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

Multi-Objective Modeling and Simulation for Decision Report

Wierzbicki, A.P.

IIASA Working Paper

WP-92-080

October 1992

Working Paper

Multi-Objective Modeling and Simulation for Decision Support

Andrzej P. Wierzbicki

WP-92-080
October 1992

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

Multi-Objective Modeling and Simulation for Decision Support

Andrzej P. Wierzbicki

WP-92-080
October 1992

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

MULTI-OBJECTIVE MODELING AND SIMULATION FOR DECISION SUPPORT

Andrzej P. Wierzbicki[1]

Summary.

Modeling and simulation of various physical, environmental or socio-economic processes is often a preliminary step for using the resulting models in decision support. With the advancements of computing technology and the methodology of decision support, it is now necessary to revise basic approaches to modeling and simulation: from the very beginning of model construction, they should aim at multi-objective analysis of the model while taking into account various optimization, sensitivity analysis and symbolic manipulation techniques treated as tools of such an analysis, not as goals. The paper illustrates how such an approach could increase the opportunities of comprehensive analysis of a model from a selected class - as an example, the class of nonlinear dynamic discrete-time models was chosen. A user-friendly format of formulating such models is discussed together with related problems of inverse, constrained and multi-objective simulation as well as structural differentiation and sensitivity analysis, fuzzy set processing and other related issues. As a new tool for the analysis of boundaries of sets generated by such models - for example, the α-level sets of fuzzy membership functions - an interactive method of using an elastic "electronic pencil" is proposed.

1. Introduction.

Mathematical, computerized modeling of expert knowledge is quite widely applied in various fields of science. A modeling exercise often starts simply as an intellectual challenge; however, the modeler should early enough decide whether his/her model will have only a *cognitive purpose* - that is, serve only him/her and some closely related specialists as a medium of scientific discussion - or rather a *knowledge encoding purpose* - that is, serve also a broader audience of users as a representation of knowledge on the specific subject.

A model is defined not only by its subject, but also by its purpose - see Wierzbicki (1984). In the knowledge-encoding case, the concerns of future users must be taken into account when constructing a model. If such a model should become a part of a decision support system (DSS), this might

- - - - - - - - - - - -

[1] Institute of Automatic Control, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland; this paper has been written during a stay at the International Institute for Applied Systems Analysis, Methodology of Decision Analysis Project, Laxenburg, Austria.

cause additional requirements. On the other hand, the methodology of using models in decision support has been considerably advanced in recent years and **some of the related techniques might be useful to modelers in specific fields even if their models have only cognitive purpose.** This is one of the main motivations of this paper, addressed not only to specialists in decision support methodology, but also to broader audience of modelers in various fields.

To see what might be learned from decision support methodology, we shall first recall (see Wessels et al., 1992) one of definitions of a DSS: *a DSS is a computerized system that supports its users in a rational organization and conduct of a decision process (or its selected phases) and, besides a data base, also contains a pertinent knowledge representation in the form of models of decision situations as well as appropriate algorithms for using these models.* A general scheme of a DSS is presented in Fig. 1.

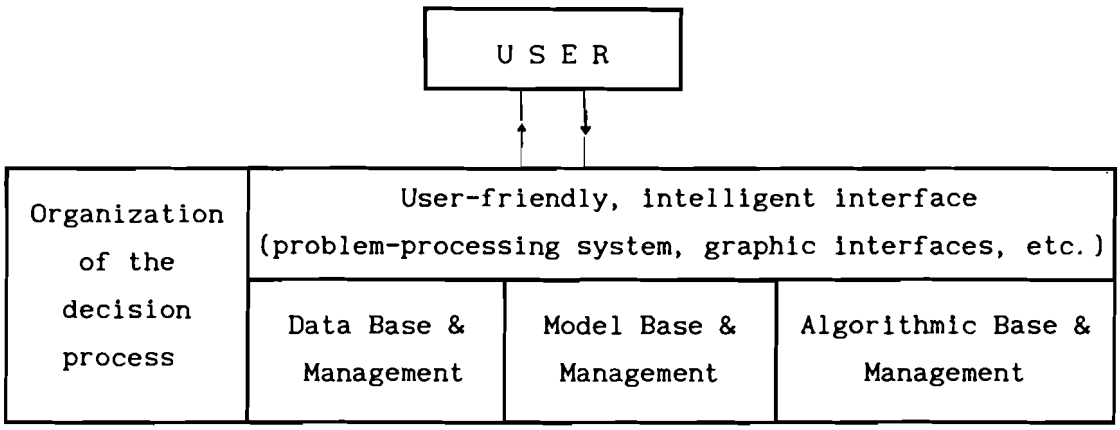


Fig. 1. A general scheme of the architecture of a DSS.

An important aspect of this definition is the concept of a **decision process**. Simon (1957) defined three main phases of this process: **intelligence, design and choice**. The phase of intelligence consists of observation, problem recognition, data gathering and diagnosis (Cooke and Slack, 1984). The phase of design contains problem specification, model and option specification; we might add here model fitting, identification and verification, simulation and preliminary analysis as well as model-based option or scenario generation (Lewandowski et al., 1989). The phase of choice consists of option evaluation and selection but might be augmented by multi-objective option analysis, sensitivity and post-optimal analysis

and include recourses to earlier phases. Finally, a fourth major phase - **implementation** - should be also included, together with the review or monitoring of results and with possible adjustments of the decisions in a feedback loop.

All these phases[2] might involve various users of a DSS - modelers and analysts in the phases of intelligence and design, actual decision makers or their advisors in the phase of choice, etc. The scheme in Fig. 1 stresses also a very important aspect of the definition of DSS: **the sovereign position of the user.**

In any phase of a decision process, *a DSS must not replace the user in his/her sovereign decision making*; it only helps him/her in various tasks. Operational decisions of repetitive nature might be automated to some degree; but even then the human decision maker has usually the authority to override the automatic equipment. In all other cases, the final responsibility for the outcomes of the decision process rests with the users of a DSS, especially when novel, strategic decisions are made and the main aim of a DSS is rather to enhance the creativity and intuition of the user[3] than to automate the decision making.

We should stress also some basic distinctions of the classes of models used in decision support. The first of them concerns **preferential** versus **substantive models**. The former are intended to represent the preferences of the user and are important in the phase of choice (although even then we should not rely too much on explicit preferential models, because we might automate this way the decision making too much and violate the principle of sovereignty of the user by replacing him/her in this crucial phase). The latter represent substantive knowledge and expertise about objective aspects of a decision situation and are the major subject of the phase of design, although they clearly influence also the phase of choice; it is a good practice to keep those two kinds of models separated in a DSS. Substantive models constitute actually the essence of decision support:

- - - - - - - - - - - -

[2] The structuring of these phases is not necessarily unique; a different specification of the phases of a decision process might be useful, for example, in the case of group decision processes or negotiations involving many decision makers.

[3] In the case of creative decision processes that allow for the role of intuition, the phases of a decision process might be defined as *recognition* of a decision problem, *deliberation or analysis* (which might include the three phases of Simon), *gestation, rationalization,* and finally *implementation* - see Wierzbicki (1992).

formulated by modelers - expert specialists and analysts - they form the basis of the knowledge encoded in a DSS. This paper is concerned mostly with substantive models, although their multi-objective formulation is related also to an elementary preference structure.

There are two basic classes of substantive models. In expert systems, artificial intelligence, etc., **logical models** are used; the methodologies of so-called *knowledge bases* (actually, model bases) and *inference engines* (actually, algorithmic bases) provide many powerful tools for decision support. But there are also inherent limitations to knowledge representation by logical models.

The second major class of **analytical models** can much better represent more complicated, non-binary cause and effect relations in terms of feedback loops; such models are also more diversified and richer than logical models. Analytical models can be in turn subdivided into *discrete* (discrete event systems, queuing theory models, Petri nets, etc.) and *continuous* (linear, nonlinear, dynamic models subdivided again into discrete-time and continuous-time, etc.). Finally, what might be most important for decision support, all analytical models can be treated either as single-objective, or multi-objective; if we truly want to enhance the creativity of various users, then substantive models must be formulated multi-objectively.

There are many possible other dimensions of classification of DSS; we just list them shortly (for a more detailed discussion, see Lewandowski et al., 1989, Wessels et al., 1992):

1) *Application area*: the functions and detailed specification of a DSS should be determined with the participation of future users, its user interface should rely on symbols and graphic representations typical for a given application area and thus well understood by the user, etc.; in short, a dedicated DSS should be **user-oriented.**

2) *Application type,* e.g. for **strategic, tactical, operational decisions**: a DSS for strategic decisions should support learning about the problem and innovative ways of solving it, whereas a DSS for operational decisions might concentrate on information processing and the optimization of typical solutions.

3) *Substantive model type,* e.g. **expert systems** with logical models and **analytical systems** with analytical (often, not quite precisely, called operations research type) models, with further subdivision - e.g. for

linear, nonlinear, dynamic continuous models, discrete models of various classes etc.

4) *Preferential model type,* or **rationality framework** together with selected algorithms of choice.

5) *The way of representing uncertainty*: we can use **probabilistic or stochastic** models, or **set-valued** models, or **fuzzy set** or even **rough set** models (see Pawlak, 1992). The minimal requirement to represent uncertainty when working with averaged (sometimes, not quite appropriately, called "deterministic") models is to include parameter ranges in the model and **sensitivity analysis tools** in the algorithmic base of a DSS.

6) *The type and variety of algorithmic tools* contained in the algorithmic base of a DSS: it should not be limited to a single algorithm but must contain a variety of them. Moreover, new **robust variants of** known **algorithms** must usually be developed before their inclusion in a DSS; they must run effectively for a broad class of models with wide parameter ranges. However, a concentration on the development of algorithms has its dangers: various **algorithms contained in a DSS should be treated as tools supporting the user, never as goals.**

7) *Principles of interaction with the user*: it is very often crucial how the user can influence a decision suggested to him/her by the DSS and how a suggested decision is explained to him/her. For a wide class of problems, the **reference point approach** (Wierzbicki 1980, 1986, Kallio et al. 1980) can be used to organize the interaction and to give the user a **full controllability** of efficient **option selection.**

8) *The number of users and the type of their cooperation*: a DSS might be designed to serve a **single user** or a **team of users** with the same interests and goals; quite different types of DSS would be designed to serve a **group of users** that might have different interests but must achieve a joint decision; yet different types of DSS are needed to support **bargaining and negotiations in game-like situations,** when each user can implement his/her own decisions.

Finally, a good DSS might be a mix of various types. For the rest of the paper we shall concentrate on a selected class of substantive models - the dynamic, discrete time, possibly nonlinear models with averaged uncertainty but supported by sensitivity analysis or fuzzy set processing - and comment on some tools developed in decision support that might be useful to a modeler even if he/she constructs a model for cognitive purposes.

2. Modeling and Simulation Are Multi-Objective.

Any experienced modeler would agree with such statement. When constructing a model, the modeler is never fully aware of its possible future applications, even if he/she knows its general purpose. The modeler cannot investigate only one outcome variable of the model; much more important are relations between many variables. Textbooks introduce usually a simple dualism between simulation and (implicitly - single-objective) optimization models; indeed, single-objective optimization models, in which only "the" objective function and some inflexibly interpreted decision constraints do matter, are in a sense closed and distinct from simulation models, in which more variables and types of relations can apparently be investigated in a more flexible way.

However, this is a mere appearance; the dualism simulation - optimization applies only in we limit our perception to single-objective optimization, treated as a goal not as an instrument. A modeler often finds it necessary to use additional algorithms, including optimization algorithms treated as varied instruments of more flexible and diversified simulation. In order to illustrate this, we shall consider typical components of a simulation model. Such a model might contain (see e.g Wierzbicki, 1984):

- **Actions** or **decisions** represented by *decision variables*. Although in simulation we might tend to treat decision variables jointly with other model parameters or "exogenous" variables, it is a good practice to distinguish such variables that might represent possible actions, called also sometimes control variables.

- Potential **objectives** represented by *outcome variables*. Although in optimization there is only one such variable and in simulation we treat almost all model variables as possible outcomes, again it is a good practice to distinguish variables that might have significance to future users of the models from those that are introduced only to help in modeling.

