Working Paper

Multicriteria Optimization in Water Resources Distribution Problem

G.G. Kotkin
D.V. Mironov

WP-90-47
September 1990

IIASA

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

Telephone: (0 22 36) 715 21 * 0 □ Telex: 079137 iiasa a □ Telefax: (0 22 36) 71313

Multicriteria Optimization in Water Resources Distribution Problem

G.G. Kotkin
D.V. Mironov

WP-90-47
September 1990

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

Telephone: (0 22 36) 715 21 * 0 □ Telex: 079 137 iiasa a □ Telefax: (0 22 36) 71313

FOREWORD

A multicriteria problem on the distribution of water resources between four reservoirs with respect to electricity production in three countries is considered. The reservoirs are connected in a unique water system. The water inflow to one of the reservoirs is considered as the result of water releases from the others at previous time. The problem is to choose water releases which would provide maximum of the electricity in each country and minimum of fluctuations of the water levels in reservoirs as well. The multicriteria non-linear programming system DISO/PC - MCNLP is used for the solution of the problem.

Prof. A.B. Kurzhanski
Chairman,
System and Decision Sciences Program

MULTICRITERIA OPTIMIZATION IN WATER RESOURCES DISTRIBUTION PROBLEM

G.G. Kotkin and D.V. Mironov

1. Principal task definition, the case of the Zambesi River

Zambezi river system is the water system composed of four linked reservoirs with three of them having electro-power stations. Kariba, which situated on the border between Zambia and Zimbabwe, Cabora Bassa in Mozambique and Kafue - Itezhitezhi hydro-power system in Zambia are three major hydroelectric facilities in Zambezi basin. The principal scheme of Zambezi river system is shown in Fig. 1.

The problem is to find out a strategy of water releases (called outflows) from each of the reservoirs which would provide maximum of power production in each of three hydro-power plants (or minimum of difference between power production and power demand) and avoid large fluctuations of the water level in each of the reservoirs as well (see Salevicz et al., 1989).

If one assumes time discretization equal to a month then the control variables are volumes of monthly outflows from four reservoirs (denoted by $\text{Outflow}_i(t)$ for $i=1,2,3,4$; $t=1,2,...,12$). Monthly energy production in each of the hydro-power plants can be presented as a product of monthly outflow and difference between water level in a reservoir having power plant and water level in a river downstream the plant denoted by $\text{Head}_i(t)$ (see Fig. 2):

$$\text{Energy}_i(t) = \text{Head}_i(t) * \text{Outflow}(t), \tag{1}$$

where $\text{Energy}(t)$ is the energy production in i-th hydro-power plant during t-th month (October is assumed to be the first one), $i=1,2,3$. Naturally $\text{Energy}_i(t)$ is not an absolute quantity of energy produced in the i-th hydro-power plant during t-th month and, consequently, it must be expressed in relative units.

All reservoirs in the Zambezi river system are linked. In the first approximation the inflow to one of them in several month (denoted by Inflow_i) can be considered as the outflow from the other in previous month. For example:

$$\text{Inflow}_3(\text{in September}) = \text{Outflow}_4(\text{in August}).$$

This formula is true for the case when the whole amount of water released from i-th reservoir comes up to $(i+1)$-th reservoir exactly in a month. Nevertheless if time discretization is equal to a month one can hope that the above supposition will not bring to significant mistakes in real situation, at least these mistakes should unlikely be more significant than that stipulated by defects of the others components of the model and

inaccuracy of the experimental data. Moreover, if inflow is expressed through outflow then the total amount of variables in our model (and consequently the time of calculation) can be decreased which is very important from the practical point of view.

Taking into account the structure of the water system (Fig. 1) we have the following expressions for inflows to the 1-st and 3-rd reservoirs:

$$\text{Inflow}_3(t) = \text{Outflow}_4(t-1),$$

$$\text{Inflow}_1(t) = \text{Outflow}_2(t-1) + \text{Outflow}_3(t-1), \tag{2}$$

where t is month number.

The inflows to the 2-nd and 4-th reservoirs are assumed to be given values for each month.

The expression for the $\text{Head}_i(t)$ in formula (1) can be written as follows:

$$\text{Head}_i(t) = \text{Level\_in\_reservoir}_i(t) - \text{Level\_in\_river}_i, \tag{3}$$

where the level in river (denoted by Level_in_river_i) is assumed to be constant, $i = 1,2,3$. It is quite understandable that the level in reservoir depends on volume of water and this dependence is defined by the shape of the reservoir bottom (see Fig. 3).

The volume of water in each reservoir in each month can be calculated after the following discrete-time formula:

$$\text{Volume}_i(t) = \text{Volume}_i(t-1) + \text{Inflow}_i(t) - \text{Outflow}_i(t) - \text{Evaporation}_i(t) \tag{4}$$

where the last term in the right-hand part of equation (4) (denoted for simplicity by $\text{Evaporation}_i(t)$) describes monthly averaged difference between evaporation and precipitation (negative when precipitation prevails upon evaporation and positive in the opposite case) for i-th reservoir and has to be expressed through the volume of water, $i = 1,...,4$.

It should be noticed that we use the discrete form of dynamic system model. If dynamic system model is used the equation (4) has to be written as follows:

$$\text{Volume}_i(t) = \text{Volume}_i(0) + \int_0^t \frac{d}{d\tau} [\text{Inflow}_i(t) - \text{Outflow}_i(\tau) - \text{Evaporation}_i(\tau)]\, d\tau \tag{4b}$$

where $\text{Volume}_i(0)$ is the volume of water at the initial moment of time $t=0$, $i=1,...,4$.

Monthly averaged differences between evaporation and precipitation depend on the area of reservoir, temperature of water surface, temperature and humidity of the near surface layer of the atmosphere, wind speed, cloudiness and many other factors:

$$\text{Evaporation}_i(t) = \text{Area}_i(t) * f_i(\text{Temperature}_i(t), \text{Wind speed}_i(t),...) \quad (5)$$

We will use the simplest approximation for functions $f_i(...)$:

$$f_i(...) = Q_i(t), \quad (6)$$

where $Q_i(t)$ are piecewise constant functions of time t only which characterize monthly averaged differences between evaporation and precipitation, $i = 1,...,4$. The multiyear averaged $Q_i(t)$ values obtained by experimental data are presented in Appendix 1 (these data have been used for the calculations which results are given in Appendix 4).

It is easy to show that the area of the water surface of the reservoir can be expressed as the first derivative of Volume with respect to Level:

$$\text{Area}\,(\,\text{Volume}\,) = d\,(\text{Volume})\,/\,d\,(\text{Level}) = \frac{d\,\text{Volume}}{d\,\text{Level}} \quad (7)$$

Let us assume that Volume vs Area and Volume vs Level relationships are known for each reservoir. In this case the energy production can be calculated for each reservoir after the formulae (1) and (4). The values of outflows must provide the minimum of differences between power productions and power demands (denoted by $\text{Demand}_i(t)$, $i=1,2,3$) in each reservoir in each month:

$$\begin{aligned} &\text{Demand}_1(t) - \text{Energy}_1(t) \rightarrow \min, \\ &\text{Demand}_2(t) - \text{Energy}_2(t) \rightarrow \min, \\ &\text{Demand}_3(t) - \text{Energy}_3(t) \rightarrow \min, \end{aligned} \quad (8)$$

where $\text{Demand}_i(t)$ are expressed in the same units as it was for $\text{Energy}_i(t)$.

At the same time the large fluctuations of water level in each reservoir have to be avoided so the second group of the criteria can be written in the form

$$
\begin{aligned}
&| \text{Level}_1(t) - \text{Reference\_Lev}_1 | \rightarrow \min, \\
&| \text{Level}_2(t) - \text{Reference\_Lev}_2 | \rightarrow \min, \\
&| \text{Level}_3(t) - \text{Reference\_Lev}_3 | \rightarrow \min, \\
&| \text{Level}_4(t) - \text{Reference\_Lev}_4 | \rightarrow \min,
\end{aligned}
\tag{9}
$$

where Reference_Lev_i denotes reference (desired) level of water in i-th reservoir, $i = 1,...,4$.

Criteria (8) and (9) together with formulae (1) - (7) allow for calculation of the optimal water releases from each of the reservoirs, i.e. for solution of the multicriteria optimization problem. Formal description of the optimization problem is discussed in the next paragraph.

2. Formal task definition.

Optimization process is characterized by the following basic features. Mathematical model describing the phenomenon includes control variables $(x^1, x^2,...,x^n)$ or x, parameters to be improved $(f^1, f^2,...,f^m)$ or f, called criteria or goal functions, and some relationships between control variables and criteria. These relationships define principal structure of a model. They can be of the following type:

1) Equalities $(h^1,...,h^S)$ or h - some parameters should be equal to given values;
2) Inequalities $(g^1,...,g^p)$ or g - some parameters should exceed given values;
3) Equalities and inequalities in differential form: relationships between derivatives or integrals with respect to some parameters should take place.. Usually time derivatives are considered. Such a model is called "dynamic model".

