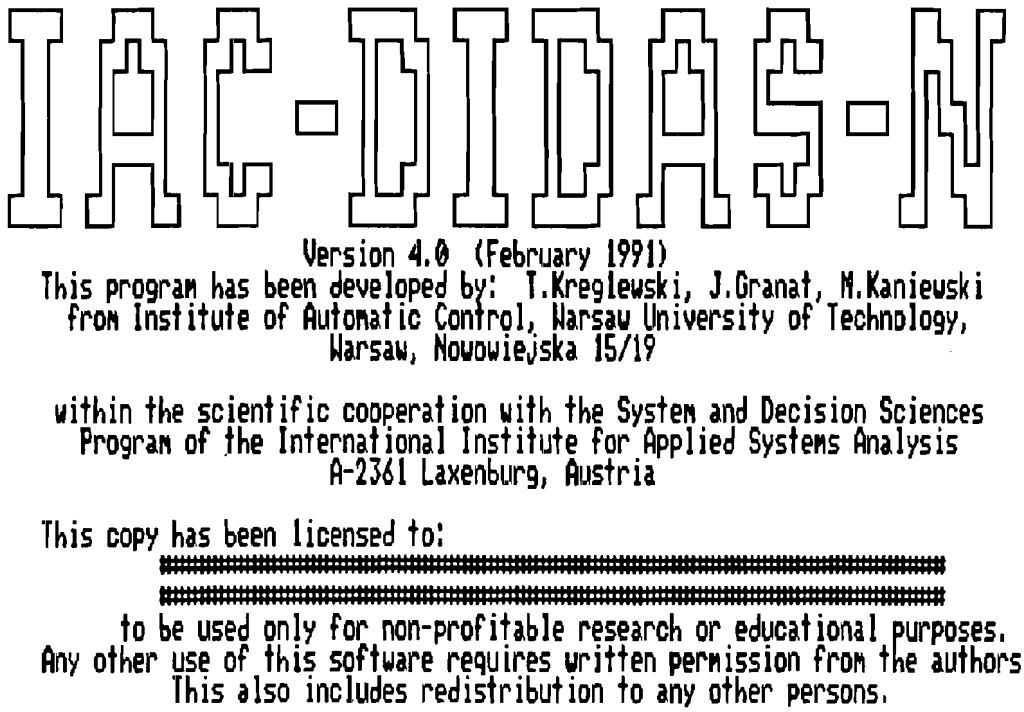

Figure 1: Initial screen

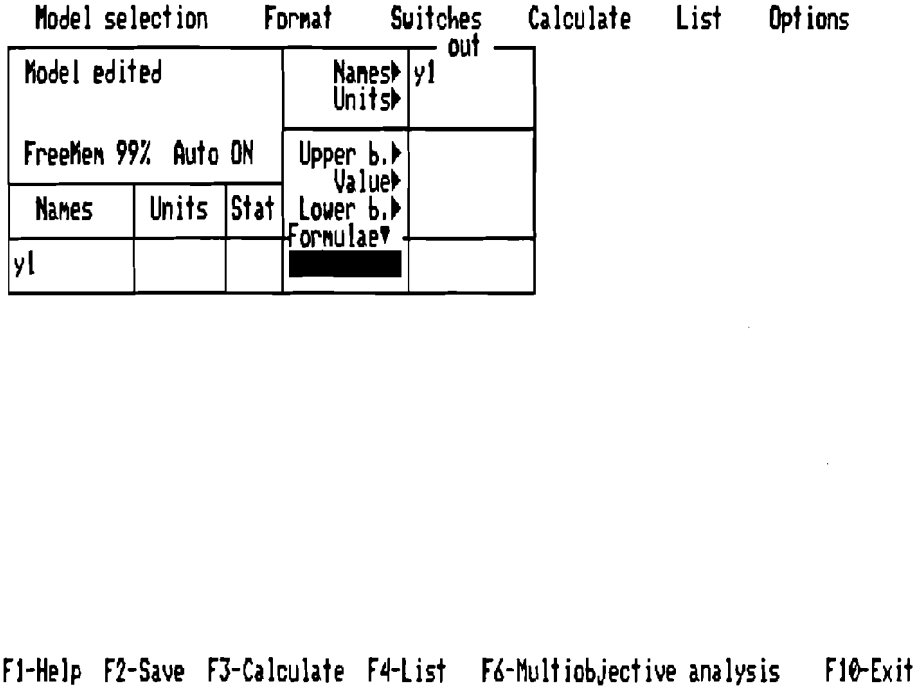

Figure 2: Initial spreadsheet

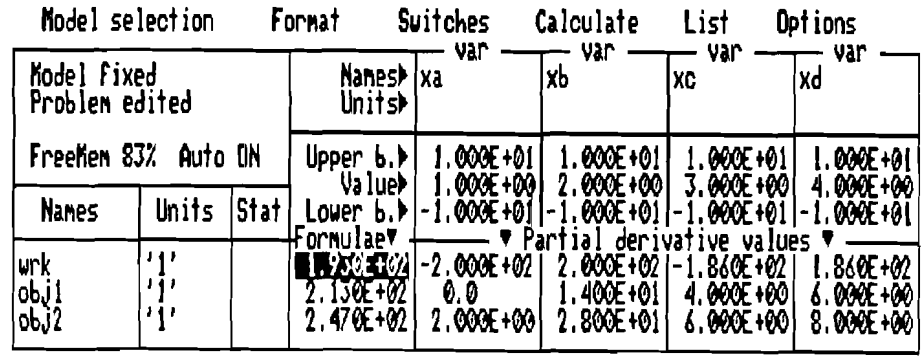

Figure 3: DEMO model loaded

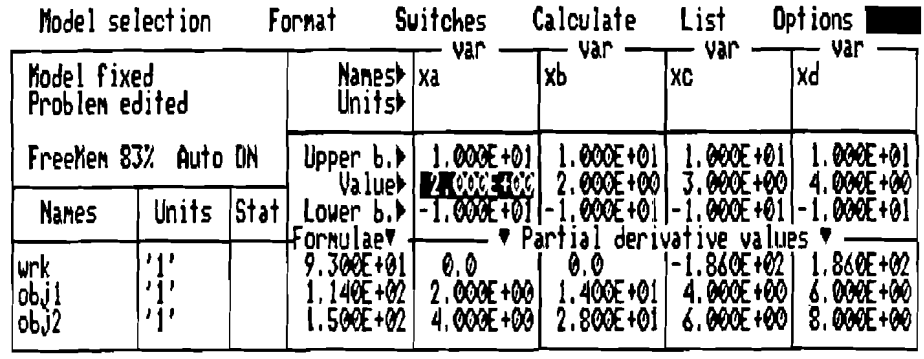

Figure 4: DEMO model recalculated

Model selection Problem selection Result selection Graphics Options

Model fixed Problem fixed Result calculated FreeMem 83% Auto DN			Model: DEMO ▶ IAC-DIDAS-N testing example ◀ Problem: demo1 ▶ IAC-DIDAS-N example problem ◀ Result: Neutral ▶ ◀				
Names	Units	Stat	Utopia	Asp. level	Neutral Solution	Res. level	Nadir
wrk	'1'				1.454E-14		
obj1	'1'	min	3.163E-26	1.667E+00	2.500E+00	3.333E+00	1.000E+01
obj2	'1'	min	2.085E-28	1.667E+00	2.500E+00	3.333E+00	1.000E+01