- Various **intermediate variables** - e.g. *state variables*, *balance variables* etc. They are useful for a flexible model formulation; even in single-objective optimization so-called **proxy** variables are often used though seldom stressed; in simulation, a good choice of intermediate variables is essential for the flexibility of modeling.

- *Parametric variables* or **parameters**. They might remain constant during model simulations but are essential for model validation and alternative model variants.

- **Constraining relations** - inequalities, equations etc. They determine the set of admissible decisions and might be divided into **direct** decision **constraints** that involve only decision variables and **indirect constraints** that involve also outcome and intermediate variables. Although in simulation we often use only direct constraints and tend to treat them rather flexibly, there are cases when indirect constraint are necessary and result in an essential increase of the difficulty of simulation.

- **Outcome relations** that determine how the outcome variables depend on the decision variables and parameters. They are often indirect, specified with the help of intermediate variables and equations such as state equations in dynamic models, sometimes with recursive or implicit formulae. In optimization, outcome relations are often treated simply as a part of constraints; while this is correct mathematically, it is certainly a bad practice in modeling or even in more sophisticated approaches to optimization.

- **A representation of model uncertainty**. Various representations of uncertainty might require the use of special, often complicated optimization algorithms.

Several methodological issues are related to the above classification; one of them is the distinction between **hard** and **soft constraints.** It is well known that constraints that are usually represented with a standard form - say, of an inequality - intend to model two quite different classes of phenomena of the real world. One of these classes contains balances that must be satisfied - such as the balance of energy in a physical model, or domains of model validity such as the edges of a table for a model of motion of a ball; these are so-called hard constraints. The other class contains balances that we would like to satisfy - such as the balance in a budget sheet; these constraints can be violated (at an appropriate cost) and are called soft constraints.

Soft constraints can be modeled even in single-objective optimization by appropriate penalty terms in the objective function; but then the question arises what are their permissible violations - thus, soft constraints are actually proxies for additional objectives. A good practice in modeling is to reflect on and take into account the possible hard or soft meaning of every inequality or even equation. In simple simulation,

soft constraints are recognized in a natural way: the modeler is interested how much they are violated, but does not think about them as real constraints. In more complicated simulation involving additional algorithms, the distinction between hard and soft constraints is useful. For example, if the model involves some indirect constraints (on outcome variables), in simple simulation we naturally assume that they are all soft; if some of them are actually hard, it is a sign that the model is not well formulated and these constraints should be rather treated as a part of implicit outcome relations that should be resolved with help of additional algorithms.

This leads to another methodological issue: the distinction between an **explicit** and **implicit model** formulation. Usually, simple simulation models are formulated in an explicit way: each outcome relation can be computed by an explicit formula and multiple outcome relations do not contain loops in the definition of subsequent variables. If we have to resolve a system of equations or inequalities to obtain model solution - which might happen if hard indirect constraints are included into outcome relations, or if multiple outcome relations contain loops - then the model has an implicit formulation and a special **resolving operation** must be defined together with the model. Such a resolving operation might be a fixed-point or another iterative algorithm - as in the case of economic equilibrium models - or an optimization algorithm. Actually, all systems of equations or inequalities might be resolved by optimization algorithms and there is (theoretically and in simplified textbooks) no need to distinguish implicit model formulation in optimization problems. On the other hand, more sophisticated optimization algorithms take also into account the structure and the possible implicit formulation of a model.

A resolving operation is iterative; additional time is thus needed to perform the iterations until an accuracy criterion is satisfied. However, there is a much deeper issue involved than just computational time. If the model represents some phenomena from real life, then the resolving operation should also have real-life interpretation - such as an equilibration mechanism on a market. The dynamics of the resolving operation should have also such an interpretation.

Suppose the model is of discrete-time dynamic type, but we would like to assume - for computational convenience - that the resolving operation is finished separately for each discrete time period. A good modeler should be, however, aware that the dynamics of a resolving operation actually

introduces a **second, faster time scale** in the model, and should address the question[4] "*Am I right in assuming that this time scale is really much faster then the original one?*". If the answer is negative, the model should not be simplified and an explicit mechanism of the resolving operation should be included in the formulation of model outcome relations, hopefully simulating the actual behavior of the process.

Another methodological issue is that of **model validation**[5]. Often, one has to estimate first model parameters; but after such parameter identification comes model validation. There are many methods of parameter estimation and model validation that depend on particular model type and are described in a broad literature. Most often, however, they are augmented by intuitive model validation, which usually relies on repetitive simulation that takes into account a particular model purpose. The model must be run many times by experts in given field of knowledge under changing assumptions about decisions and parameters or their scenarios - and the obtained outcomes must be compared against the formal knowledge and the intuition of experts[6].

An essential aspect of model validation is the treatment of various constraints. Typical for simple simulation and existing simulation languages is the inclusion of direct decision constraints only - say, of admissible ranges of decision variables; neither indirect constraints nor the distinction between hard and soft constraints are then included.

- - - - - - - - - - - -

[4] A failure in the analysis of such questions can lead to severely distorted conclusions from the model; consider, for example, the mistaken estimate that the transition of the reforming economies of Middle and Eastern Europe might take one or two years. Another assumption that might be mistaken is that a process which is continuous in time (as in physics, mechanics, etc.) must be modeled in a computer also in continuous time, by solving appropriate differential equations; this can lead to distortions when computing gradients, see the section on sensitivity analysis.

[5] According to modern philosophy of science - see e.g. Popper (1983) - a scientific theory concerning empirical phenomena should be rather invalidated, falsified than validated. But a computerized model is not a scientific theory, much rather an encoding of theoretical and empirical knowledge in a given field, with a definite purpose. Thus, the prevailing practice is to validate models, that is, to check whether they are good enough for their purposes.

[6] It has often been stressed that most valuable are models that can produce also counter-intuitive results; but the experts must be able to internalize such results, that is, explain to themselves why these results are obtained and check with their intuition (also by additional research and experiments) whether these results can occur in the real world; otherwise, counter-intuitive results are useless in learning.

However, expert users of simulation models are often interested in inverse simulation, in which desirable trajectories of model outcomes are specified by the user and decision variables should be chosen during the simulation to result in model outcomes close to the specified trajectories. Inverse simulation is particularly useful in model validation, but also in scenario generation. Moreover, good simulation techniques should make it possible to perform sensitivity analysis of simulated solutions along with simulation runs.

All these issues can be included in sufficiently sophisticated methods of simulation that use optimization techniques and multi-objective approaches as tools of simulation support. For example, soft constraints are most naturally interpreted as additional objectives in multi-objective modeling and optimization and thus included in the overall evaluation of a multi-objective solution; we shall illustrate later in the paper the related techniques. Therefore, modeling and simulation are indeed multi-objective: *multi-objective optimization models can be formulated as a natural, open extension of simulation models.*

3. A Format for Discrete-Time Dynamic Models.

Textbooks advise to formulate time-discrete dynamic models in the state equation form:

$$\mathbf{w}[t+1] = \mathbf{f}(\mathbf{w}[t],\mathbf{x}[t],t), \text{ where } t = 0,1,..,T \text{ and } \mathbf{w}[0] \text{ is given} \quad (1)$$

whereas $\mathbf{w}[t]$ is the vector of state variables and $\mathbf{x}[t]$ - the vector of decision variables, while the square brackets are used in order to stress the discrete character of time.

The concept of the state of a dynamic system is essential for a good understanding of dynamic modeling, but the form (1) **is a wrong standard of definition of dynamic multi-objective simulation models.** A good modeler does not think in such terms. He/she much rather thinks in terms of various outcomes or intermediate variables $y_i[t]$ that depend on selected actions or decision variables $x_i[t]$ or also on parametric variables $z_i[t]$. Moreover, he/she usually defines outcome variables recursively: the next ones depend on the previously defined ones, not only directly on decision variables.

The modeler should keep in mind his/her state variables $\mathbf{w}[t]$ - but as a part of outcome variables, $\mathbf{w}[t] = \{y_i[t]\}_{i \in I_{yw}}$ or $\mathbf{w}[t] = I_{wy}\mathbf{y}[t]$, where I_{yw} is either interpreted as an index set or as a linear selection operator determining such outcome variables that have the properties of the dynamic

state[7]. Thus, the modeler encodes his/her knowledge about structural properties of the problem or process modeled: every time he/she writes a formula for the variable $y_i[t]$, he/she implicitly defines also several index sets or selection operators I_{ix}, I_{iz} etc. that tell the computer which components of the vectors $\mathbf{x}[t]$, $\mathbf{z}[t]$ etc. should be taken into account when computing $y_i[t]$.

This might be illustrated by a simple example of a static structural model of a system of water quality control (adapted from Jaszkiewicz, 1992), see Fig. 2.

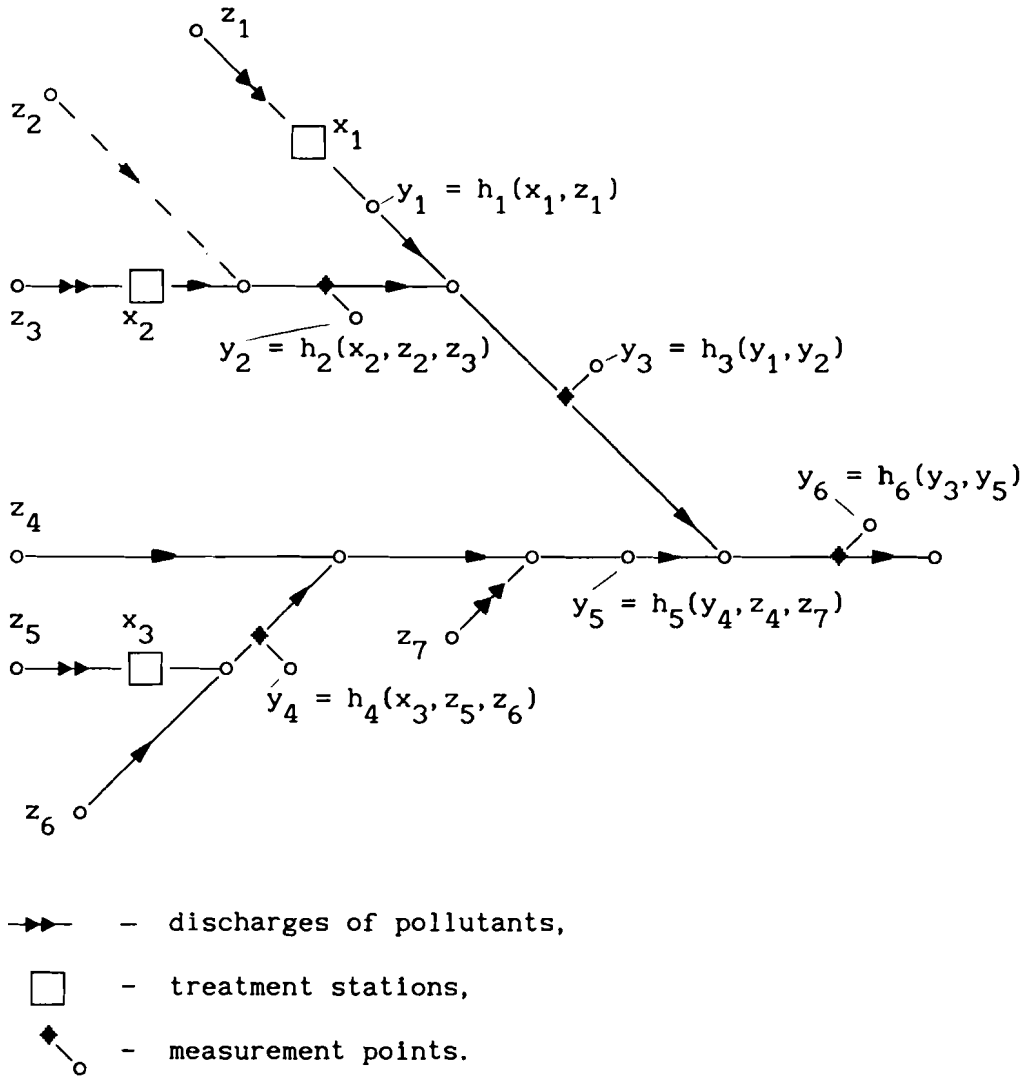


Fig. 2. The structure of a model of water quality control.

- - - - - - - - - - - -

[7] That is, such that their initial values must be specified in order to uniquely determine a solution of the model, given the (time-trajectories of) decision variables and parameters. An index set can be alternatively interpreted as a simplest linear operator - a matrix with entries 0 or 1 - and thus called a selection operator.