The usual multicriteria optimization problem can be expressed as follows:

$$
\begin{aligned}
&f^1(x^1,...,x^n) \rightarrow \min, \\
&f^2(x^1,...,x^n) \rightarrow \min, \\
&\text{.....},
\end{aligned}
$$

$$f^m(x^1,\ldots,x^n) \to \min;$$

subject to

$$g^1(x^1,\ldots,x^n) \le d^1,$$

$$g^2(x^1,\ldots,x^n) \le d^2,$$

$$\ldots\ldots\ldots,$$

$$g^p(x^1,\ldots,x^n) \le d^p;$$

$$h^1(x^1,\ldots,x^n) = c^1,$$

$$h^2(x^2,\ldots,x^n) = c^2,$$

$$\ldots\ldots\ldots,$$

$$h^S(x^1,\ldots,x^n) = c^S;$$

$$a^1 \le x^1 \le b^1,$$

$$a^2 \le x^2 \le b^2,$$

$$\ldots\ldots\ldots,$$

$$a^n \le x^n \le b^n; \qquad (10)$$

where $a^1,\ldots,a^n$; $b^1,\ldots,b^n$; $c^1,\ldots,c^S$; $d^1,\ldots,d^p$ are given values. This problem can be written in more compact form:

$$\min \quad f(x),$$

$$g(x) \le d,$$

$$h(x) = c,$$

$$a \le x \le b, \qquad (10')$$

where f,g,h are convex vector-functions, $x \in R^n, a,b,c,d$ are given vector.

The water resources distribution (w.r.d.) problem may be considered as a dynamic model (see formula (4b)). In order to reduce the time of calculation we use the usual

(discrete time) optimization problem (see formula (4)). In this case we may take into account some specific features of the problem.

It is well known that any dynamic optimization problem may be reduced to usual (static) optimization problem (Evtushenko, 1985). We have used some kind of reduction. All formulae of previous paragraph have been written in discrete form. We have assumed that the time is discrete : $t = 1,2,...,12$. In order to finish this reduction we have to exclude the time t from the formulae in Section 1.

The optimization task variables are the model parameters which we have to choose by the optimization process. The outflows are the variables of w.r.d. problem. Let us denote them by out:

$$\begin{aligned} \text{out} &= (\text{out}^1,...,\text{out}^{48}) = \\ &= \{\text{Outflow}_1(1), \text{Outflow}_1(2),..., \text{Outflow}_1(12), \\ &\quad \text{Outflow}_2(1),...,\text{Outflow}_2(12), \\ &\quad ..., \text{Outflow}_4(12)\}. \end{aligned} \tag{11}$$

We do not change the notation and will use below the time t like superscript.

We need to obtain nonrecursive formulae for volumes (see (4)) to write all the functions of the optimization problem. We will suggest that the values of volume functions $\text{Volume}_{it}(\text{out})$ are the values for the last day of the month t ($i = 1,2,3,4$; $t = 1,2,...,12$).

To simplify the notation we denote $\text{Volume}_{i0}(\text{out}) = \text{Ini}_i$, where Ini_i is volume of water in i-th reservoir at the initial moment of time $t = 1$, $i = 1,...,4$.

To form an optimization problem we have to add the analytical functions describing Level vs Volume and Area vs Volume relationships to the formulae (1)-(9). Experimental data about these relationships are available so we have to solve so-called identification task of determination of function type and function parameters by given data. We have chosen the following polynomial functions:

$$\begin{aligned} \text{Level}_{it} &= h1_i * [\text{Volume}_{it}(\text{out})]^2 + h2_i * \text{Volume}_{it}(\text{out}) + h3_i, \\ \text{Area}_{it} &= n1_i * [\text{Volume}_{it}(\text{out})]^2 + n2_i * \text{Volume}_{it}(\text{out}) + n3_i \end{aligned} \tag{12}$$

where $h1_i$, $h2_i$, $h3_i$, $n1_i$, $n2_i$, $n3$ are given coefficients determined by experimental data, $i = 1,...,4$, $t = 1,...,12$ (we took circumstance into consideration that our model is discrete and have used Volume_{it-1} (out) instead of Volume_{it} (out) to simplify the model).

It is easy to show using (4) that the following formula is true:

$$\text{Volume}_{it}\,(\text{out}) = \text{Ini}_i + \sum_{k=1}^{t} [\text{Inflow}_{ik}\,(\text{out}) - \text{out} - \tag{13}$$

$$\text{Evaporation}_{ik}\,(\text{out})].$$

Here Ini_i are the initial amounts of water in each of the reservoirs (given values), Inflow_{ik} (out) are the functions with respect to outflows and given values of inflows (denoted by Inf_{11}, Inf_{2t}, Inf_{31}, Inf_{4t}):

$$\text{Inflow}_{it}\,(\text{out}) = \text{Inf}_{11}, \quad \text{if } t = 1,$$

$$\text{Inflow}_{it}\,(\text{out}) = \text{out}_{2\,t-1} + \text{out}_{3\,t-1}, \quad \text{if } t > 1,$$

$$\text{Inflow}_{2t}\,(\text{out}) = \text{Inf}_{2t},$$

$$\text{Inflow}_{3t}\,(\text{out}) = \text{Inf}_{31}, \quad \text{if } t = 1,$$

$$\text{Inflow}_{3t}\,(\text{out}) = \text{out}_{4\,t-1}, \quad \text{if } t > 1,$$

$$\text{Inflow}_{4t}\,(\text{out}) = \text{Inf}_{4t}, \tag{14}$$

where $t = 1,...,12$ is the month number.

According to (5), (6) and (12) the differences between evaporation and precipitation depend on volume in the following way:

$$\text{Evaporation}_{it}\,[\text{Volume}_{it-1}\,(\text{out})] = Q_{it}\{n1_i * [\text{Volume}_{it-1}\,(\text{out})]^2 + \tag{15}$$

$$+ n2_i * \text{Volume}_{it-1}\,(\text{out}) + n3_i\},$$

where Q_{it} are given values for each reservoir for each month, $i = 1,...,4$, $t = 1,...,12$.

We have the following expressions for volumes:

$$\text{Volume}_{it}\,(\text{out}) = \text{Ini}_i + \sum_{k=1}^{t} \{\text{Inflow}_{ik}\,(\text{out}) - \text{out}_{ik} - \tag{16}$$

$$- Q_{ik} * [nl_i * (\text{Volume}_{ik-1}(\text{out}))^2 - n2_i * \text{Volume}_{ik-1}(\text{out}) - n3_i]\}$$

where $i = 1,...,4,\ t = 1,...,12$.

For any month number the expression (16) written for the previous time $t-1$ can be inserted in the equation (16) for the time t. So we can obtain 48 equations for $\text{Volume}_{it}(\text{out})$ in each month. For instance, for $t=2$ we have:

$$\begin{aligned}
&\text{Volume}_{i2}(\text{out}) = \text{Ini}_i + \text{Inflow}_{i1}(\text{out}) - \text{out}_{i1} - \\
&- Q_{i1} * (n1_i * \text{Ini}_i^2 + n2_i * \text{Ini}_i + n3_2) + \text{Inflow}_{i2}(\text{out}) - \text{out}_{i2} - \\
&- Q_{i2} * \{n1_i * [\text{Ini}_i + \text{Inflow}_{i1}(\text{out}) - \text{out}_{i1} - Q_{i1} * (n1_i * \text{Ini}_i^2 + \\
&+ n2_i * \text{Ini}_i + n3_i)]^2 + n2_i * [\text{Ini}_i + \text{Inflow}_{i1}(\text{out}) - \text{out}_{i1} - \\
&- Q_{i1} * (n1_i * \text{Ini}_i^2 + n2_i * \text{Ini}_i + n3_i)] + n3_i\},\ i = 1,...,4. \qquad (17)
\end{aligned}$$

It is not necessary to write all these equations because they can be obtained recursively after the following formula:

$$\begin{aligned}
\text{Volume}_{it}(\text{out}) = \text{Volume}_{it-1}(\text{out}) + \text{Inflow}_{it}(\text{out}) - \qquad (18) \\
- \text{out}_{it} - \text{Evaporation}_{it}(\text{out}),
\end{aligned}$$

where $i = 1,...,4,\ t = 1,...,12$.