F1-Help F2-Save F3-Calculate F4-List F5-Model editing F9-Graphics F10-Exit

Figure 5: DEMO1 problem loaded

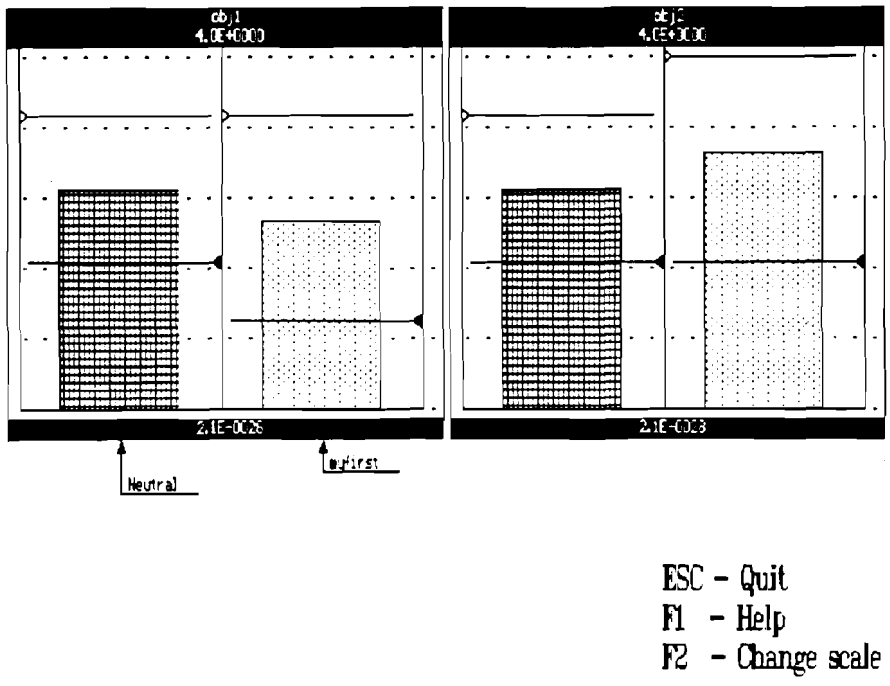

Figure 6: Graphical comparison

< Alt M >	[model selection menu appears in the upper left corner of the screen]
< G >	[list of accessible models appears in the small window with the DEMO model name being the first; if it is not the first, then we select the DEMO name by moving the marking bar with < Up > and < Down > cursor keys]
< Enter >	[DEMO model is loaded and displayed]
< F5 >	[DEMO model is stored on a disk with the status 'fixed', thus the interactive analysis spreadsheet is automatically selected; we switch to the model edition spreadsheet (Fig. 3)]

5. Using the cursor movement keys we move to the row Values and to the column **xa**. The marked cell contains the current value of the variable **xa** — this value is 1.0. We enter a new value, just typing < 2 > and pressing the < Enter > key. The spreadsheet is immediately recalculated (Fig. 4).

6. Now we switch to the interactive analysis spreadsheet by pressing the < F6 > key and we load example problem definition DEMO1 for the model DEMO by pressing < Alt P > < G > < Enter > keys. The problem definition together with utopia and nadir points, neutral solution and some proposed aspiration and reservation levels are then displayed (Fig. 5).

7. Now we will try to recalculate the problem and obtain the same results. We begin with the definition of a new problem by pressing < Alt P N > keys. Next we press < Alt P U > keys and the system determines the utopia point (analytical solution is `( obj1, obj2 ) = ( 0.0, 0.0 )`, obtained values are `( 3.163E-26, 2.085E-28 )` and the nadir point (analytical solution is `( obj1, obj2 ) = ( 10.0, 10.0 )`, obtained results are the same).

8. Let the system determine the neutral solution — we press < Alt P T > (It is also possible to press single function key < F3 > instead of < Alt P N > and < Alt P T > to calculate in sequence utopia point, nadir point and neutral solution). Analytical solution is `x = ( 0.5, 0.5, 0.5, 0.5 )`, `( wrk, obj1, obj2 ) = ( 0.0, 2.5, 2.5 )`, obtained results for `obj1` and `obj2` are exactly the same. We check results for variables by using the command < Alt R V > — they are exactly equal to the analytical ones. Now we fix the problem (by pressing < Alt P F > keys and by giving it any valid name — e.g. DEMO2). Before doing it we can submit some problem description (by < Alt P D >).