Note that not all y_i in the above example correspond to the actually measured water quality indicators; some are introduced as intermediate variables just for the convenience of modeling, though they might represent hypothetical measurement points. We could also construct a dynamic model for the example in Fig. 1, by taking into account the time needed by pollutants to reach certain points in this river basin; but we would like then to preserve the essential structure, implied by the topography of the tributaries.

Thus, **the right standard for dynamic discrete-time models** is rather as follows. We subsequently define formulae for various intermediate or possible outcome variables $y_i[t]$, while using previously defined $y_j[t]$ as convenient. We keep in mind that they might also depend on the state variables w[t-1] (it is more natural to remember that the state from the *previous* time instant might influence the *current* outcome):

$$y_1[t] = h_1(w[t-1], x[t], z[t], t), \tag{2}$$

$$y_2[t] = h_2(w[t-1], x[t], z[t], y_1[t], t),$$

.....

$$y_n[t] = h_n(w[t-1], x[t], z[t], y_1[t], .., y_{n-1}[t], t)$$

The modeler must be also aware which of his/her intermediate variables $y_i[t]$ are also state variables:

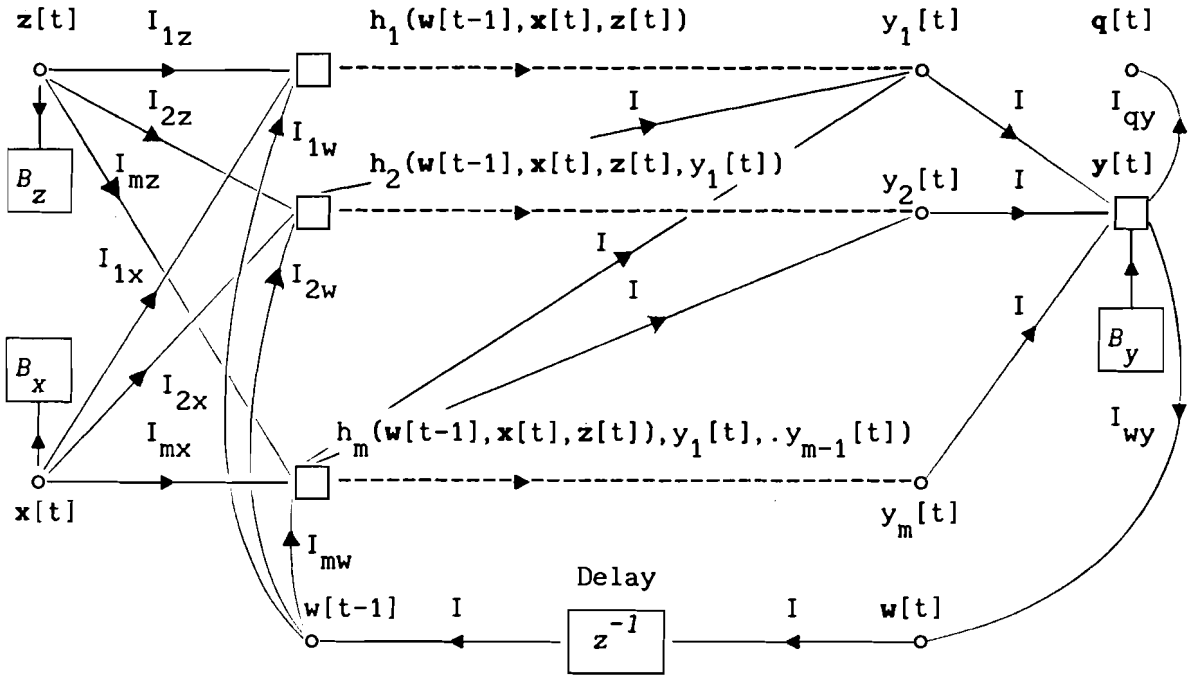
$$w[t] = \{y_i[t]\}_{i \in I_{wy}} \quad \text{or} \quad w[t] = I_{wy} y[t]; \quad w[0] \text{ - given} \tag{3}$$

The modeler is actually not obliged to enumerate the variables $y_i[t]$ consequently; an appropriate symbolic software could later check what is the best order of computations and whether there are no loops in the dependence of $y_i[t]$ on other $y_j[t]$. *If there are no loops*, we obtain this way **an explicit structural form of the dynamic model**. *If there are loops*, the symbolic software should warn the user. He/she can always reformulate the model by taking an "offending" equation out of the model definition and treating it as a soft constraint. Alternatively, the modeler can specify a resolving operation, while being careful that this operation does not change the concept of state variables.

In the explicit structural form, all constraints can be represented by bounds either on decision variables - the case of direct constraints, or on intermediate or outcome variables - the case of indirect constraints, interpreted as soft. Some intermediate variables might be defined solely

for the purpose of expressing such soft constraints; all hard constraints are assumed to be included in the definition of outcome relations.

The functions of symbolic software that might be used to analyze a structural form of a model are represented by a directed graph in Fig. 3:



o - variable nodes,

□ - information nodes,

--▸-- - processing arcs,

—▸— - information arcs,

z^{-1} - unit delay,

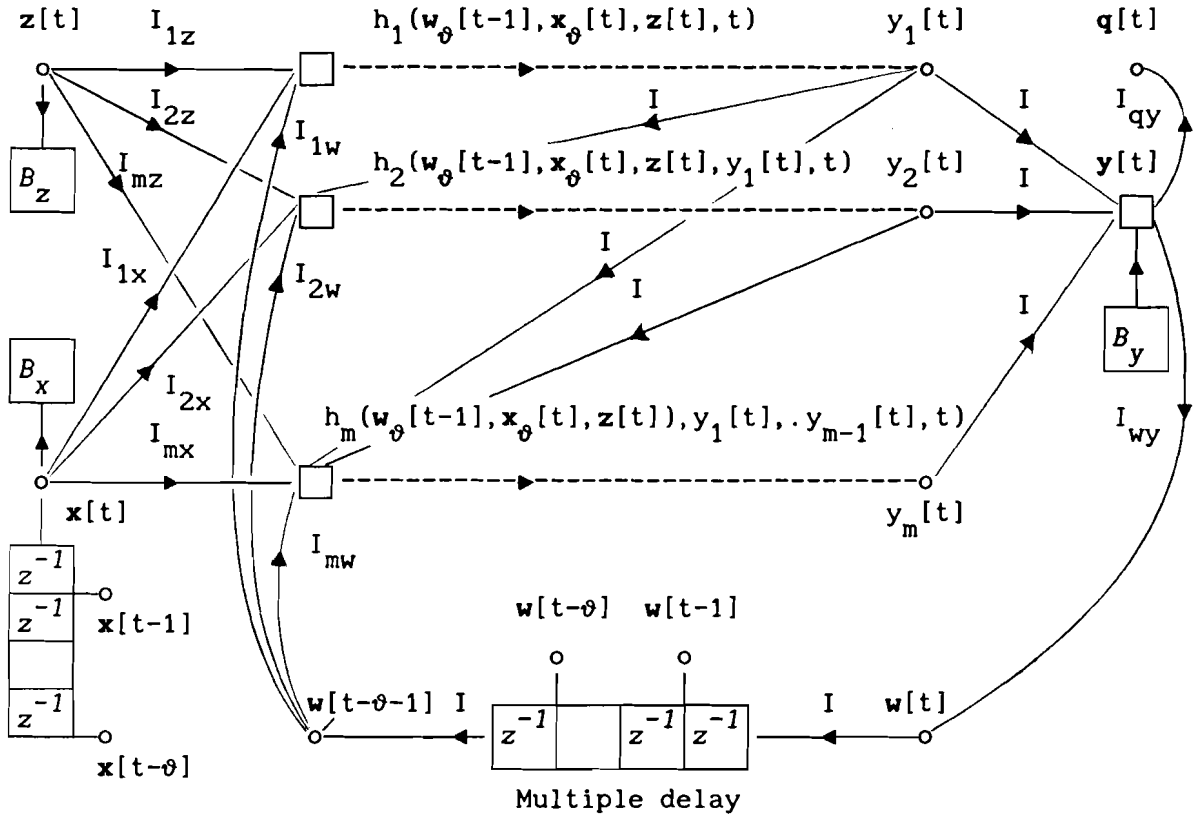
B_x, B_y, B_z - constraining bounds,

I_{ij} (I_{qy} etc.) - selection operators.

Fig. 3. Directed graph of an explicit structural form of a discrete-time dynamic model.

In the above graph, $\mathbf{q}[t] = I_{qy}\mathbf{y}[t]$ denotes outcome variables as selected by the user of the model, e.g. for the purposes of multiple-objective optimization. The selection operators I_{ij} are either determined by the symbolic analysis software once the modeler specified a model formula, or, as I_{qy}, specified by the choice of outcome variables by the user, or, as I_{wq}, determined symbolically but explicitly checked by the modeler. The graph in Fig. 3 represents the enumeration of variables $y_i[t]$ consistent with their symbolic analysis - telling us how the computer would enumerate and consecutively evaluate them - while the modeler might have specified them in a different order.

A similar graph might be constructed also for a broader class of dynamic models with multiple delays in state and decision variables:



o - variable nodes,

□ - information nodes,

--▸-- - processing arcs,

——▸—— - information arcs,

z^{-1} - unit delay,

B_x, B_y, B_z - constraining bounds,

I_{ij} (I_{qy} etc.) - selection operators.

Fig. 4. Directed graph of an explicit structural form of a discrete-time dynamic model with multiple delays.

The standard of defining the discrete-time dynamic model with multiple delays (for t = 1, ... T, while ϑ denotes the maximal delay) is as follows:

$$y_1[t] = h_1(\mathbf{w}_\vartheta[t-1], \mathbf{x}_\vartheta[t], \mathbf{z}[t], t), \tag{4}$$

$$y_2[t] = h_2(\mathbf{w}_\vartheta[t-1], \mathbf{x}_\vartheta[t], \mathbf{z}[t], y_1[t], t),$$

.....

$$y_n[t] = h_n(\mathbf{w}_\vartheta[t-1], \mathbf{x}_\vartheta[t], \mathbf{z}[t], y_1[t], .., y_{n-1}[t], t)$$

where:

$$\mathbf{x}_\vartheta[t] = \{\mathbf{x}[t], \mathbf{x}[t-1], \ldots, \mathbf{x}[t-\vartheta]\} \qquad (5)$$

$$\mathbf{w}_\vartheta[t-1] = \{\mathbf{w}[t-1], \mathbf{w}[t-2], \ldots, \mathbf{w}[t-\vartheta-1]\}$$

while $\mathbf{w}[t]$ is defined as in (3). The **complete state** $\tilde{\mathbf{w}}[t-1]$ - such that $\tilde{\mathbf{w}}[0]$ contains all initial conditions necessary to start a simulation of the model - includes also past decisions $\mathbf{x}[t-1], \ldots \mathbf{x}[t-\vartheta]$ -but not the current decision $\mathbf{x}[t]$, $\tilde{\mathbf{w}}[t-1] = \{\mathbf{w}_\vartheta[t-1], \mathbf{x}_\vartheta[t]\} \setminus \{\mathbf{x}[t]\}$.

The functions h_i in models (2), (3) or (4), (5) might, as indicated, depend explicitly on t. Often, such dependence is parameterized (expressed in terms of a parameter $z_j[t]$ that changes in time) and thus might be omitted; a software supporting the modeling might then just automatically repeat the model for all t = 1, ... T. However, it is better to have also the option that, after such automatic generation, the modeler can explicitly correct model formulae for any selected t.

4. Inverse, Constrained and Multi-Objective Simulation.

As indicated in previous sections, a modeler might need more diverse simulation tools than simple simulation. We can define now more precisely some related concepts.