We have to obtain now the approximations for Energy vs Level relationships. They can be written as follows:

$$\begin{aligned}
&\text{Energy}_{it}(\text{out}) = \text{Head}_{it}(\text{out}) * \text{out}_{it}, \\
&\text{Head}_{it}(\text{out}) = \text{Level}_{it}(\text{out}) - \text{Level_in_river}_i, \qquad (19)
\end{aligned}$$

where $i = 1,2,3,\ t = 1,...,12$; Level_in_river_i values are known and $\text{Level}_{it}(\text{out})$ are expressed after the first one of formulae (12).

Let us denote the functions which take into account only positive values by " ( ) " : $(y) = \max\{y,0\}$, where y is a real number.

Finally the formal task definition can be written as follows:

$$[\text{Demand}_{1t} - \text{Energy}_{1t}\,(\text{out})] \to \min,\ t = 1,...,12,$$

$$[\text{Demand}_{2t} - \text{Energy}_{2t}\,(\text{out})] \to \min,\ t = 1,...,12,$$

$$[\text{Demand}_{3t} - \text{Energy}_{3t}\,(\text{out})] \to \min,\ t = 1,...,12,$$

$$|\ \text{Level}_{1t}\,(\text{out}) - \text{Reference}_1\ | \to \min,\ t = 1,...,12,$$

$$|\ \text{Level}_{2t}\,(\text{out}) - \text{Reference}_2\ | \to \min,\ t = 1,...,12,$$

$$|\ \text{Level}_{3t}\,(\text{out}) - \text{Reference}_3\ | \to \min,\ t = 1,...,12,$$

$$|\ \text{Level}_{4t}\,(\text{out}) - \text{Reference}_4\ | \to \min,\ t = 1,...,12,$$

$$0 < \text{out} < M,$$

$$\begin{aligned}
&\text{Volume}_{1t}\,(\text{out}) \geq 0,\ t = 1,...,12,\\
&\text{Volume}_{2t}\,(\text{out}) \geq 0,\ t = 1,...,12,\\
&\text{Volume}_{3t}\,(\text{out}) \geq 0,\ t = 1,...,12,\\
&\text{Volume}_{4t}\,(\text{out}) \geq 0,\ t = 1,...,12,
\end{aligned} \tag{20}$$

$$\text{Volume}_1\,[t=12]\,(\text{out}) = \text{Ini}_1,$$

$$\text{Volume}_2\,[t=12]\,(\text{out}) = \text{Ini}_2,$$

$$\text{Volume}_3\,[t=12]\,(\text{out}) = \text{Ini}_3,$$

$$\text{Volume}_4\,[t=12]\,(\text{out}) = \text{Ini}_4,$$

where Demand_{jt}, Reference_i, $M = (M^1,...,M^{48})$, Ini_i are given values; Energy_{jt}, Level_{it}, Volume_{it} are given functions (see formulae (11)-(19)); $i = 1,...,4$; $j = 1,2,3$; $t == 1,...,12$.

As it was noticed above Volume_{it} (out) represent the volumes of water in reservoirs at the end of the month so the volumes of water in the beginning of the first month should be equal to the initial amounts of water in reservoirs (Ini_i). The volumes of water in the

last month of the year should be equal to the volumes of water at the first time.

So we have 84 goal functions described by equations (20). It is convenient to use some kind of convolution functions to decrease a total amount of goal functions. For example, if functions $Y(y^1,...,y^{12}) = \max y$ are used we have the following task:

$$\max_t [\text{Demand}_{it} - \text{Energy}_{it}] \to \min, \quad i = 1,2,3,$$

$$\max_t | \text{Level}_{it} - \text{Reference}_i | \to \min, \quad i = 1,...,4, \tag{21}$$

subjected to the same constraints.

We will use the functions $\Psi_p(y^1,...,y^{12}) = \sum_{t=1}^{12} (y^t)^p$ which allow us to change the penalty coefficient p for rate of divergences of y values from zero. Finally we have the following task:

$$\sum_{t=1}^{12} (\text{Demand}_{it} - \text{Energy}_{it})^{P_i} \to \min, \quad i = 1,2,3,$$

$$\sum_{t=1}^{12} (\text{Level}_{it} - \text{Reference}_i)^{P_i} \to \min, \quad i = 1,...,4,$$

subject to

$$\text{Volume}_{it} \geq 0, \quad t = 1,...,12, \quad i = 1,...,4, \tag{22}$$

$$\text{Volume}_{i12} = \text{Ini}_i, \quad i = 1,...,4,$$

$$0 \leq \text{out} \leq M.$$

Mathematical tool and numerical algorithm are described in the next paragraphs.

3. Mathematical tool.

The water resources distribution problem was solved by the dialogue system of multicriteria nonlinear programming DISO/PC-MCNLP. DISO/PC-MCNLP is based on new ideas in multicriteria and nonlinear optimization associated with parametric optimization, sensitivity analysis and inverse optimization problems (see Kotkin, 1988, 1989).

The great variety of numerical algorithms, dialogue procedures, parametric and inverse study possibilities and flexible control are the main features of DISO/PC-MCNLP system. DISO/PC-MCNLP reduces any optimization problem (nonlinear programming or multicriteria optimization) to parametric or inverse optimization problem. Therefore all functions such as goal, constraints or parametric constraints are equal on the basic level of DISO/PC-MCNLP system programm.

Some geometric parametric ideas are the basis of DISO/PC-MCNLP program. The function identification is a property of some environmental level. Some protection is the assignment of next level (see Fig. 4). The flexible control system Field Manager and analytical differentiation language DIFALG build the user level. One can use the "C" language to make a task definition if one prefers "C" to DIFALG.

The unconstrained optimization algorithm, nonlinear programming algorithm, dialogue procedure of multicriteria search, any parameters of numerical method and some parameters of applied task can be changed asynchronically with respect to calculation process. Field Manager allows one to adjust interface to one's own applied task. The selection of numerical algorithm and its parameters is the beginning of possible adjustment. Preparation of windows (if one needs) , choice of forms of presentation of system and task objects (numeric, histograms, pictograms, pictures), defining the applied task objects, names, etc., is the second part of the adjustment. One can change the values of parameters which lie at the basis of one's own applied task, write and read this parameters and other information from disk asynchronically to calculation process. All these features of DISO/PC-MCNLP allow to construct easily the dialogue system for applied optimization task.

Just the same dialogue system have been constructed for water resources distribution problem. The applied dialogue optimization system POOL allows one both to solve the water resources multicriteria task and to define preliminary the model parameters using experimental data as well.

Let us consider some specific features of numerical calculations of the water resources distribution problem. The user-friendly interface POOL system will be considered in the next paragraph. The recursive calculation reflecting the dynamic nature of the problem (see formulae (16) and (18)) is the main feature of water resources distribution problem. The "C" language recursion have not been used in POOL system. In this case the massive for temporary saving of volumes values is needed. The scheme of calculation process is shown in Fig. 5. The text of program is represented in Appendix 2. To provide flexible

control we used "C" language for task definition.

Reading the text of program one should remember that the definition of parameters is the first stage of calculations. Separate optimization task is solved and all coefficients are saved on a disk by DISO/PC-MCNLP system possibilities. The water resources optimization problem takes all of them from disk in the form of the massive of parameters. The POOL system process is shown in Fig. 6.

We will describe briefly the identification task. There are several relationships between y and x values. Usually the available data are presented in the form of the couples (x^1, y^1), (x^2, y^2),...,(x^m, y^m). One has to find the function $f(x)$ satisfying the following conditions:

$$f(x^1) = y^1, f(x^2) = y^2, \ldots, f(x^m) = y^m.$$

Usually the following approximate condition are used:

$$f(x^1) = y + \varepsilon^1,\ f(x^2) = y^2 + \varepsilon^2, \ldots, f(x^m) = y^m + \varepsilon^m,$$

where $\varepsilon^1, \varepsilon^2, \ldots, \varepsilon^3$ are small values.

The identification task is shared into two stages:

1) selection of the type of function (polynomial, exponential, trigonometric, etc.) ,
2) parameters definition.

We chose quadratic polynoms for approximation of Level vs Volume relationships:

$$\text{Level} = f(\text{Volume}) = h1 * (\text{Volume}) + h2 * \text{Volume} + h3.$$

Using the experimental data, $(\text{level}^1, \text{volume})$, $(\text{level}^2, \text{volume}^2)$,...,$(\text{level}^m, \text{volume}^m)$, we have to minimize the values of $\varepsilon^1, \varepsilon^2, \ldots, \varepsilon^m$ in the following formulae picking the $h1, h2, h3$ values : $\text{level}^1 = f(\text{volume}) + \varepsilon^1$, $\text{level}^2 = f(\text{volume}^2) + \varepsilon^2$,..., $\text{level}^m = f(\text{volume}^m) + \varepsilon^m$

Thus we have to solve the following optimization task:

$$| \text{level}^1 - f(\text{volume}^1) | \rightarrow \min,$$

$$| \text{level}^2 - f(\text{volume}^2) | \rightarrow \min,$$

$$\ldots\ldots\ldots\ldots,$$

$$| \text{level}^m - f(\text{volume}^m) | \rightarrow \min.$$

Usually such a problem is reduced to unconstrained optimization problem solved using the least square solution method:

$$[\text{Level}^i - f(\text{Volume}^i)]^2 \rightarrow \min. \qquad (23)$$

The same procedure has been carried out in the case of Area vs Volume relationships (see Appendix 2).