9. Let the system determine an efficient solution corresponding to aspiration level of objective `obj1` changed from `1.6667` to `1.0` and reservation level of objective `obj2` changed from `3.3333` to `4.0`; we move the spreadsheet marker to the respective cells and enter new values. The < F3 > function key initiates calculations. After a while we check obtained results; they are `2.124` and `2.906` for objectives `obj1` and `obj2`, respectively, and `3.014E-21` for outcome wrk. We check results for variables — they are `0.539114, 0.539114, 0.539114, 0.539114`. We save obtained result (by < Alt R S > and by giving it any valid name — e.g. MYFIRST). Before doing it we can submit some result description (by < Alt R D >).

10. Now we can compare two already obtained results by using graphical representation — for this purpose it is enough to press < F9 > function key. We obtain a screen with bars representing our results, it is better to change the scaling method by pressing the key < F2 > (Fig. 6).

4.2 Tutorial example

Typical procedure of working with the **DIDASN** program and various aspects of the use of several program commands are discussed in this section. This discussion is done by exploiting a real-life example specially designed for this purpose. The model used in this example is a very rough approximation of the much more complicated model of acid deposition in forest soil, described by Hettelingh and Hordijk (1987).

4.2.1 Description of the model

We consider two regions (denoted by the index $k = 1, 2$) burning one type of fuel (say, coal) and emitting sulphur dioxide. The problem is, in fact, a dynamic one and should be considered in many one year time periods; here we simplify it by considering only three periods (denoted by $t = 1, 2, 3$), each of them five years long.

The sulphur dioxide emission in each region and time period is determined by:

$$S_{k,t} = S^p_{k,t}(1 - p_{k,t}) \tag{38}$$

where $S^p_{k,t}$ is the potential emission, specified exogenously. It may be described in the form:

$$S^p_{k,t} = E_{k,t}\frac{z_{k,t}}{h_{k,t}}(1 - r_{k,t}) \tag{39}$$

where $E_{k,t}$ is the total energy production in region k and in time period t, $h_{k,t}$ is the heat content of the fuel, $z_{k,t}$ is the sulphur content of the fuel, $r_{k,t}$ is the reduction coefficient resulting from sulphur remaining in ashes. However, for the purpose of this simplified model, $S^p_{k,t}$ are assumed to be given as model parameters (for $k = 1, 2$ and $t = 1, 2, 3$). In the computerized model $S^p_{k,t}$ are denoted as parameter names **Sktp** and $S_{k,t}$ are denoted as outcome names **Skt** where k are digits $1, 2$ representing two regions and t are digits $1, 2, 3$ representing three time periods.

The reduction coefficients $p_{k,t}$ in (38) describe the effects of the pollution control measures. These coefficients serve as the main decision variables, therefore, there are actually six decision variables $p_{k,t}$ $(k = 1, 2,\ t = 1, 2, 3)$. In the computerized model they are denoted as variable names **pkt**.

It is assumed that the decision maker in k-th region is interested in:

- the costs of pollution control measures $C_{k,t}$ for each period;
- the level of pH (denoted here as $pH_{k,t}$ and denoted as outcome names **pHkt** in the model) in forest soil for each period;

or in two objective outcome trajectories each of three period length. In the **DIDAS** methodology, however, we investigate cooperative actions of both decision makers, therefore, the joined "decision maker" is interested in four outcome trajectories — two cost trajectories and two pH trajectories each of three period length, representing together twelve objective outcomes.