Having selected some objective outcomes $\mathbf{q}[t] = I_{qy}\mathbf{y}[t]$, the modeler is often interested in entire **objective outcome trajectory**, which can be denoted as $\mathbf{q} = \{\mathbf{q}[t]\}_{t=1,..T}$, $q_i = \{q_i[t]\}_{t=1,..T}$. Thus, we can speak about an **objective space** $\mathbb{Q}$, $\mathbf{q} \in \mathbb{Q}$ (actually, a space of objective outcome trajectories; we assume the space to be normed with a norm $\|.\|$ selected by the modeler[8]). Similarly, we can speak about a (normed) **decision space** $\mathbb{X}$ with elements $\mathbf{x} = \{\mathbf{x}[t]\}_{t=1,..T}$ and an **admissible decision set** X_o in this space, defined e.g. by the direct constraints:

$$X_o = \{x \in \mathbb{X}: x_{ilow}[t] \le x_i[t] \le x_{iupp}[t], \text{ all t and i}\} \qquad (6)$$

The model is equivalent to a function transforming each point $\mathbf{x} \in X_o$ into a point $\mathbf{q} \in \mathbb{Q}$; although this function is usually quite complicated, we shall denote it simply by $F: X_o \to \mathbb{Q}$ and assume that this function is continuous and differentiable. The modeler does not need to know the explicit form of this function, but can learn about its properties by

- - - - - - - - - - - -

[8] Since the space is finite dimensional for discrete-time models with finite horizon T, any norm (Euclidean, Chebyshev, various l_p norms) might be used; mathematically, such a space is always complete - and thus, a Banach space.

performing simple simulation runs. While doing this, he/she becomes able to specify **reference trajectories** $\bar{q}$, $\bar{x}$ - interpreted as some desired outcome or some basic decision. Thus, the modeler might be interested in the following **inverse simulation** problem - find such $\hat{x}$ and $\hat{q} = F(\hat{x})$ that:

$$\hat{x} \in \underset{x \in X_o, q=F(x)}{\text{Argmin}} \|q-\bar{q}\| \tag{7}$$

where Argmin denotes the set of points minimizing - in this case, the norm. As indicated, $\hat{x}$ might be not unique; therefore, it is usually better to solve a **regularized inverse simulation problem**, with a parameter $\rho \in (0;1)$:

$$\hat{x} \in \underset{x \in X_o, q=F(x)}{\text{argmin}} (\rho\|q-\bar{q}\|+(1-\rho)\|x-\bar{x}\|) \tag{8}$$

where the use of argmin stresses the fact that the solution is unique[9]; for $\rho \to 0$, problem (8) gives a regularized solution to problem (7). A generalization of this problem is **softly constrained simulation**: the modeler would like to determine such decisions (or model parameters, which he/she can always redefine as a part of decisions), possibly close to a specified decision reference trajectory $\bar{x}$, that result in a possibly smallest violations of some specified constraints on objective outcomes. Suppose indirect soft outcome constraints are given by:

$$q_i[t] \le q_{iupp}[t], \text{ all } i \text{ and } t \tag{9}$$

Correspondingly, the **violation of soft constraints** must be defined, e.g.:

$$\Delta q = \{(q[t]-q_{upp}[t])_+\}_{t=1,..T} \tag{10}$$

where $(.)_+$ denotes the (component-wise) positive part of a vector. Hence, the problem (8) can be generalized to:

$$\hat{x} = \underset{x \in X_o, q=F(x)}{\text{argmin}} (\rho\|\Delta q\|+(1-\rho)\|x-\bar{x}\|) \tag{11}$$

All these problems of inverse, regularized inverse and softly constrained simulation **can be again generalized and considered as special cases of multi-objective reference trajectory optimization.** For this purpose, we note first that by specifying requirements on reference decisions, we actually include decisions as a (trivial) part of the outcomes. Thus, the modeler should be simply able to include decisions as components of the outcome space.

- - - - - - - - - - - -

[9] Locally, under mild conditions concerning Lipschitz-continuity of the model; the model might be not convex (the images of convex sets such as X_o might be not convex in Q), thus global uniqueness cannot be guaranteed.

An important concept is modelers indication of his/hers preference structure - actually, a simple statement how to treat various components of objective space. For example, the modeler can indicate that some outcome components should be **stabilized**, that is, kept possibly close to their reference values. If all outcomes are stabilized, this clearly corresponds to the inverse or regularized inverse simulation. But some outcomes might be also **minimized versus reference values**, that is, minimized if they are above the reference values and possibly still minimized, but with smaller weighting coefficients, if they are below reference values; this includes the case of softly constrained simulation. Similarly, outcomes that are **maximized versus reference values** are defined (maximized below the reference values, maximized with smaller weights above the values).

Because we are dealing with the trajectories of outcomes and references in the dynamic case, there are also other possibilities: e.g. the increments of some outcome trajectories might be stabilized close to zero, in order to obtain a smooth trajectory, but their final value at t = T might be maximized, see also Makowski et al. (1989) for other possible formulations. However, the modeler should be able to specify his/hers outcome space arbitrarily - e.g. include the increments of an outcome as a trajectory and the final value of the same outcome as a separate outcome component. Hence, it is sufficient to assume that his/hers indications of preference structure are uniform on trajectories. This means that if a trajectory should be maximized, then the maximization concerns uniformly all time instants t = 1, ... T, and if a trajectory should be stabilized, then the Chebyshev norm corresponding to its maximal deviation from the reference trajectory for t = 1, ... T should be used and minimized.

Mathematically, the indication of preference structure is equivalent to a definition of a **domination cone** in the objective outcome space:

$$C = \{\Delta q \in Q: \Delta q_i \leq 0, i=1,..p_1; \Delta q_i \geq 0, i=p_1+1,..p_2; \Delta q_i = 0, i=p_2+1,..p\} \qquad (12)$$

where $\Delta q_i = q_i - \bar{q}_i$ and the first p_1 objectives are assumed to be minimized, the next from $p_1 + 1$ to p_2 - maximized, and the last from $p_2 + 1$ - stabilized. In relation to such generalized domination cone, various specific techniques of multi-objective optimization can be compared for their effectiveness, see e.g. Wierzbicki (1986). Many of them are ineffective, both for the specific domination cone (12) and because of other drawbacks (for example, a simple linear combination of weighted objectives is the possibly worst technique: it cannot work well in

nonconvex cases, but even in convex cases it does not give the modeler a full controllability of selection of outcomes). The most effective technique consists in minimizing an **order-consistent achievement scalarizing function**[10], e.g. of the form:

$$s(\mathbf{q},\bar{\mathbf{q}}) = \max_{1\le t\le T} \max_{1\le i\le p} \delta q_i[t] + \varepsilon \sum_{t=1}^{T} \sum_{i=1}^{p} \delta q_i[t] \qquad (13)$$

where $\varepsilon > 0$ is a small coefficient and:

$$\delta q_i[t] = (q_i[t]-\bar{q}_i[t])/(q_{iupe}[t]-q_{iloe}[t]),\ i = 1,..p_1$$

$$\delta q_i[t] = (\bar{q}_i[t]-q_i[t])/(q_{iupe}[t]-q_{iloe}[t]),\ i = p_1+1,..p_2$$

$$\delta q_i[t] = |q_i[t]-\bar{q}_i[t]|/(q_{iupe}[t]-q_{iloe}[t]),\ i = p_2+1,..p$$

while $q_{iupe}[t]$, $q_{iloe}[t]$ might be equal initially to the bounds $q_{iupp}[t]$, $q_{ilow}[t]$ specified by the modeler. More refined estimates of such bounds restricted to a narrower set of outcomes - such that are efficient in the sense of the cone C - can be computed for the modeler by appropriate software, see e.g. Lewandowski et al. (1989).

Although the function $s(\mathbf{q},\bar{\mathbf{q}})$ has discontinuous derivatives at $\mathbf{q} = \bar{\mathbf{q}}$, its nondifferentiability can be accounted for by appropriate optimization techniques. The use of such a function has been tested in many multicriteria decision support systems and found a good instrument of multi-objective model analysis. For example, the system DIDAS-N uses this function with models of a similar format to that described above (only essentially of a static character, see Kręglewski et al. 1989). Thus, by solving the problem:

$$\hat{\mathbf{x}} = \operatorname*{argmin}_{x\in X_o,\, q=F(x)} s(\mathbf{q},\bar{\mathbf{q}}),\ \hat{\mathbf{q}} = F(\hat{\mathbf{x}}) \qquad (14)$$

we can not only cover its special cases of inverse, regularized inverse and softly constrained simulation (with Chebyshev norm), but also obtain attainable outcomes $\hat{\mathbf{q}}$ which are efficient[11] with respect to the cone C and have the desirable property of being Lipshitz-continuously controllable by changes of $\bar{\mathbf{q}}$ specified by the modeler. This property results in a

- - - - - - - - - - - -

[10] Actually, an achievement function should be maximized; but for the sake of a better interpretation of multi-objective trajectory optimization, we change here signs when necessary and assume that this function is minimized.

[11] In fact, not only efficient but also ε-properly efficient, such that the trade-off coefficients between their components (normalized by the scales $q_{iupe}[t]-q_{iloe}[t]$) are bounded by the factor $1 + 1/\varepsilon$.

particular flexibility of **scenario generation** with the help of reference-point approach: we can assume a reference trajectory $\bar{\mathbf{q}}$, compute the corresponding scenario $\hat{\mathbf{q}}$ as in (14) and, if necessary, adjust the scenario by small changes of $\bar{\mathbf{q}}$.

5. Sensitivity Analysis for Multi-Objective Models.

When developing software that would support inverse, regularized inverse and softly constrained simulation together with multi-objective analysis of discrete-time dynamic models, an essential problem is related to the differentiation of the model. There exist rather robust nonlinear optimization algorithms[12] that can be adapted for the dynamic case; but in order to apply them, gradients of objective outcomes and subgradients of the achievement scalarizing function with respect to decision variables must be computed. On the other hand, modelers do not use such gradients (which would be useful to them, for example, in estimating the parametric sensitivity of their models) for a very good reason: even if a modeler would have the time to analyze structural relations between various derivatives in his/hers complicated model - a task additionally complicated by the dynamic structure of the model - *a programming by hand of a large number of derivatives leads inevitably to errors.*

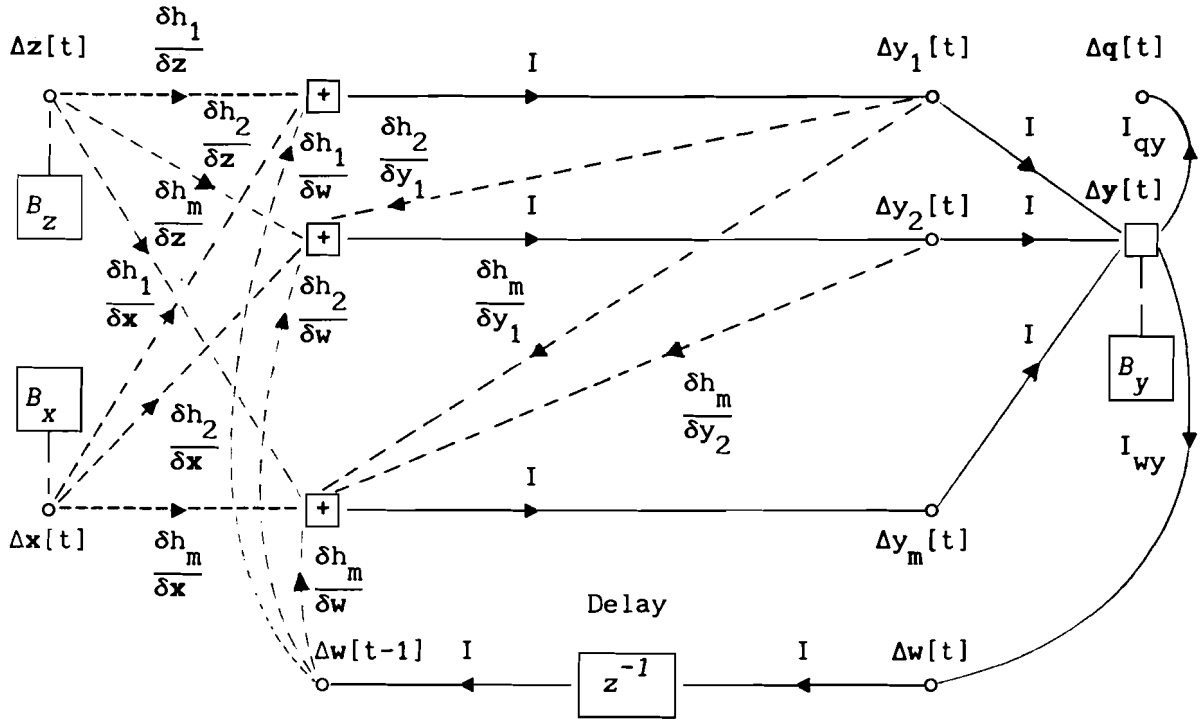
Today, there exists also symbolic differentiation software. However, it supports mostly the differentiation of single formulae - and the computer time necessary for the simplification of derivative expressions explodes exponentially with the complexity of the formula. On the other hand, there are known methods of **structural differentiation**, based e.g. on a structural form of a generalized implicit function theorem (see Wierzbicki, 1984); it takes much less computer time to symbolically differentiate a complicated formula, if we split it into several parts and define an explicit structural model similarly, say, as in Fig. 1.