4. Interface.

The interface possibilities of DISO/PC-MCNLP (Mazouric, 1988) and POOL system are provided by Field Manager system. Field Manager allows to construct a number of windows and to receive any information in these windows. The main idea of F.M.s. is as follows. We may look out and correct a set of data which have been marked in our program. There are the following kind of data to be marked (called objects): scalars, vectors, matrixes, functions with respect to real, integer, character or string values. Running the program we can select the following form of presentations of the objects: numeric, histograms, pictograms, pictures. We can also link some reaction with special user actions. For example, control menu is just the vector of string with some reaction linked with pushing the "enter" key (e.g., running some process).

Writing the program we do not need to think about it. We have only to mark objects we are interested in and to decide how often do we have to redraw them on the display.

When the program has been run we may construct a number of windows with several objects in them in some form of presentation with some names (labels). We may prepare in this way some standard windows and create additional windows whenever we wish.

The set of DISO/PC-MCNLP system objects is fixed. These objects allow to select various numerical algorithms and dialogue multicriteria procedures, change theirs parameters, run and stop processes and correct some task definition parameters. We may in-

spect and correct the following task definition parameters:

1) functions type (goal, inequality or equality constraint),
2) right sides of inequality or equality constraints,
3) lower and upper bounds of parrallelepiped constraints,
4) vector of parameters introduced by user if one wish to correct the applied task parameters during calculations.

Therefore, there are not only universal decision making procedures in DISO/PC-MCNLP system. The special procedure also exists which can be prepared by user. The control parameters in this procedure are the desirable "applied" parameters.

The penalties for differences between energy productions and demands are the additional user control parameters in w.r.d. problem. Let us consider, for instance, the following situation. In some country the energy production in any month is almost equal to the demand. The other country has stable level of energy production which is lower than the demand. Which situation is more favorable ? Of course, it depends on payment for energy deficit for each country. In this case we may establish various penalties for deficit in different countries. The penalty may be changed during decision making process.

So the interface of applied optimization system POOL consists of universal set DISO/PC-MCNLP system windows (see Kotkin, 1989, and Appendix 3)) and original POOL system windows (see Appendix 3).

5. Numerical methods and results.

It should be noticed that the nonlinear programming study differs from linear programming by a grate effort with respect to task definition and universal nonlinear programming method elaboration (see Fig. 7). A great variety of numerical algorithms and dialogue procedures are available in DISO/PC-MCNLP system. The proper algorithm for solution of the problem may be chosen. The DISO/PC-MCNLP has multi level structure with respect to numerical methods (see Fig. 8). Several unconstrained minimization methods are available on the basic level. They are the result of long time experience of the group of the scientists from Computer Center of the USSR Academy of Sciences (Evtushenco, 1985). Since any multicriteria programming problem is usually reduced to one goal programming problem the next level is built by a number of nonlinear program-

ming techniques. A series of decision making procedures builds the last level.

The DISO/PC-MCNLP includes the following methods.

Unconstrained minimization methods:

1) coordinate descent ,
2) direct search (two modifications) ,
3) random search method ,
4) conjugate gradient ,
5) Newton method.

Nonlinear programming methods:

1) center method modifications ,
2) penalty functions method modifications ,
3) barrier methods ,
4) exact penalty function method modifications.

Decision making methods:

1) gradient method (Geoffrion) ,
2) parametric programming method (Guddat) ,
3) reference point method modifications ,
4) scalarization method modifications ,
5) nonlinear parametric programming method.

All these methods are included in POOL system however the POOL system interface is adjusted in the following way:

1) random search is used when identification task is being solved ,
2) conjugate gradient is used on the basis level of multicriteria task for solution of the additive unconstrained minimization problem ,
3) center method modification is used for solution of the additive nonlinear programming problem ,
4) reference point method modification or parametric programming method are used for decision making process.

One can see the set of some optimal solutions (called Pareto optimal solutions) in Appendix 4.

References

1. Evtushenko Yu.G. (1985) Numerical Optimization Technique. - Springer Verlag. New-York.

2. Kotkin G.G. (1988) Topological Properties of Perturbed Pareto Set (in russian). - Computer Center of the USSR Academy of Sciences. Moscow.

3. Kotkin G.G. (1989). Nonlinear and Multicriteria Programming: Theory and Software. - IIASA Working Paper. (in press).

4. Mazourik V.P. (1988) Field Manager Package. - WP-89-009. IIASA. Laxenburg. Austria.

5. Salewicz K.A., Loucks D.P. and McDonald A. (1989) Desision support systems for managing large international rivers. - Report. IIASA, A-2361 Laxenburg, Austria.

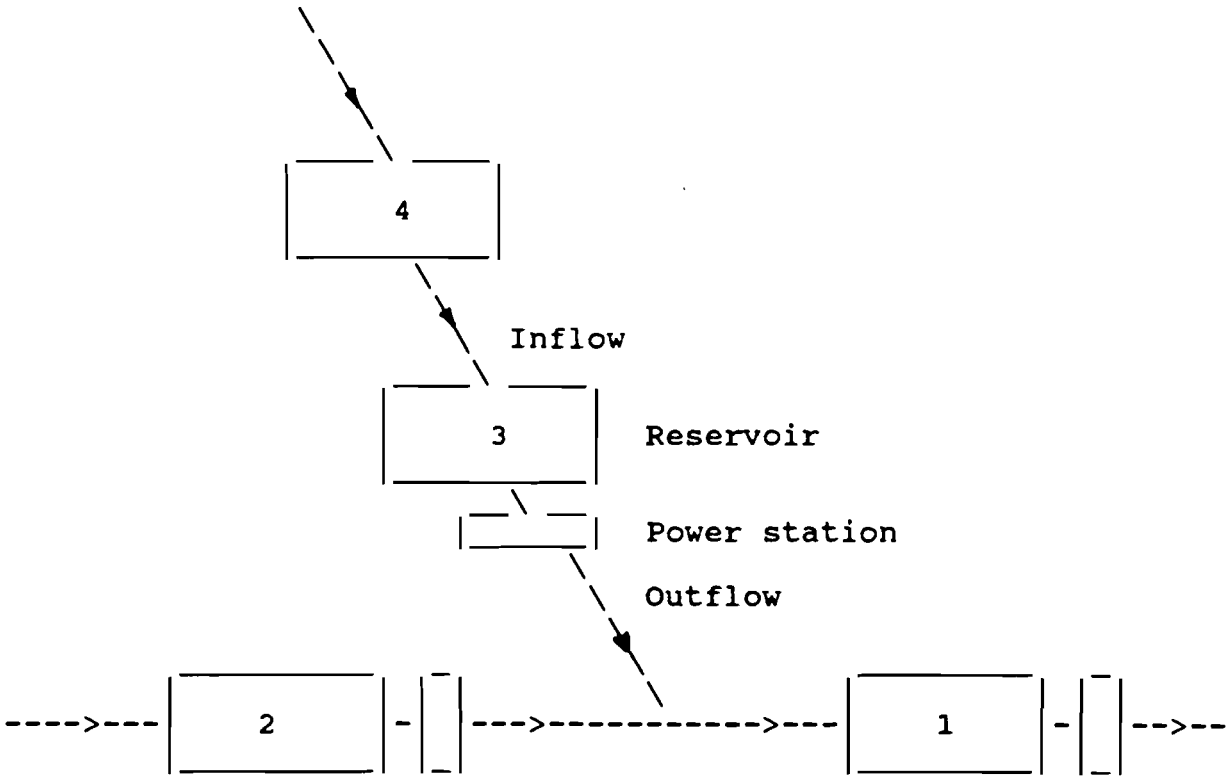


Fig.1. *Principle scheme of Zambesi river system. Numbers 1,2,3 and 4 denote Cabora Bassa, Kariba, Kasaka and Itezhitezhi reservoirs respectively.*