The cost $C_{k,t}$ (denoted in the model as outcome names `Ckt`) is function of the potential emission $S_{k,t}$ and of the reduction coefficient $p_{k,t}$. Actually, the situation is more complicated, since the costs have also dynamic character: there is a high investment cost of pollution control devices, but these devices are not so expensive in maintenance; on the other hand, once installed, the devices give defined coefficient $p_{k,t}$. However, when considering only five year periods we can apply much simpler model of the pollution control costs, understood as joined cost of investments and maintenance for five year period and dependent on the average reduction coefficient achieved during this period:

$$C_{k,t} = c_k S^p_{k,t} \frac{p_{k,t}}{2(1 - p_{k,t})} \tag{40}$$

where c_k is the cost of reducing the emission by half per one unit of potential emission. This is a very simple approximation of actual cost curves and it can be replaced by any other more exact approximation. The form of this approximation express, however, the fact that it becomes increasingly costly to obtain reduction coefficients close to 1 . Because of numerical reasons, the reduction coefficient mustn't be close to 1, thus it should be constrained in a range, say, $0 \leq p_{k,t} \leq 0.99$ (in the computerized model reduction coefficients are measured in percents and bounded from 0% to 99%).

The level of pH in forest soil is assumed to have more long-time dynamic aspects and thus it is modeled by dynamic equations. When approximating more complicated relations described in (Hettelingh and Hordijk, 1987), we must take into account that acid absorption and reduction capacities of forest soil are nonlinear, that they are the strongest in the carbonate range (pH = 8.0–6.2, but we will take pH = 7 as an upper bound), quite different and not so strong in the silicate range (pH = 6.2–5.0) and again the stronger in the cation exchange range (pH = 5.0–4.2) while any pH level below 4.0 might be considered as catastrophic. Therefore, instead of including more realistic and complicated models that may be considered in further variants of this application, in the tutorial example we consider only a nonlinear dynamic model for the pH range 7–4 of the approximate form:

$$pH_{k,t} = \left(pH_{k,t-1} + \alpha_k \left(7 - pH_{k,t-1}\right)\right) \left(1 - \frac{3}{7} \Phi \left(\frac{D_{k,t}}{CAP_k}\right)\right) \tag{41}$$

where $D_{k,t}$ (denoted in the model as outcome names `DktR`) are sulphur deposits in given region and period, CAP_k are the five year carrying absorption capacities (if $D_{k,t} \geq CAP_k$ then it is assumed that $pH_{k,t}$ drops to 4 or below), and the function Φ expresses the essential nonlinearity of absorption and reduction of acid by forest soil. A convenient form of this function is:

$$\Phi(x) = \begin{cases} 0, & \text{if} \quad x \leq 0 \\ 3x^2 - 2x^3, & \text{if} \quad 0 \leq x \leq 1 \\ 1, & \text{if} \quad x \geq 1 \end{cases} \tag{42}$$

This is a twice-differentiable (except at $x = 0$ and $x = 1$, where it is only once-differentiable) spline function.

If $x = D_{k,t}/CAP_k = 0$, then the dynamic part of (41) illustrates the self-regeneration of forest soil with a regeneration coefficient α_k (during a five year period); this coefficient characterizes how many times the distance between pH=7 and the actual pII level will decrease in five years.

The initial value $pH_{k,0}$ (for $t = 0$) is given as a parameter for both regions $k = 1, 2$. It should be stressed again that the nonlinear dynamic model (41) is only a very rough approximation of actual forest soil chemistry and must be updated by specialists for more realistic policy analysis performed for other than only tutorial example purposes.

The sulphur deposits $D_{k,t}$ are the results of sulphur-emissions $S_{k,t}$ as determined by a deposition model, which, in this simplified case, is again assumed in the simplest possible form:

$$D_{k,t} = a_{k,1}S_{1,t} + a_{k,2}S_{2,t}, \qquad k = 1, 2 \tag{43}$$

where $a_{k,j}$ $(0 \leq a_{k,j} \leq 1)$ are transfer coefficients from region j to region k (for simplicity, we assume $a_{k,1} + a_{k,2} = 1, \ k = 1, 2$).