The specific form of the directed graph in Fig. 3, with its *distinction between information arcs and processing arcs, was chosen in order to automate its differentiation.* In terms of this graph, a generalized implicit function theorem can be stated as follows - see Fig. 5. Suppose all functions h_i in the graph form Fig. 3 are differentiable, and

- - - - - - - - - - - -

[12] In DIDAS-N, a combination of a conjugate direction algorithm with a shifted penalty (augmented Lagrangian) method of accounting for indirect constraints has turned out to be quite robust; additional modifications of the conjugate direction algorithm for the dynamic case are also possible, see e.g. Wierzbicki (1984).

the input variables **x**, z in this graph are changed by increments **Δx**, **Δz**. Then the full differentials of all variables in the graph - Δy_i, **Δw**, **Δq** - are determined by a linear **primal associated sensitivity model** as in Fig. 5, while the initial state for this model depends on an assumed increment **Δw**[0] (that is, **Δw**[0] = 0 if we investigate the pure impact of increments **Δx**, **Δz**, but we can also assume e.g. **Δx** = 0, **Δz** = 0 and investigate the sensitivity to pure changes of initial conditions **Δw**[0]).



o - variable nodes,

□ - information nodes,

[+] - summation nodes,

--▸-- - processing arcs,

—▸— - information arcs,

z^{-1} - unit delay,

B_x, B_y, B_z - constraining bounds (usually treated as inactive),

I_{ij} (I_{qy} etc.) - selection operators.

Fig. 5. Directed graph of the primal associated sensitivity model of a discrete-time dynamic model form Fig 3.

Note that the associated sensitivity model from Fig. 5 is obtained by shifting the derivatives of a processing arc from Fig. 3 to the respective preceding information arcs and by exchanging the character of these arcs; the information gathering node before a processing arc in Fig. 3 is changed to a summation node in Fig. 5. This operation, together with a symbolic

determination of all necessary partial derivatives, can be performed by symbolic manipulation software[13]; thus, an associated sensitivity model can be automatically generated. This conclusion applies even to **implicit models with a specified resolving operation** (see Wierzbicki, 1984) under the assumption that this resolving operation functions also for the linearized model.

While solving an associated sensitivity model such as in Fig. 5 numerically, one must remember that the linear operators represented by the vectors of partial derivatives in this model (such as $\frac{\delta h_i}{\delta \mathbf{x}}$) depend actually nonlinearly on the input and outcome variables **x**[t], **z**[t], **w**[t-1], **y**[t] of the basic model from Fig. 3; thus, these two models might be computed either parallel for each instant of time, or sequentially. Moreover, if a formula for a function h_i is rather complicated, the symbolic software can split it in a more detailed structural form (by introducing additional intermediate variables); the software can also search many derivative formulae for repeated terms (which often occur in partial differentiation). This shortens the time needed not only for the symbolic simplification of formulae for derivatives, but also for the numeric evaluation of the model. In fact, the associated sensitivity model from Fig. 5 is usually evaluated numerically much faster then the basic model from Fig. 3.

The associated sensitivity model from Fig. 5 is useful when estimating the impact of given changes of input variables on **all** model variables - similarly, as a model from Fig. 3 might be used while checking not the linear estimates, but the full impact of finite increments of inputs. Therefore, the model is called **primal**.

However, it would be difficult to use this model to determine the gradient of **one**, selected outcome with respect to all components of an input variable. For example, the decision variable **x** has the number of components equal to T times the number of components of **x**[t]; hence we would have to run the primal sensitivity model that many times while assuming appropriate unit vectors as inputs). Since we need the gradients for optimization, we use also a different, **dual** form of the associated sensitivity model. While this dual form is rather complicated, this should not bother the modeler: it could be prepared also by symbolic, structural

- - - - - - - - - - - - -

[13] It has been implemented until now, however, only in the DIDAS-N system. Commercially available symbolic manipulation software usually does not include structural differentiation.

differentiation software. A good simulation software should in near future[14] include such possibilities.

A derivation of the dual sensitivity model is given in Appendix; we present here its final form. Suppose we want to compute the gradients of a selected outcome component $q_i[\tau]$ at a selected time $\tau = 1,\ldots,T$ with respect to all decisions $\mathbf{x}[t]$ and parameters $\mathbf{z}[t]$, $t = 1,\ldots\tau$. This outcome component is treated as a function of entire trajectories $\mathbf{x}$, $\mathbf{z}$, $q_i[\tau] = J_{i\tau}(\mathbf{x},\mathbf{z})$. Actually, this component depends also on $\mathbf{w}$, $\mathbf{y}$ which in turn depend on $\mathbf{x}$, $\mathbf{z}$ through model equations, but precisely this dependence must be taken into account and a reduced gradient computed. We must then solve - in a reverse direction of time, "counting down" from $t = \tau-1$ to $t = 1$ - the adjoint equations:

$$\eta[\tau-1] = -\left(\frac{\delta \mathbf{h}}{\delta \mathbf{w}};\tau\right)^*\left(I - \frac{\delta \mathbf{h}}{\delta \mathbf{y}};\tau\right)^{*-1} I^*_{qiy};$$

$$\eta[t-1] = \left(\frac{\delta \mathbf{h}}{\delta \mathbf{w}};t\right)^*\left(I - \frac{\delta \mathbf{h}}{\delta \mathbf{y}};t\right)^{*-1} I^*_{wy}\eta[t] \text{ for } t = \tau-1,\ \tau-2,\ \ldots\ 1 \qquad (15)$$

where $\mathbf{h} = \{h_1, \ldots h_i, \ldots h_m\}$, $\frac{\delta \mathbf{h}}{\delta \mathbf{w}};t = \frac{\delta \mathbf{h}}{\delta \mathbf{w}}(w[t-1],\mathbf{x}[t],\mathbf{z}[t],\mathbf{y}[t],t)$ etc, the star * denotes the transpose of a vector or a matrix[15]. The matrix $\frac{\delta \mathbf{h}}{\delta \mathbf{y}};t$ is composed of rows $\frac{\delta h_i}{\delta \mathbf{y}};t$; the matrix $I - \frac{\delta \mathbf{h}}{\delta \mathbf{y}};t$ is always invertible. Parallel, we can compute the gradient equations (where I_{qiw} denotes the selection operator for the component q_i):

$$\frac{\delta J_{i\tau}}{\delta \mathbf{x}^*[t]} = -\left(\frac{\delta \mathbf{h}}{\delta \mathbf{x}};t\right)^*\left(I - \frac{\delta \mathbf{h}}{\delta \mathbf{y}};t\right)^{*-1} I^*_{wy}\eta[t],\ t = 1,\ .\ \tau-1;$$

$$\frac{\delta J_{i\tau}}{\delta \mathbf{x}^*[\tau]} = \left(\frac{\delta \mathbf{h}}{\delta \mathbf{x}};\tau\right)^*\left(I - \frac{\delta \mathbf{h}}{\delta \mathbf{y}};\tau\right)^{*-1} I^*_{qiy} \qquad (16)$$

- - - - - - - - - - - - -

[14] The development of such software was started this year in the Institute of Automatic Control, Warsaw University of Technology; it might take yet another year.

[15] While computing reduced gradients, it is useful to distinguish the row vector form of a gradient, denoted here e.g. by $\frac{\delta J_{i\tau}}{\delta \mathbf{x}[t]}$, from its column vector form, denoted here by $\frac{\delta J_{i\tau}}{\delta \mathbf{x}^*[t]}$; $\mathbf{x}[t]$ and $\Delta\mathbf{x}[t]$ are assumed to be column vectors. In return for this somewhat strange convention we obtain a simple notation for the full differential $\Delta J_{i\tau} = \frac{\delta J_{i\tau}}{\delta \mathbf{x}[t]}\,\Delta\mathbf{x}[t]$ (a row vector times a column vector is a scalar product) and an insight why we must have many transposed entries in more complicated gradient computations.

$$\frac{\delta J_{i\tau}}{\delta z^*[t]} = -\left(\frac{\delta h}{\delta z};t\right)^*\left(I - \frac{\delta h}{\delta y};t\right)^{*-1} I^*_{wy}\eta[t],\ t = 1, \ldots\tau-1;$$

$$\frac{\delta J_{i\tau}}{\delta z^*[\tau]} = \left(\frac{\delta h}{\delta z};\tau\right)^*\left(I - \frac{\delta h}{\delta y};\tau\right)^{*-1} I^*_{qiy} \qquad (17)$$

If we assume that the parameters $\mathbf{z}[t] = \mathbf{z}$ remain constant for all time instants, then the gradient equation with respect to parameters modifies to the form:

$$\frac{\delta J_{i\tau}}{\delta z^*} = \left(\frac{\delta h}{\delta z};\tau\right)^*\left(I - \frac{\delta h}{\delta y};\tau\right)^{*-1} I^*_{qiy} - \sum_{t=1}^{\tau-1}\left(\frac{\delta h}{\delta z};t\right)^*\left(I - \frac{\delta h}{\delta y};t\right)^{*-1} I^*_{wy}\eta[t] \qquad (18)$$

Additionally, if we want to determine the sensitivity of the outcome component $q_i[\tau]$ to the initial conditions $\mathbf{w}[0]$, we obtain:

$$\frac{\delta J_{i\tau}}{\delta w^*[0]} = -\eta[0] \qquad (19)$$

It might seem that we must solve the dual sensitivity model also a rather large number of times, if we want to compute gradients of many outcome variables in a dynamic model - where an entire outcome trajectory consisting of T components can be also considered as an objective outcome. However, if we use a scalarizing achievement function of the form (13) and want to compute its subgradients - which are determined by convex combinations of appropriate gradients related to this function - we can exploit its structural properties and reduce the number of repeated solutions of the dual sensitivity model to a few times.

We must anyway solve first the basic model (from Fig. 3); but we augment it by an additional outcome variable:

$$q_o[T] = \varepsilon \sum_{t=1}^{T}\sum_{i=1}^{p} \delta q_i[t];\ q_o[t] = \varepsilon \sum_{i=1}^{p} \delta q_i[t] + q_o[t-1],\ q_o[0] = 0 \qquad (20)$$

where ε and $\delta q_i[t]$ are defined as in (13). Equation (20) implies that $q_o[t]$ is also an additional state variable, hence the number of adjoint variables increases correspondingly. We need actually all $\delta q_i[t]$, $i = 1,\ldots,p$, $t = 1,\ldots,T$, as additional outcome variables, but can treat them as re-scaled original outcome variables which reduces computations. When solving the basic model with given $\mathbf{x}$, $\mathbf{z}$, we check which $\delta q_i[t]$ are **active** - that is, which of them contribute to the maximum of the following nondifferentiable, auxiliary term:

$$\tilde{q}_1[T] = \max_{1\le t\le T} \max_{1\le i\le p} \delta q_i[t] \qquad (21)$$

Denote by τ_1, τ_2, .. the time instants at which the maximum in (21) is attained - there are usually only a few of them - and by $I_{\tau 1}$, $I_{\tau 2}$, .. the sets of indices i of such $\delta q_i[\tau_1]$, $\delta q_i[\tau_2]$, .. that are equal to the maximum; the total number of such cases, $|I_{\tau 1}|+|I_{\tau 2}|+$ ... is also usually small. Because $s(q,\bar{q}) = q_o[T] + \tilde{q}_1[T]$, it is enough to determine the gradients related to $q_o[T]$ and to $\delta q_i[\tau_1]$, $\delta q_i[\tau_2]$, .. for $i \in I_{\tau 1}$, $I_{\tau 2}$, .. in order to determine vectors forming the basis of the subgradient of $s(q,\bar{q})$. Thus, we must solve the dual sensitivity model $1 + |I_{\tau 1}|+|I_{\tau 2}|+$ .. times. But a solution of the dual sensitivity model takes usually less time than that of the basic model, hence the necessary computations to prepare data even for a nondifferentiable optimization algorithm are not excessive.