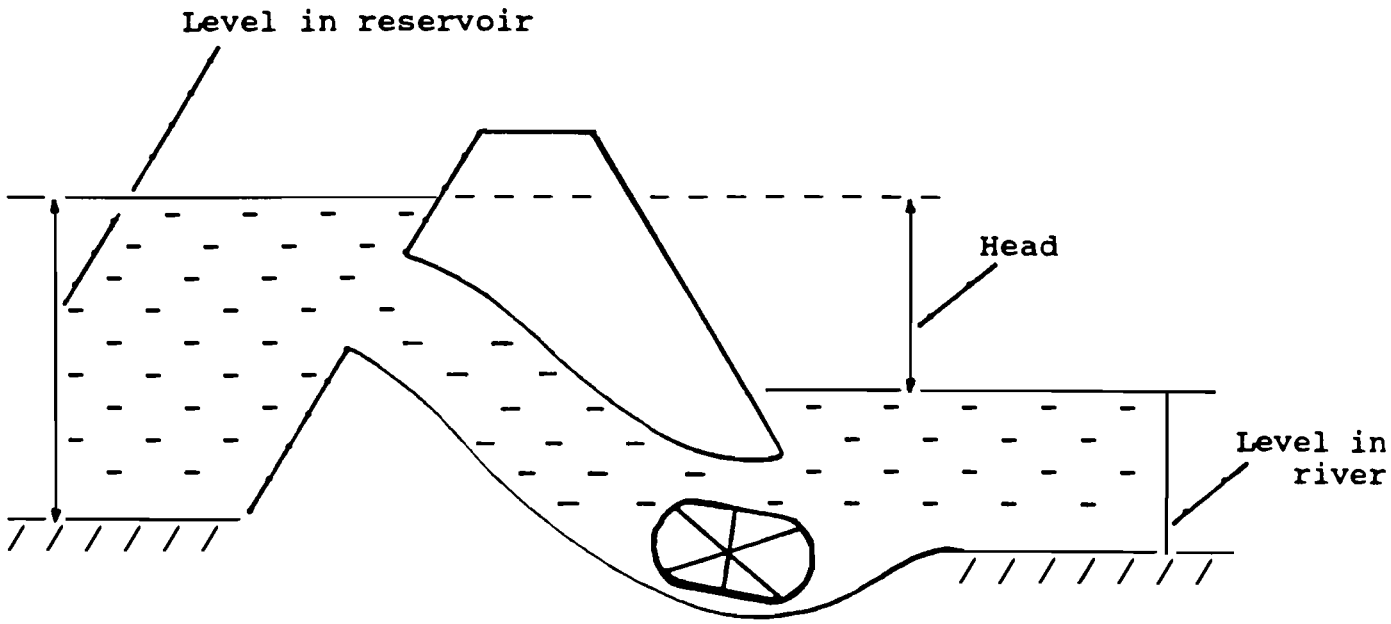


Fig.2. *Principle scheme of an electro-power station.*

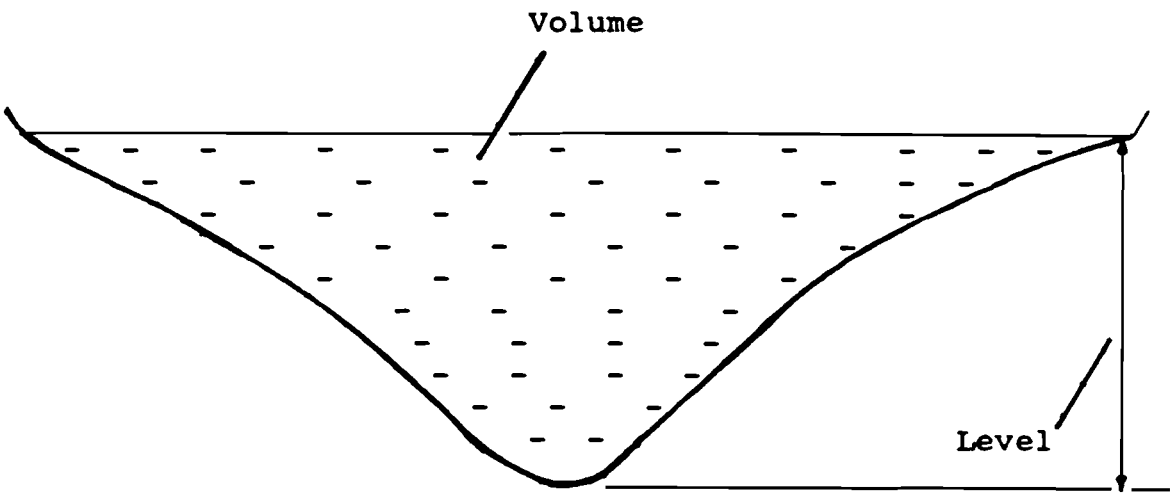


Fig.3. *The typical configuration of reservoirs composing Zambezi river system.*

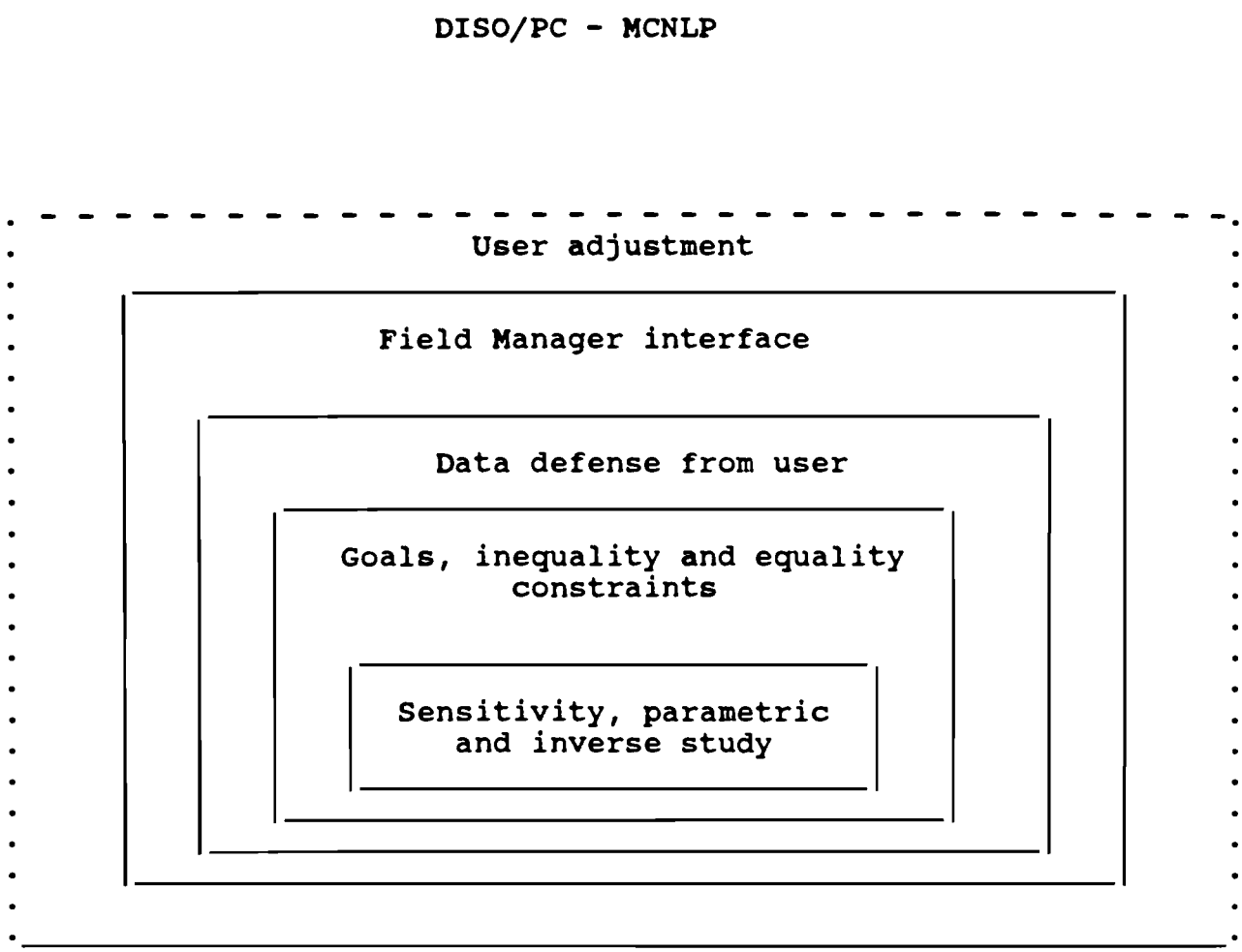


Fig.4. *The structure of interactive system DISO/PC-MCNLP.*

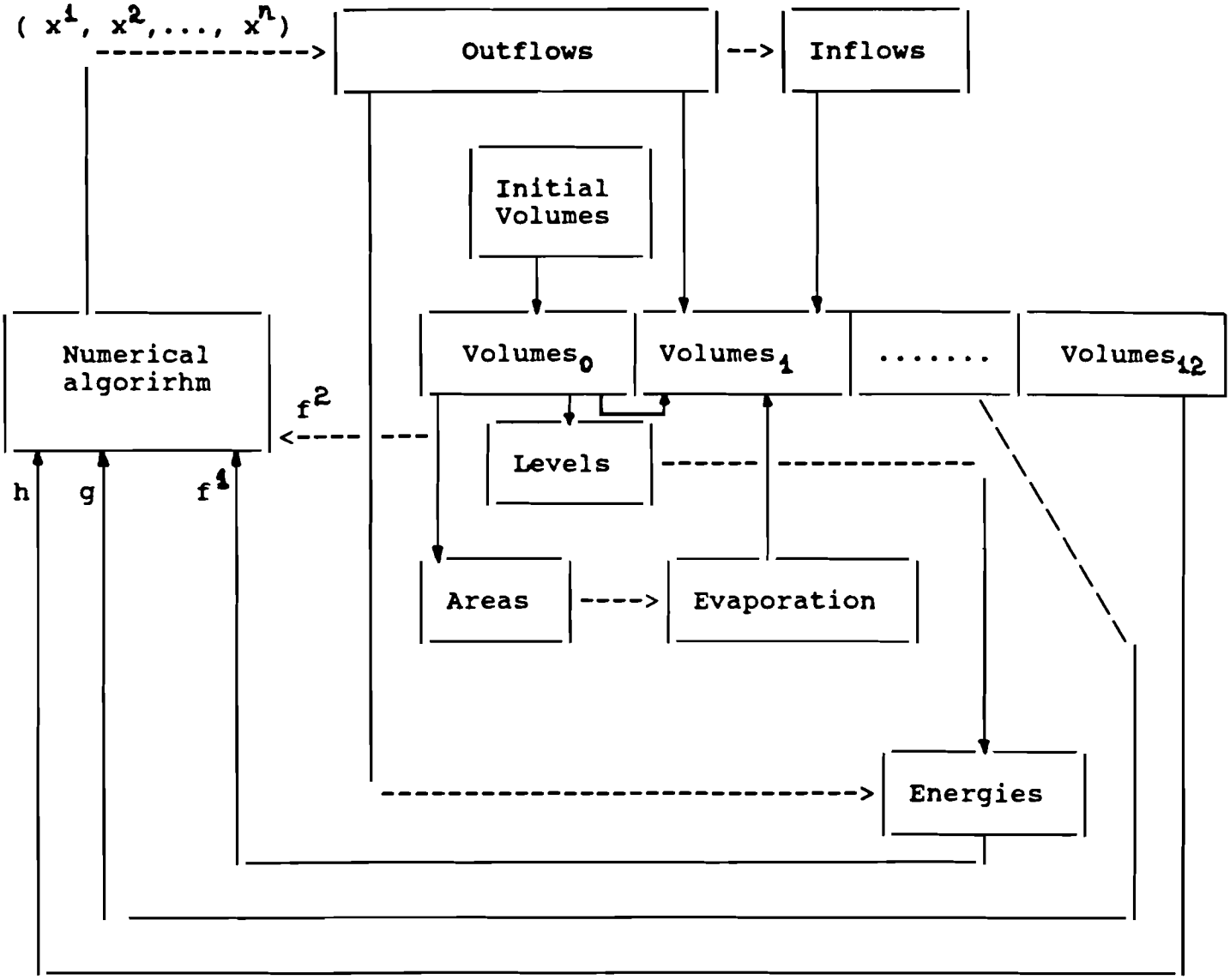


Fig.5. *Calculation scheme of the water resources distribution problem.*

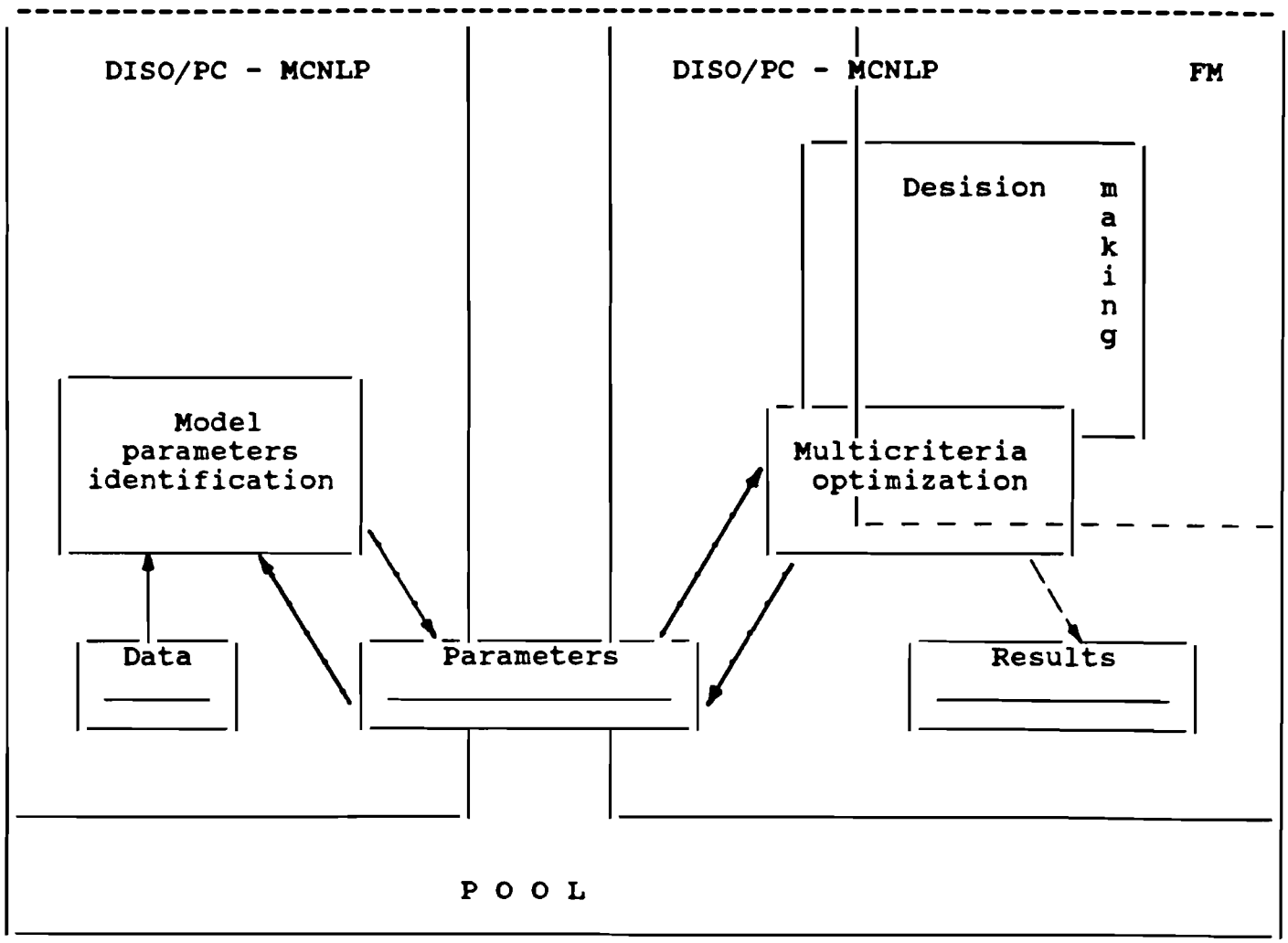


Fig.6. *The structure of the POOL interactive system.*

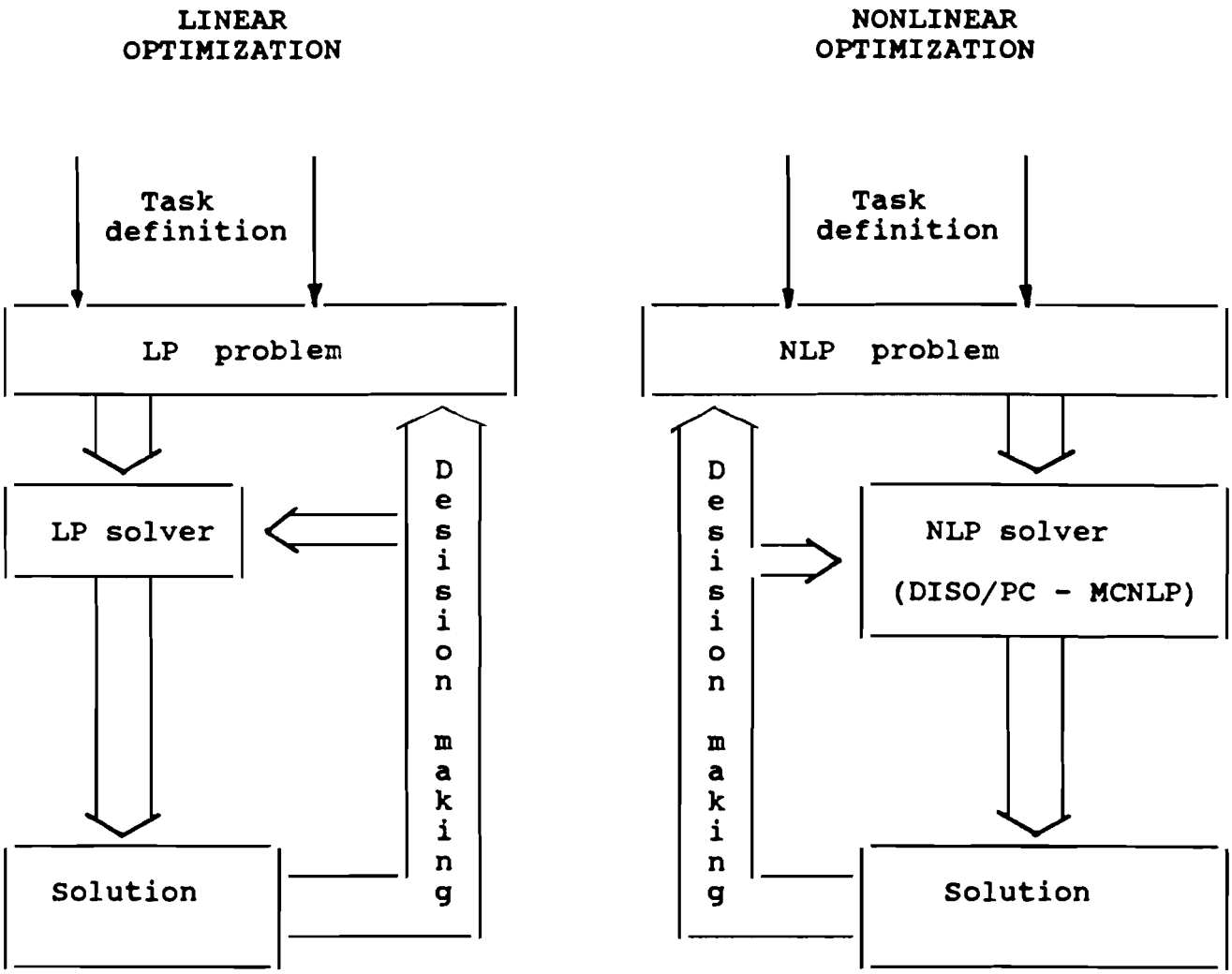


Fig.7. *Decision making process in linear and nonlinear programming.*

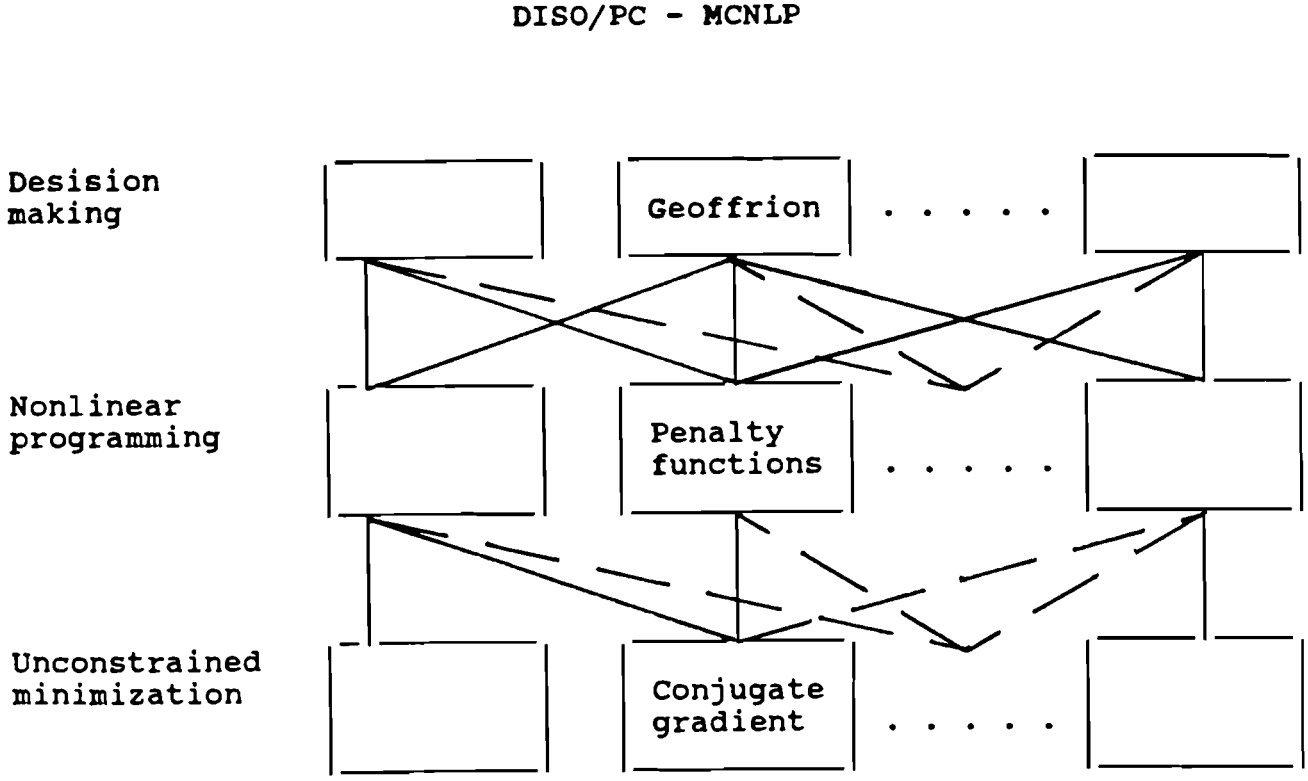


Fig.8. *The seria of numerical methods implemented in DISO/PC- MCNLP system.*

Appendix 1