With such simplified model we can illustrate the issues of multiobjective dynamic and nonlinear analysis of the effects of pollution control. The analysts or the decision makers can jointly analyze in this model:

- what can be the maximal pollution reduction rates, if there are limited funds for pollution control in each of time periods, and what are the corresponding effects on forest soil acidity;
- what are the possibilities of multiobjective dynamic compromises between the trajectories of costs in all periods and trajectories of forest soil acidity.

For both purposes, the `DIDAS` methodology can be applied. The multiobjective analysis can be performed by specifying reference (aspiration or aspiration and reservation) trajectories for the costs and for the pH levels, while the `DIDASN` system can compute multiobjectively optimal (effective) trajectories for these variables that are consistent with the model (feasible) and, in a sense best, attuned to the reference trajectories.

4.2.2 Sample session

The model described in the previous section is already prepared as a disk file `RAIN` and can be loaded into the IAC-DIDAS-N spreadsheet by using the command < Alt M G > (Model selection — Get from disk). It is stored as a 'fixed' model, thus to make some experiments with the model we must use the command < Alt M N > (Model selection — New). Now we can change all upper and lower bounds, values of parameters and outcome formula. Following each change of variable or parameter value the spreadsheet is automatically recalculated. After some play with the model we load again the original one and start the second phase of the work.

We define the decision problem now. We select outcomes to be minimized or maximized. First we take into account only one region performances: we mark costs `C11 C12 C13` as minimized and pH levels `pH11 pH12 pH13` as maximized. Bounds on all outcomes are defined in the model and we don't redefine them. We ask the system to calculate Utopia and Nadir points by using the command < Alt P U > (Problem selection — Utopia) and after a while we get the results. Next we ask the system to calculate a compromise solution, so called neutral solution, being the starting point for further interaction. We enter the command < Alt P T > (Problem selection — neuTral) and again wait a while. When the neutral solution is calculated and displayed, the problem definition is finished and we can 'fix' the problem by using the command < Alt P F > (Problem selection — Fix and save). We are asked to give a name of the problem, it may be `ONEREG`. Basing on values of Utopia, Nadir and Neutral points the system proposes us initial values of aspiration and reservation levels. Further interaction consists in a sequence of three or four actions:

- modification of aspiration and/or reservation levels (it is enough to change only one value); it is obtained through the edition of appropriate spreadsheet cells.

- calculation of the efficient solution corresponding to current levels of aspirations and reservations. Optimization process is initiated by the use of the < F3 > (Calculate) command.

- if the result (efficient solution) is not satisfactory, we can discard it using the command < Alt R N > (Result selection — New) and go back to the first action. Otherwise, we save the result by using the command < Alt R S > (Result selection — Save and new).

- optionally, we can compare several results, obtained for current problem, using graphical representation; directly by the command < F9 > or with some selections of objectives and results to be displayed within < Alt G > (Graphics) menu.

Because all interesting results are stored on a disk, interaction session can be stopped at any time and next resumed.

Now we continue the interaction, but for the previously defined problem. We load the problem `RAIN1` by using the command < Alt P G > (Problem selection — Get from disk). The problem definition with calculated utopia and nadir values together with the neutral solution are loaded. There are twelve objectives now: costs `C11 C12 C13` and pH levels `pH11 pH12 pH13` for the first region and costs `C21 C22 C23` and pH levels `pH21 pH22 pH23` for the second region. Please observe, that neutral solution values for the first region are worse than in the previous problem. It is due to the fact that now the neutral solution is a compromise between interests of both regions.