All these preparatory re-formulation and additional computations can be hidden from the software user - the modeler - in a common operation of reference-point optimization. However, he/she should be aware what are the essential elements of this operation. For example, if he/she uses typical simulation software for time-continuous dynamic models with an automatic time-discretization (such as the automatic choice of a step-size in a Runge-Kutta method), and then tries to compute gradients through an automatic discretization of continuous adjoint equations, the results will be severely distorted[16] Moreover, the modeler should be able to use the full possibilities. For example, when trying to adjust model parameters to given data, he/she should be able to define him/herself an additional model outcome such as $q_o[T]$, but expressing e.g. the sum of squares of deviations of selected model outcomes from available data. The modeler might then run the optimization algorithm contained in the software in order to find a best fit, or just to check what is the sensitivity of such additional outcome on some parameters not included in optimization, etc.

6. Related Issues: Fuzzy Set Processing.

When simulating models with time-horizons of several decades of years (such as models of sustainable development issues, climatic, demographic etc. models), a frequent case relates to a non-probabilistic uncertainty of

- - - - - - - - - - - -

[16] In order to obtain correct gradient expressions, it is necessary to examine the time-discrete model as generated by the automatic discretization (Runge-Kutta, etc.) method and then compute the gradient expressions as in (15) .. ((18). A failure to to this has lead often to inefficiency in applying dynamic optimization algorithms.

parameters. We might have several parameter values, resulting e.g. from different theoretical approaches, but the best what we can say about these parameter values is an assessment of their **possibility distribution.** Such a distribution expresses our judgment to which degree these values might be true - in terms of multi-valued, not binary logic. Such a distribution is also called a **membership function** for a **fuzzy set** of these values, as shown in Fig. 6.

Suppose we have a rather complicated model with outcomes that depend on such fuzzy parameters and want to address the question: *what might be the possibility distribution*, i.e. the membership function *of a selected model outcome*? The answer to this question is rather elementary, but involves cumbersome computations; possibly for this reason, fuzzy set processing is not used in model simulation. We shall show that, while using multi-objective simulation software with optimization support, fuzzy set processing can be also performed.

Even in a dynamic model, the dependence of a selected outcome $q_i[\tau] = q_i$ on a given parameter z_j is a static function (although, might be, quite complicated); we shall denote it by $q_i = F_{ij}(z_j)$. The transformation of a given membership function $\mu_j(z_j)$ by F_{ij} into the corresponding membership function $\mu_i(q_i)$ might be defined by the formula (cf. e.g. Kacprzyk et al. 1988):

$$\mu_i(q_i) = \bigvee_{q_i=F_{ij}(z_j)} \mu_j(z_j) \tag{22}$$

where $\vee$ denotes the operation of fuzzy logical '*or*', usually defined as the maximum over $\mu_j(z_j)$ such that z_j satisfies the indicated condition. Assume that μ_j has a bounded and connected support - that there is a bounded interval $Z_{oj} = \{z_j \in \mathbb{R}^1: \mu_j(z_j)>0\} = (z_{jlow}; z_{jupp})$. Moreover, assume F_{ij} is continuous; then the corresponding support of μ_i is also bounded and connected. When approximating the values of $\mu_i(q_i)$ numerically we would first establish the interval $Q_{oi} = \{q_i \in \mathbb{R}^1: \mu_i(q_i)>0\} = (q_{ilow}; q_{iupp})$. Let $\bar{Z}_{oj} = \langle z_{jlow}; z_{jupp} \rangle$; the corresponding bounds of the interval Q_{oi} can be determined as:

$$q_{ilow} = \min_{z_j \in \bar{Z}_{oj}} F_{ij}(z_j); \quad q_{iupp} = \max_{z_j \in \bar{Z}_{oj}} F_{ij}(z_j) \tag{23}$$

A direct approach to approximate (22) numerically might be not the most effective one. Suppose we select in Q_{oi} a number of uniformly distributed points $\bar{q}_i$, distant from each other by an increment $\Delta\bar{q}_i$, and

approximate the formula (22) on intervals $\langle \bar{q}_i; \bar{q}_i + \Delta\bar{q}_i \rangle$:

$$\mu_i(\bar{q}_i) = \max_{z_j \in Z_{\bar{q}_i}} \mu_j(z_j); \quad Z_{\bar{q}_i} = \{z_j \in Z_{oj}: \ \bar{q}_i \le F_{ij}(z_j) \le \bar{q}_i + \Delta\bar{q}_i\} \qquad (24)$$

However, as illustrated in Fig. 6, the sets $Z_{\bar{q}_i}$ might be disconnected, thus we would need a rather complicated global optimization algorithm to solve (24).

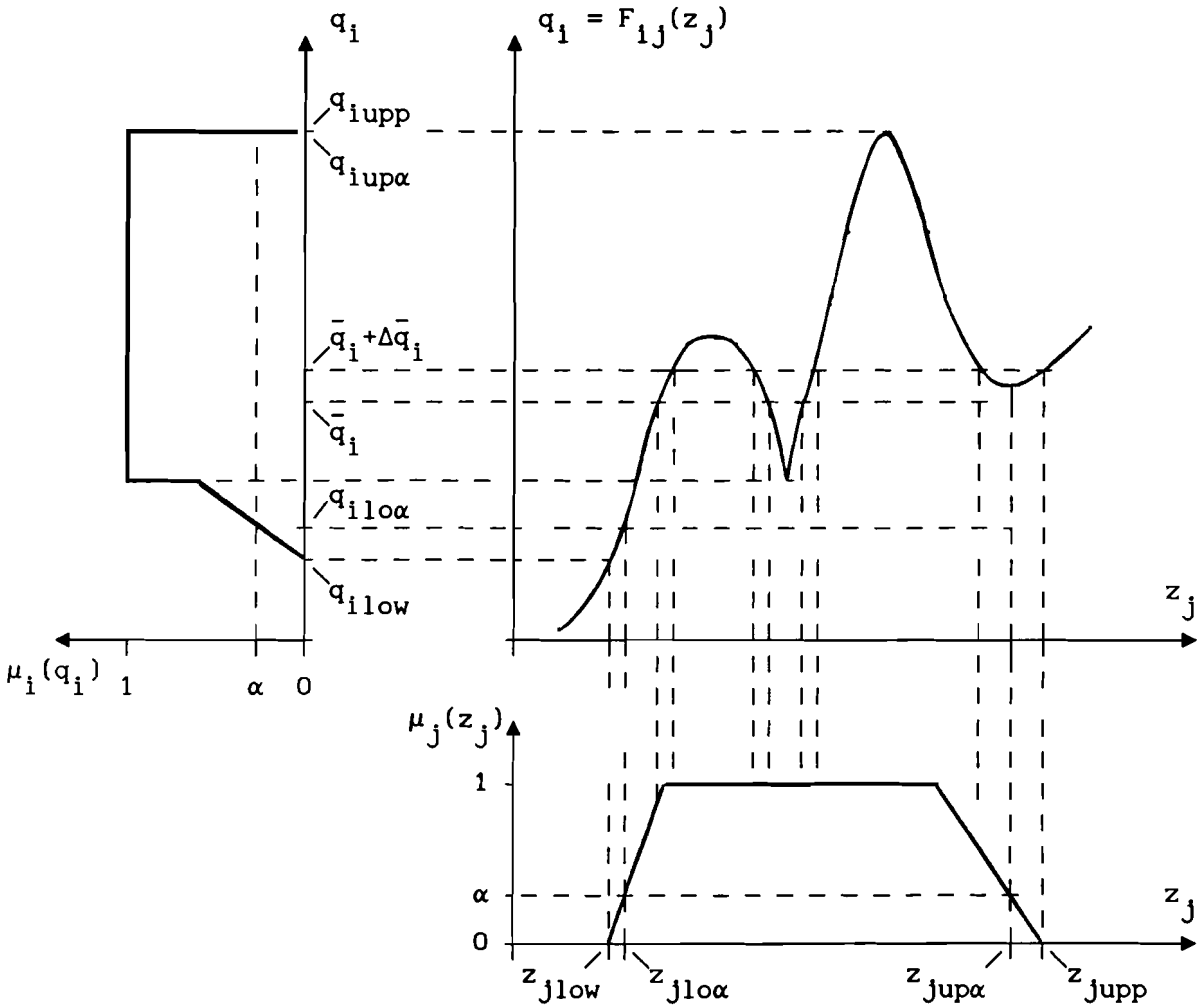


Fig. 6. Processing of fuzzy sets through a nonlinear function.

A different, possibly better approach is to compute how the α-level sets of function μ_j are transformed into the α-level sets of function μ_i:

$$Z_{\alpha j} = \{z_j \in Z_{oj}: \ \mu_j(z_j) \ge \alpha\}; \quad Q_{\alpha i} = \{q_i \in Q_{oi}: \ \mu_i(q_i) \ge \alpha\}, \ \alpha \in (0;1] \qquad (25)$$

whereas relation (22) implies[17]:

$$Q_{\alpha i} = F_{ij}(Z_{\alpha j}) \tag{26}$$

As long as we consider single input and outcome variables, the numerical determination of the function $\mu_i(q_i)$ in terms of its level sets $Q_{\alpha i}$ is relatively simple: we just have to choose several values of α and to determine the upper bounds $q_{iup\alpha}$ and the lower bounds $q_{ilo\alpha}$ of these sets:

$$q_{ilo\alpha} = \min_{z_j \in Z_{\alpha j}} F_{ij}(z_j); \quad q_{iup\alpha} = \max_{z_j \in Z_{\alpha j}} F_{ij}(z_j) \tag{27}$$

Here the required optimization might be also global, but easier - performed on connected sets, see Fig. 6.

Consider now the case of a vector $\mathbf{z}$ composed of many fuzzy parameters z_j, $j = 1,\ldots,k$, with given membership functions $\mu_j(z_j)$ of bounded interval support Z_{oj}, $\mathbf{z} \in Z_o = \prod_{j=1}^{k} Z_{oj}$ and denote by $\mathbf{q} = F(\mathbf{z})$ the model that defines a vector of outcomes $\mathbf{q} \in \mathbb{R}^p$ of these parameters. We must be clear what is the (fuzzy) logical relation between these parameters: the relation *'and'* would mean that, for a given vector $\mathbf{z}$ of parameter values, we attach to it (and to its outcome $\mathbf{q}$) the lowest degree of possibility attached to any of the components z_j, while the relation *'or'* would mean the highest degree of possibility. Usually, if we investigate the impacts of uncertainty of many parameters, we are interested in the relation *'and'*[18].

Together with membership functions $\mu_j(z_j)$, their α-level sets $Z_{\alpha j}$ are given. In the case of relation *'and'*, it is sufficient to take a common α for all j[19]; thus, we might consider $Z_\alpha = \prod_{j=1}^{k} Z_{\alpha j}$. For the outcome vector $\mathbf{q}$, however, we must define and compute α-level sets in the p-dimensional outcome space defined by a common membership function $\mu(\mathbf{q})$, $Q_\alpha = \{\mathbf{q} \in \mathbb{R}^p: \mu(\mathbf{q}) \geq \alpha\}$. Again, the simple formula is valid:

- - - - - - - - - - - - -

[17] Actually, relation (26) can be also derived directly from the definitions of fuzzy sets - see Zadeh, (1978) - and can be therefore considered more basic as (22).

[18] There are, however, some exceptions - we might relate some parameters by *'or'* and then the resulting logical value by *'and'* to other parameters. In such a case, the processing of membership functions becomes slightly more but not essentially complicated.

[19] Since the lowest α_j between all j is meaningful, a common α results in the broadest α-level sets.

$$Q_\alpha = F(Z_\alpha) \tag{28}$$

but the computation of Q_α is more complicated than in the case of a single outcome and is closely connected to multi-objective optimization.