Monthly averaged differences between evaporation and precipitation for reservoirs composing Zambezi river system (in mm/month)

month	Cabora Bassa	Itezhitehzi, Kasaka, Kariba
Oct.	171.0983	167.2
Nov.	124.2300	115.8
Dec.	-4.36535	-39.5
Jan.	-9.49065	-36.8
Feb.	-30.5228	-66.9
Mar.	40.37027	7.4
Apr.	98.91526	91.0
May.	111.9368	111.3
Jun.	101.1411	99.4
Jul.	86.61708	86.5
Aug.	112.8515	111.3
Sep.	147.1193	146.4

Appendix 2

1. Program for the identification task.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mem.h>
#include <alloc.h>
#include "fm.def"
#include "ucm_mon.def"
#include "conv_.def"
#define DATA_NUMBER  9
double L_fun1 () ;
double L_fun2 () ;
double L_fun3 () ;
double L_fun4 () ;
double A_fun1 () ;
double A_fun2 () ;
double A_fun3 () ;
double A_fun4 () ;
static double *param;
static double *area1;
static double *area2;
static double *area3;
static double *area4;
static double *volume1;
static double *volume2;
static double *volume3;
static double *volume4;
static double *level1;
static double *level2;
static double *level3;
static double *level4;
static double *nu1;
static double *nu2;
static double *h1;
static double *h2;
static double *h3;
static double *n1;
static double *n2;
static double *n3;
static double *inf2;
static double *inf4;
void main ()
{
int dim_x = 24;
int dim_y = 8;
   double * low  = (double*) calloc (dim_x, sizeof (double));
   double * up   = (double*) calloc (dim_x, sizeof (double));
   double * x    = (double*) calloc (dim_x, sizeof (double));
   p_fun funs    = (p_fun)   calloc (dim_y, sizeof (d_fun));
   int * obj_fun = (int*)    calloc (dim_y, sizeof (int));
   double * point = (double*)  calloc (dim_y, sizeof  (double));
   double * region = (double*)  calloc (dim_y, sizeof  (double));
   int i;
```