We find, that costs in the second region are decisively too large, but pH in both regions can be accepted. Thus, we try to decrease costs in the second region by decreasing reservation levels for costs `C21 C22 C23` from 2180 to 1500. We press < F3 > then we wait for the result and save it using the command < F2 >. To compare the result with the neutral solution we use graphical representation. First, we select objectives to be displayed — only ten of them can be displayed simultaneously. We enter the command < Alt G O > (Graphics — Objectives selection). The system displays the list of all twelve objectives with first ten being marked. Changes of pH in the last period are more important for us are than those in the first period. Therefore, we 'unmark' objectives `pH11 pH21` and 'mark' objectives `pH22 pH23`. Both operations are performed by moving the marking bar with the < Up > and < Down > cursor keys and by pressing the < Enter > key. We needn't enter the Graphics — Results selection menu because currently there are only two results, that are automatically selected. Now we execute now the display command in the graphics menu and we obtain the bar representation of results. We can press the < F1 > (Help) function key to get help information on the meaning of several elements of the picture. However, we find the picture rather not legible. The standard scaling method (Normal scale ON) is based on the distances from lower to upper bounds. In our case, the changes of efficient values are much smaller than these distances. Thus, it seems that the second (Normal scale OFF) scaling method will be useful — it is based on distances between utopia and reservation values (or solution values if they are worse than reservations). The < F2 > key in the graphical representation toggles between both scaling methods; we press this key once.

Now the picture is legible and we can see that, in fact, the costs in the second region are decreased, but simultaneously the costs in the first region are increased and the pH levels

are decreased. Now we can either increase back reservations for the costs in the second region or increase aspirations and/or reservations for the pH levels in both regions. We try to explore the second possibility. We increase reservations for `pH12` `pH13` from 5.983 to 5.990 and from 6.265 to 6.270, respectively, and we calculate the efficient solution again, save it and look at the graphical representation — we press in sequence three function keys: < F3 > (Calculate), < F2 > (Save) and < F9 > (Graphics). The previous selection of objectives is still active and now three results are displayed.

The pH levels are now acceptable, but the costs in the first region are very high. To balance the costs in both regions we slightly increase reservations in the second region (from 1500 to 1700) and we decrease reservations in the first region (from 1090 to 900). The fourth obtained result seems to be close to the acceptable solution of the multiobjective decision problem.

5 References

Dreyfus, S. (1984). Beyond rationality. In M. Grauer, M. Thompson, A. P. Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Hettelingh, J. P. and L. Hordijk (1987). Environmental Conflicts: The Case of Acid Rain in Europe. RR-87-9, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kaden, S. (1985). Decision support system for long-term water management in open-pit lignite mining areas. In G. Fandel, M. Grauer, A. Kurzhanski and A. P. Wierzbicki (eds), Large Scale Modeling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Kaden, S. and T. Kreglewski (1986). Decision support system MINE — problem solver for nonlinear multi-criteria analysis. CP-86-5, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kreglewski, T. and A. Lewandowski (1983). MM-MINOS — an integrated decision support system. CP-83-63. International Institute for Applied Systems Analysis, Laxenburg, Austria.

Korhonen, P. (1985). Solving discrete multiple criteria decision problems by using visual interaction. In G. Fandel, M. Grauer, A. Kurzhanski and A. P. Wierzbicki (eds), Large Scale Modeling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Lewandowski, A., M. Grauer, A. P. Wierzbicki (1983). `DIDAS`: theory, implementation. In M. Grauer, A. P. Wierzbicki (eds), Interactive Decision Analysis, Proceedings Laxenburg 1983. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 229).

Lewandowski, A., T. Kreglewski, T. Rogowski, A. P. Wierzbicki (1987). Decision Support Systems of `DIDAS` Family. In A. Lewandowski, A. P. Wierzbicki (eds), Theory,

Software and Testing Examples for Decision Support Systems. WP-87-26, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Lewandowski, A., A. P. Wierzbicki (1988). Aspiration Based Decision Analysis and Support, Part I: Theoretical and Methodological Backgrounds. WP-88-3, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Makowski, M., and J. Sosnowski (1984). A decision support system for planning and controlling agricultural production with a decentralized management structure. In M. Grauer, M. Thompson, A. P. Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Messner, S. (1985). Natural gas trade in Europe and interactive decision analysis. In G. Fandel, M. Grauer, A. Kurzhanski and A. P. Wierzbicki (eds), Large Scale Modeling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Michalevich, M. (1986). Stochastic approaches to interactive multicriteria optimization problems. WP-86-10. International Institute for Applied Systems Analysis, Laxenburg, Austria.