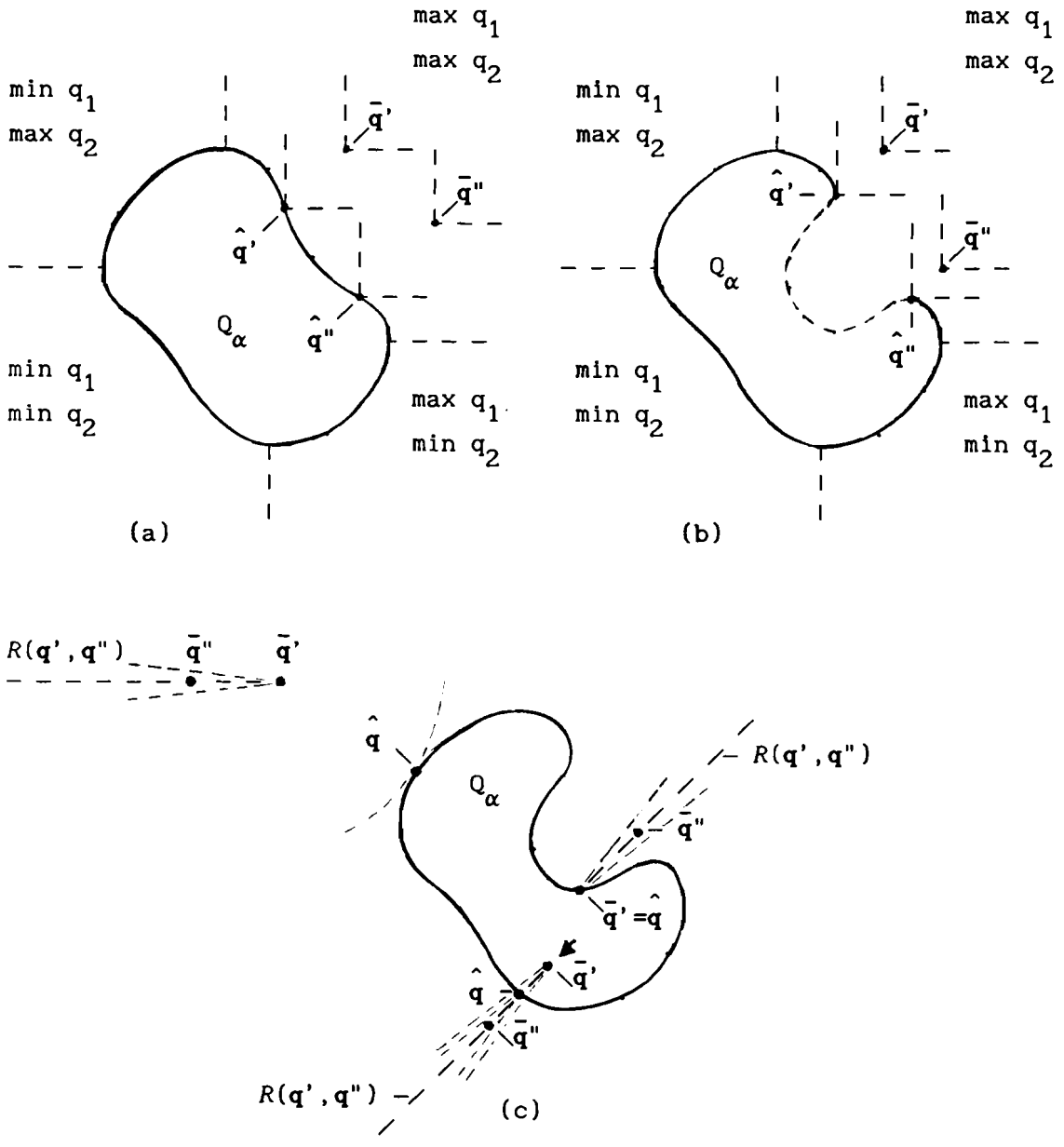


Fig. 7. Analyzing the boundary of α-level sets: (a, b) with the help of the multi-objective reference point optimization approach; (c) with the help of an elastic electronic pencil.

Consider the examples with two outcomes q_1, q_2 ($p = 2$) represented in Fig. 7. If the set Q_α is not too "strongly non-convex" (strictly speaking, if its various Pareto frontiers are connected) as in Fig. 7(a), we can compute any point on its boundary by an multi-objective approach. To do

this, we first assume that both q_1 and q_2 are maximized, then consider one of them minimized, then both minimized, etc. The achievement scalarizing function (13) with suitably distributed reference points $\bar{q}'$, $\bar{q}''$, ..., might be thus used for the purpose of computing a representative sequence $\hat{q}'$, $\hat{q}''$, ..., of points on the boundary of Q_α. However, if Q_α is "strongly nonconvex" (if its Pareto frontiers are not connected) as in Fig. 7(b), such an approach would leave "gaps" in the computational approximation of the boundary of Q_α.

The problem of determining boundaries of sets defined as in (28) is rather frequent in various modeling tasks (e.g. when determining the sets of attainability of dynamic systems), not only when processing fuzzy membership functions. Therefore, it is useful to design a general tool for such purposes; some basic concepts related to the achievement scalarizing function (13) can be modified in order to construct such a tool.

As a result of such modification, we can propose here the following **electronic pencil function** - which might be used as an elastic pencil to draw around the boundary of a set defined as in (28). The pencil can be defined by two reference points $\bar{q}'$, $\bar{q}'' \in \mathbb{R}^p$, where $\bar{q}'$ is interpreted as the sharp point of the pencil while $\bar{q}''$ as its other end. The position of the pencil in the space defines its axis; we need rather a well defined **ray of pencil** emanating from the point $\bar{q}'$ in the direction of $\bar{q}''$:

$$R(\bar{q}',\bar{q}'') = \{q\in\mathbb{R}^p:\ q = \bar{q}' + \gamma(\bar{q}'' - \bar{q}'),\ \gamma \geq 0\} \tag{29}$$

For any point $q \in \mathbb{R}^p$, we can compute its projection on the ray of pencil and the value $\hat{\gamma}(q,\bar{q}',\bar{q}'')$ that defines the position of this projection on the ray. Together with the projection, we compute also a normalized distance $\delta(q,\bar{q}',\bar{q}'')$ of the point q from the ray:

$$\hat{\gamma}(q,\bar{q}',\bar{q}'') = \max\left(0,\ \frac{(\bar{q}'' - \bar{q}')*(q - \bar{q}')}{\|\bar{q}'' - \bar{q}'\|^2}\right); \tag{30}$$

$$P(\bar{q}',\bar{q}'') = \frac{(\bar{q}'' - \bar{q}')(\bar{q}'' - \bar{q}')*}{\|\bar{q}'' - \bar{q}'\|^2}$$

$$\delta(q,\bar{q}',\bar{q}'') = \begin{cases} \dfrac{\|(I-P(\bar{q}',\bar{q}''))(q - \bar{q}')\|}{\|\bar{q}'' - \bar{q}'\|}, & \text{if } \hat{\gamma}(q,\bar{q}',\bar{q}'') > 0, \\ \dfrac{\|q - \bar{q}'\|}{\|\bar{q}'' - \bar{q}'\|} & \text{otherwise,} \end{cases}$$

where the Euclidean norm is used, the numerator of $\hat{\gamma}(q,\bar{q}',\bar{q}'')$ is a scalar product, $P(\bar{q}',\bar{q}'')$ is the projection matrix and its numerator - an outer

product (a matrix). With the help of these expressions, the electronic pencil function can be defined as:

$$\phi(q,\bar{q}',\bar{q}'') = \delta(q,\bar{q}',\bar{q}'') - \varepsilon\hat{\gamma}(q,\bar{q}',\bar{q}'') \tag{31}$$

where $\varepsilon > 0$ is a small coefficient[20], similarly as in (13). This function is minimized over $q \in Q_\alpha$ (or rather over $q = F(q)$, $q \in Z_\alpha$). It is easy to show that if Q_α is bounded, then the minimal points of the electronic pencil function are at the boundary of Q_α. Thus, even if the pencil is arbitrarily pointed and far away from Q_α, see Fig. 7(c), some point on this boundary will be found; if the pencil is close to the boundary and pointed in an appropriate way, the (local) minimum of the function (31) closest to the pencil point can trace even highly nonconvex parts of the boundary. Moreover, the pencil is elastic: if its point is put inside the set Q_α, then the minimal value of (31) becomes negative and a minimal point $\hat{q}$ of (31), on the boundary of Q_α, belongs also to the pencil ray. Thus, an appropriate software can easily "pop out" the pencil of Q_α by automatically shifting the point $\bar{q}'$ to $\hat{q}$. The name *electronic pencil* and its properties are best illustrated by the level sets of this function, see Fig. 8.

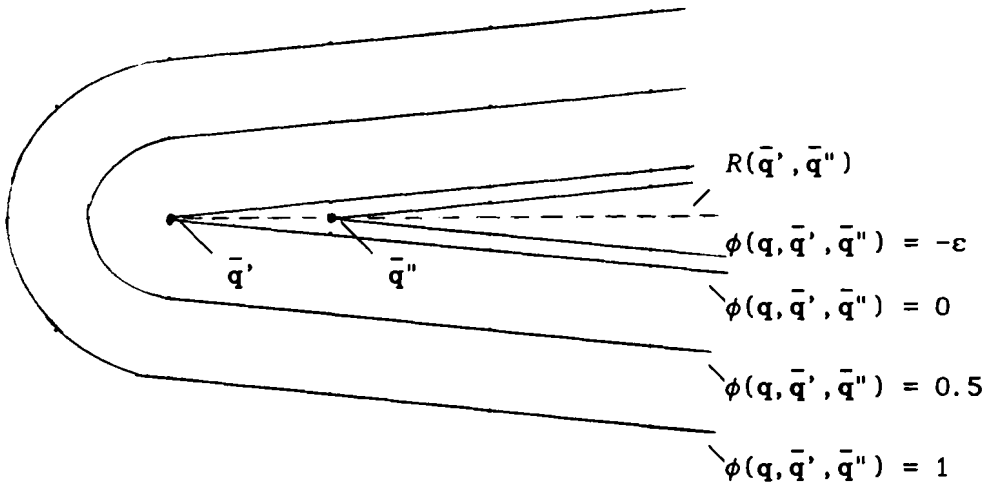


Fig. 8. Level sets of the electronic pencil function.

The electronic pencil should be used interactively, with a graphic interface and a user-friendly way of changing $\bar{q}'$, $\bar{q}''$ (which might present some challenges for $p \geq 3$).

- - - - - - - - - - - - -

[20] This is a nondifferentiable version of the electronic pencil (sharp at its point). If necessary for computational simplicity, a differentiable version can be also defined.

7. Conclusions.

As we have shown, there are enough methodological advancements in multi-objective decision support, modeling and simulation to combine with the existing computer technology for more flexible multi-objective modeling including inverse or constrained simulation, scenario generation by reference point optimization, symbolic sensitivity analysis support, and fuzzy set processing. New tools of more flexible model analysis, such as an interactive electronic pencil, can be also developed. The information about such flexible modeling tools might be important for specialists in various substantive fields.

Appendix: Derivation of the dual sensitivity model.