Wierzbicki, A. P. (1983). A mathematical basis for satisficing decision making. *Mathematical Modeling* 3, 391–405.

Wierzbicki, A. P (1984). Models and Sensitivity of Control Systems. Elsevier, Amsterdam.

Wierzbicki, A. P. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spektrum* 8, 73–87.

A Installation guide

The distribution diskette contains the following files:

`DIDASN.EXE`	— compiled code of the program
`DEMO.MOD`	— simple nonlinear model (testing example)
`RAIN.MOD`	— nonlinear model used in Tutorial Example
`READ.ME`	— last time notes and corrections
`INSTALL.BAT`	— batch command file to install the program and both models on a hard disk or on a working diskette.

To install the program and models:

- insert the distribution diskette in floppy disk drive,
- change current drive and current directory to the drive and directory where you want to install the program,

- enter the command

  `a:install a:`

  where `a` is a drive letter of the drive where the distribution diskette is inserted (typically it is just drive A)

Installation procedure makes the sub directory `DIDASN`, writes the program code (`DIDASN.EXE`) there, next makes subdirectory `DIDASN\MODELS`, and writes into it two files with demonstrative nonlinear models together with examples of problems and results.

B Selection of colors

The program can work on `IBM PC/XT/AT` or compatible computers with Hercules Graphics Card (`HGC`), Color Graphics Adapter (`CGA`) and Enhanced Graphics Adapter (`EGA` or `VGA`). The user can select (independently for each particular type of a graphics card) his own set of colors/attributes to display several items on the screen by using menu entry Options — command Colors.

There are 9 items with some restrictions on selections of their colors:

Spreadsheet cells
Spreadsheet lines
Spreadsheet labels
Spreadsheet marker
Editing windows
Message windows
Warning messages
Error messages
Screen background

First three items form spreadsheets and, therefore, they have the same background color, the spreadsheet marker should have color that make it visible in all spreadsheet cells (empty and not empty).

Highlighted double-triangles marker selects one of 9 items and can be moved with < Up > and < Down > cursor keys. For the selected item < Home > and < End > cursor keys change foreground color, whereas < PgUp > and < PgDn > cursor keys change background color. The selected item is displayed in the colors/attributes currently choosen for it. Simultaneously names of the colors (such as displayed on the CGA adapter) are displayed on the right.

< Esc > key cancels all colors changes and causes return to the spreadsheet. < Enter > key makes the changes effective; additionally, the program asks about making these changes permanent. In case of answer < Y > (or < y >) the changes are recorded on a disk and will also be effective during subsequent runs of the program.

C Alternative solver

New solver based on nondifferentiable optimization algorithms from `NOA1` software package[2] was tested as an alternative to the standard one described in the Theoretical

[2]see: K. Kiwiel and A. Stachurski (1988). `NOA1`: A `FORTRAN` Package of Nondifferentiable Optimization Algorithms Methodological and User's Guide, WP-88-116. International Institute for Applied Systems Analysis, Laxenburg, Austria.

manual. Currently, it was implemented using non-standard hardware (**OA-Link** boards) as a remote solver running on different computer and interchanging all necessary data with the model created and calculated in the IAC-DIDAS-N system. The first results are hopeful, therefore, the **NOA1** solver will be included in the future versions of the system. However, the **NOA1** package is written in **FORTRAN** and cannot be directly connected to the IAC-DIDAS-N system written in **TURBO PASCAL**. Because of the size of both software packages some distributed, network hardware and software must be used.