The equations of the structural sensitivity model from Fig. 5 can be written in an aggregated matrix form for t = 1, ..., T:

$$\Delta y[t] = \frac{\delta h}{\delta w};_t \Delta w[t-1] + \frac{\delta h}{\delta x};_t \Delta x[t] + \frac{\delta h}{\delta z};_t \Delta z[t] + \frac{\delta h}{\delta y};_t \Delta y[t] \qquad (A1)$$

$$\Delta w[t] = I_{wy} \Delta y[t], \quad \Delta q[t] = I_{qy} \Delta y[t]; \quad \Delta w[0] \text{ - given}$$

where $\frac{\delta h}{\delta x};_t = \frac{\delta h}{\delta x}(w[t-1], x[t], z[t], y[t], t)$ etc. The matrix $\frac{\delta h}{\delta y};_t$ is composed of rows $\frac{\delta h_i}{\delta y};_t$ and has non-zero elements only under its diagonal; hence, the matrix $I - \frac{\delta h}{\delta y};_t$ is invertible. The standard state equation form of the model is - with $\Delta w[0] = \Delta w_o$ given:

$$\Delta w[t] = I_{wy}(I - \frac{\delta h}{\delta y};_t)^{-1}(\frac{\delta h}{\delta w};_t \Delta w[t-1] + \frac{\delta h}{\delta x};_t \Delta x[t] + \frac{\delta h}{\delta z};_t \Delta z[t]) \qquad (A2)$$

$$\Delta y[t] = (I - \frac{\delta h}{\delta y};_t)^{-1}(\frac{\delta h}{\delta w};_t \Delta w[t-1] + \frac{\delta h}{\delta x};_t \Delta x[t] + \frac{\delta h}{\delta z};_t \Delta z[t])$$

$$\Delta q[t] = I_{qy}(I - \frac{\delta h}{\delta y};_t)^{-1}(\frac{\delta h}{\delta w};_t \Delta w[t-1] + \frac{\delta h}{\delta x};_t \Delta x[t] + \frac{\delta h}{\delta z};_t \Delta z[t])$$

We would like to determine the gradient of a selected outcome component $q_i[\tau]$ at a selected time $\tau = 1, \ldots, T$ with respect to entire trajectories of x and z (all their components at all time instants). Let the selection operator for this component be I_{qiy}; additionally, denote:

$$\tilde{f}(\Delta w[t-1], \Delta x[t], \Delta z[t], t) =$$

$$= I_{wy}(I - \frac{\delta h}{\delta y};_t)^{-1}(\frac{\delta h}{\delta w};_t \Delta w[t-1] + \frac{\delta h}{\delta x};_t \Delta x[t] + \frac{\delta h}{\delta z};_t \Delta z[t]) \qquad (A3)$$

$$\tilde{g}(\Delta w[\tau-1], \Delta x[\tau], \Delta z[\tau], \tau) =$$

$$= I_{qiy}\left(I - \frac{\delta h}{\delta y}_{;\tau}\right)^{-1}\left(\frac{\delta h}{\delta w}_{;\tau}\Delta w[\tau-1] + \frac{\delta h}{\delta x}_{;\tau}\Delta x[\tau] + \frac{\delta h}{\delta z}_{;\tau}\Delta z[\tau]\right) \qquad (A4)$$

For the problem of determining the gradients of $\Delta q_i[\tau] = J_{i\tau}(\Delta x, \Delta z)$, where Δx, Δz are the appropriate input trajectories and the dependence on other trajectories Δw, Δy is reduced by taking into account the sensitivity model, we can use the Lagrangian function (see e.g. Wierzbicki 1984):

$$J_{i\tau}(\Delta x, \Delta z) = L(\eta, \Delta w, \Delta x, \Delta z) = \tilde{g}(\Delta w[\tau-1], \Delta x[\tau], \Delta z[\tau], \tau) +$$

$$+ \eta_o^*(\Delta w[0] - \Delta w_o) + \sum_{t=1}^{\tau-1} \eta^*[t](\Delta w[t] - \tilde{f}(\Delta w[t-1], \Delta x[t], \Delta z[t], t)) \qquad (A5)$$

where η_o, $\eta[t]$ are (vectors of) Lagrange multipliers and * denotes the transpose of a vector or a matrix. By shifting summation indices for $\eta^*[t]\Delta w[t]$ and assuming $\eta[0] = \eta_o$ we obtain:

$$L(\eta, \Delta w, \Delta x, \Delta z) = \tilde{g}(\Delta w[\tau-1], \Delta x[\tau], \Delta z[\tau], \tau) + \eta^*[\tau-1]\Delta w[\tau-1] - \eta_o^*\Delta w_o +$$

$$+ \sum_{t=1}^{\tau-1} \eta^*[t-1]\Delta w[t-1] - \eta^*[t]\tilde{f}(\Delta w[t-1], \Delta x[t], \Delta z[t], t)) \qquad (A6)$$

Since the functions $\tilde{g}$, $\tilde{f}$ are linear, we can select $\eta[t]$ in such a way that this expression will become independent of $\Delta w[t-1]$; for this purpose, $\eta[t]$ must satisfy the following adjoint equation:

$$\eta[\tau-1] = -\frac{\delta \tilde{g}}{\delta w^*}_{;\tau}; \quad \eta[t-1] = \left(\frac{\delta \tilde{f}}{\delta w}_{;t}\right)^*\eta[t] \text{ for } t = \tau-1, \tau-2, \ldots 1 \qquad (A7)$$

If this equation is satisfied, the derivatives of $L(\eta, \Delta w, \Delta x, \Delta z)$ are non-zero only with respect to the components of Δx and Δz (the derivative with respect to η is zero, if the state equations (A2) is satisfied). These non-zero derivatives are identical with those of the function $J_{i\tau}(\Delta x, \Delta z)$:

$$\frac{\delta J_{i\tau}}{\delta x^*[t]} = -\left(\frac{\delta \tilde{f}}{\delta x}_{;t}\right)^*\eta[t], \quad t = 1, \ldots \tau-1; \quad \frac{\delta J_{i\tau}}{\delta x^*[\tau]} = \frac{\delta \tilde{g}}{\delta x^*}_{;\tau} \qquad (A8)$$

$$\frac{\delta J_{i\tau}}{\delta z^*[t]} = -\left(\frac{\delta \tilde{f}}{\delta z}_{;t}\right)^*\eta[t], \quad t = 1, \ldots \tau-1; \quad \frac{\delta J_{i\tau}}{\delta z^*[\tau]} = \frac{\delta \tilde{g}}{\delta z^*}_{;\tau}$$

The last equation applies if all components $z[t]$ are independent; if we add the requirement that $\Delta z[t] = \Delta z$ represents parameters that are constant in time, then this equation modifies to:

$$\frac{\delta J_{i\tau}}{\delta z^*} = \frac{\delta \tilde{g}}{\delta z^*}_{;\tau} - \sum_{t=1}^{\tau-1}\left(\frac{\delta \tilde{f}}{\delta z}_{;t}\right)^*\eta[t] \qquad (A9)$$

Note that the derivatives of relations in the basic model from Fig. 3 are the same as the derivatives of relations in the primal sensitivity model from Fig. 5. It remains to take into account the matrices that according to (A3,4) define the derivatives of functions $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{g}}$. Thus, **the dual sensitivity model** - the full equations defining the gradients of a selected outcome component $q_i[\tau] = J_{i\tau}(\mathbf{x},\mathbf{z})$ with respect to decision and parameter components $\mathbf{x}[t]$, $\mathbf{z}[t]$ or to constant parameters z - is as follows:

Adjoint equations:

$$\eta[\tau-1] = -\left(\frac{\delta h}{\delta w;\tau}\right)^*\left(I - \frac{\delta h}{\delta y;\tau}\right)^{*-1} I^*_{qiy};$$

$$\eta[t-1] = \left(\frac{\delta h}{\delta w;t}\right)^*\left(I - \frac{\delta h}{\delta y;t}\right)^{*-1} I^*_{wy}\eta[t] \text{ for } t = \tau-1,\ \tau-2,..1 \qquad \text{(A10)}$$

Gradient equations:

$$\frac{\delta J_{i\tau}}{\delta x^*[t]} = -\left(\frac{\delta h}{\delta x;t}\right)^*\left(I - \frac{\delta h}{\delta y;t}\right)^{*-1} I^*_{wy}\eta[t],\ t = 1,\ldots\tau-1;$$

$$\frac{\delta J_{i\tau}}{\delta x^*[\tau]} = \left(\frac{\delta h}{\delta x;\tau}\right)^*\left(I - \frac{\delta h}{\delta y;\tau}\right)^{*-1} I^*_{qiy},$$

$$\frac{\delta J_{i\tau}}{\delta z^*[t]} = -\left(\frac{\delta h}{\delta z;t}\right)^*\left(I - \frac{\delta h}{\delta y;t}\right)^{*-1} I^*_{wy}\eta[t],\ t = 1,\ldots\tau-1;$$

$$\frac{\delta J_{i\tau}}{\delta z^*[\tau]} = \left(\frac{\delta h}{\delta z;\tau}\right)^*\left(I - \frac{\delta h}{\delta y;\tau}\right)^{*-1} I^*_{qiy} \qquad \text{(A11)}$$

Gradient equation for constant parameters:

$$\frac{\delta J_{i\tau}}{\delta z^*} = \left(\frac{\delta h}{\delta z;\tau}\right)^*\left(I - \frac{\delta h}{\delta y;\tau}\right)^{*-1} I^*_{qiy} - \sum_{t=1}^{\tau-1}\left(\frac{\delta h}{\delta z;t}\right)^*\left(I - \frac{\delta h}{\delta y;t}\right)^{*-1} I^*_{wy}\eta[t] \qquad \text{(A12)}$$

Additionally, we obtain $\frac{\delta J_{i\tau}}{\delta w^*[0]} = -\eta[0]$ as a consequence of (A6). For the case of a model with multiple delays from Fig. 4, the derivation of adjoint and gradient equations can proceed similarly, but for each additional time-retarded term depending on $w[t-\vartheta]$ in state equations we must add a corresponding additional time-advanced term depending on $\eta[t+\vartheta]$ in adjoint equations, and for each dependence on a retarded decision $\mathbf{x}[t-\vartheta]$ we must add a corresponding time-advanced term depending on $\eta[t+\vartheta]$ in gradient equations (see Wierzbicki, 1984, where the adjoint equations for continuous-time difference-differential equations are discussed). The inclusion of time-advanced terms in adjoint equations stresses only the fact that they must be solved (also for other reasons, e.g. their numerical stability) in the reverse direction of time, i.e. starting with $t = T-1$ or $t = \tau-1$ and counting time "down to zero".

References.

Andriole, S.J. (1989): Handbook of Decision Support Systems. TAB Professional and Reference Books, Blue Ridge Summit, PA.

Cooke, S. and N. Slack (1984): Making Management Decisions. Prentice-Hall, Englewood Cliffs.

Jaszkiewicz, A. (1992): A Prototype Version of a Decision Support System for Planning Municipal Waste Water Treatment in a River Basin. IIASA Working Paper, to appear.

Kacprzyk J. and M. Fredrizzi (eds.) (1988): Combining Fuzzy Imprecision with Probabilistic Uncertainty in Decision Making. Lecture Notes in Economics and Mathematical Systems Vol. 310, Springer-Verlag, Berlin - Heidelberg.

Kallio, M., A. Lewandowski and W. Orchard-Hays (1980): An implementation of the reference point approach for multi-objective optimization. WP-80-35, IIASA, Laxenburg 1980.

Kręglewski, T., J. Paczyński, J. Granat and A.P. Wierzbicki (1989): IAC-DIDAS-N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models. In A. Lewandowski and A.P. Wierzbicki, eds.: Aspiration Based Decision Support Systems. Lecture Notes in Economics and Mathematical Systems Vol. 331, pp. 378-381, Springer Verlag, Berlin - Heidelberg.

Lewandowski, A., T. Kręglewski, T. Rogowski and A.P. Wierzbicki (1989): Decision Support Systems of DIDAS Family. In A. Lewandowski and A.P. Wierzbicki, eds.: Aspiration Based Decision Support Systems. Lecture Notes in Economics and Mathematical Systems Vol. 331, pp. 21-47, Springer-Verlag, Berlin - Heidelberg.

Makowski, M. and J. Sosnowski (1989): HYBRID - a Mathematical Programming Package. In A. Lewandowski and A.P. Wierzbicki, eds.: Aspiration Based Decision Support Systems. Lecture Notes in Economics and Mathematical Systems Vol. 331, Springer Verlag, Berlin - Heidelberg.

Pawlak, Z. (1992): Rough Sets. Some Aspects of Reasoning about Knowledge. Kluwer Academic Publishers, Dordrecht (to appear).

Popper, K.R. (1983): Realism and the Aim of Science. Hutchinson, London.

Simon, H.A. (1957): Models of Man. Macmillan, New York.

Wessels, J. and A.P. Wierzbicki, eds. (1982): User-Oriented Methodology and Techniques of Decision Analysis and Support. Lecture Notes in Economic and Mathematical Systems, Springer-Verlag, forthcoming.

Wierzbicki, A.P. (1980): The use of reference objectives in multiobjective optimization. In G. Fandel, T. Gal (eds.): Multiple Criteria Decision Making; Theory and Applications, Lecture Notes in Economic and Mathematical Systems Vol. 177, pp. 468-486, Springer-Verlag, Berlin - Heidelberg.

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, Vol. 8 (1986) pp. 73-87.

Wierzbicki, A.P. (1992): The Role of Intuition and Creativity in Decision Making. IIASA Working Paper, forthcoming.

Zadeh, L.A. (1978): Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems Vol. 1, pp. 3-28.