```
char **points;
area1 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
area2 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
area3 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
area4 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
volume1 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
volume2 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
volume3 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
volume4 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
level1 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
level2 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
level3 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
level4 = (double*)  calloc (DATA_NUMBER, sizeof  (double));
param =  (double*)  calloc (N_USER_PAR, sizeof  (double));
nu1    =  (double*)  calloc (12, sizeof  (double));
nu2    =  (double*)  calloc (12, sizeof  (double));
h1     =  (double*)  calloc (4, sizeof  (double));
h2     =  (double*)  calloc (4, sizeof  (double));
h3     =  (double*)  calloc (4, sizeof  (double));
n1     =  (double*)  calloc (4, sizeof  (double));
n2     =  (double*)  calloc (4, sizeof  (double));
n3     =  (double*)  calloc (4, sizeof  (double));
inf2   =  (double*)  calloc (12, sizeof  (double));
inf4   =  (double*)  calloc (12, sizeof  (double));
for (i = 0; i < N_USER_PAR; i++) {
   param [i] = 0.0;
}
  low [0] = -1e8;   x [0] =  1.0e-6;   up [0] = 1e8;
  low [1] = -1e8;   x [1] =  1.0e-6;   up [1] = 1e8;
  low [2] = -1e8;   x [2] =  1.0e-6;   up [2] = 1e8;
  low [3] = -1e8;   x [3] =  1.0e-6;   up [3] = 1e8;
  low [4] = -1e8;   x [4] =  1.0e-2;   up [4] = 1e8;
  low [5] = -1e8;   x [5] =  1.0e-2;   up [5] = 1e8;
  low [6] = -1e8;   x [6] =  1.0e-2;   up [6] = 1e8;
  low [7] = -1e8;   x [7] =  1.0e-2;   up [7] = 1e8;
  low [8] = -1e8;   x [8] =  1.0e+2;   up [8] = 1e8;
  low [9] = -1e8;   x [9] =  1.0e+2;   up [9] = 1e8;
  low [10] = -1e8;  x [10] =  1.0e+2;  up [10] = 1e8;
  low [11] = -1e8;  x [11] =  1.0e+2;  up [11] = 1e8;
  low [12] = -1e8;  x [12] =  1.0e-6;  up [12] = 1e8;
  low [13] = -1e8;  x [13] =  1.0e-6;  up [13] = 1e8;
  low [14] = -1e8;  x [14] =  1.0e-6;  up [14] = 1e8;
  low [15] = -1e8;  x [15] =  1.0e-6;  up [15] = 1e8;
  low [16] = -1e8;  x [16] =  1.0e-2;  up [16] = 1e8;
  low [17] = -1e8;  x [17] =  1.0e-2;  up [17] = 1e8;
  low [18] = -1e8;  x [18] =  1.0e-2;  up [18] = 1e8;
  low [19] = -1e8;  x [19] =  1.0e-2;  up [19] = 1e8;
  low [20] = -1e8;  x [20] =  1.0e+3;  up [20] = 1e8;
  low [21] = -1e8;  x [21] =  1.0e+3;  up [21] = 1e8;
  low [22] = -1e8;  x [22] =  1.0e+3;  up [22] = 1e8;
  low [23] = -1e8;  x [23] =  1.0e+3;  up [23] = 1e8;
obj_fun [0] =  1;   point [0] = 0.0;   region [0] = 50.0;
obj_fun [1] =  1;   point [1] = 0.0;   region [1] = 50.0;
obj_fun [2] =  1;   point [2] = 0.0;   region [2] = 50.0;
obj_fun [3] =  1;   point [3] = 0.0;   region [3] = 50.0;
obj_fun [4] =  1;   point [4] = 0.0;   region [4] = 50.0;
obj_fun [5] =  1;   point [5] = 0.0;   region [5] = 50.0;
obj_fun [6] =  1;   point [6] = 0.0;   region [6] = 50.0;
```

```
   obj_fun [7] =  1;   point [7] = 0.0;   region [7] = 50.0;
   funs [0] = L_fun1;
   funs [1] = L_fun2;
   funs [2] = L_fun3;
   funs [3] = L_fun4;
   funs [4] = A_fun1;
   funs [5] = A_fun2;
   funs [6] = A_fun3;
   funs [7] = A_fun4;
   get_data ();
   conv_init (dim_x, dim_y, low, up, x, funs);
   set_aim (obj_fun, point, region);
   set_par_free (param);
   /* file_read (); */
   for (i = 0; i < dim_y; i++) {
      if (c_aim -> obj_fun [i] == 1) {
         c_fold -> p_r [i] = 1;
         c_fold -> p_l [i] = 1;
         c_fold -> u_r [i] = 1.0;
         c_fold -> u_l [i] = 1.0;
      }else {
         c_fold -> p_r [i] = 2;
         c_fold -> p_l [i] = 2;
      }
   }
   c_fold -> meth_kind = 3;
   UCM_accuracy   = 2.0e-7;
   UCM_it_max     = 300;
   unicon ();
   /* Coef-ts:  0 - Cabora Bassa, 1 - Kariba, 2 - Kasaka, 3 -
  Itezhitezhi */
   h1 [0] = c_fold -> x [0 ];
   h1 [1] = c_fold -> x [1 ];
   h1 [2] = c_fold -> x [2 ];
   h1 [3] = c_fold -> x [3 ];
   h2 [0] = c_fold -> x [4 ];
   h2 [1] = c_fold -> x [5 ];
   h2 [2] = c_fold -> x [6 ];
   h2 [3] = c_fold -> x [7 ];
   h3 [0] = c_fold -> x [8 ];
   h3 [1] = c_fold -> x [9 ];
   h3 [2] = c_fold -> x [10];
   h3 [3] = c_fold -> x [11];
   n1 [0] = c_fold -> x [12];
   n1 [1] = c_fold -> x [13];
   n1 [2] = c_fold -> x [14];
   n1 [3] = c_fold -> x [15];
   n2 [0] = c_fold -> x [16];
   n2 [1] = c_fold -> x [17];
   n2 [2] = c_fold -> x [18];
   n2 [3] = c_fold -> x [19];
   n3 [0] = c_fold -> x [20];
   n3 [1] = c_fold -> x [21];
   n3 [2] = c_fold -> x [22];
   n3 [3] = c_fold -> x [23];
   coef_prep ();
   conv_term (x, points);
}
```

```
int get_data ()
{
   FILE *stm;
   char line [100];
     int i;
   /*  1 - Cabora Basa  */
   level1 [0] = 295.0;
   level1 [1] = 296.0;
   level1 [2] = 301.0;
   level1 [3] = 306.0;
   level1 [4] = 311.0;
   level1 [5] = 316.0;
   level1 [6] = 321.0;
   level1 [7] = 326.0;
   level1 [8] = 331.0;
   area1 [0] =  845.0;
   area1 [1] =  885.0;
   area1 [2] = 1120.0;
   area1 [3] = 1375.0;
   area1 [4] = 1650.0;
   area1 [5] = 1960.0;
   area1 [6] = 2310.0;
   area1 [7] = 2680.0;
   area1 [8] = 3040.0;
   volume1 [0] =     0.0;
   volume1 [1] =   865.0;
   volume1 [2] =  5840.0;
   volume1 [3] = 12060.0;
   volume1 [4] = 19640.0;
   volume1 [5] = 28640.0;
   volume1 [6] = 39320.0;
   volume1 [7] = 51750.0;
   volume1 [8] = 66010.0;
   /*  2 - Kariba  */
   level2 [0] = 475.5;
   level2 [1] = 475.9;
   level2 [2] = 477.0;
   level2 [3] = 479.0;
   level2 [4] = 481.0;
   level2 [5] = 483.0;
   level2 [6] = 485.0;
   level2 [7] = 487.0;
   level2 [8] = 489.5;
   area2 [0] = 4354.0;
   area2 [1] = 4395.0;
   area2 [2] = 4507.0;
   area2 [3] = 4709.0;
   area2 [4] = 4901.0;
   area2 [5] = 5081.0;
   area2 [6] = 5261.0;
   area2 [7] = 5440.0;
   area2 [8] = 5668.0;
   volume2 [0] =    50.0;
   volume2 [1] =  1830.0;
   volume2 [2] =  6710.0;
   volume2 [3] = 15910.0;
   volume2 [4] = 25480.0;
   volume2 [5] = 35430.0;
```

```
volume2 [6] = 45780.0;
volume2 [7] = 56510.0;
volume2 [8] = 70410.0;
/*  3 - Kafue-Kasaka  */
level3 [0] = 974.0;
level3 [1] = 974.4;
level3 [2] = 974.8;
level3 [3] = 975.2;
level3 [4] = 975.6;
level3 [5] = 976.2;
level3 [6] = 976.8;
level3 [7] = 977.4;
level3 [8] = 977.9;
area3 [0] =   78.0;
area3 [1] =   95.0;
area3 [2] =  143.0;
area3 [3] =  290.0;
area3 [4] =  562.0;
area3 [5] = 1048.0;
area3 [6] = 1497.0;
area3 [7] = 1901.0;
area3 [8] = 2196.0;
volume3 [0] =   18.0;
volume3 [1] =   22.0;
volume3 [2] =   31.0;
volume3 [3] =   73.0;
volume3 [4] =  179.0;
volume3 [5] =  544.0;
volume3 [6] = 1125.0;
volume3 [7] = 2020.0;
volume3 [8] = 2930.0;
/*  4 - Itezhitezhi  */
level4 [0] =  990.0;
level4 [1] =  995.0;
level4 [2] = 1005.0;
level4 [3] = 1015.0;
level4 [4] = 1020.0;
level4 [5] = 1025.0;
level4 [6] = 1035.0;
level4 [7] = 1045.0;
level4 [8] = 1050.0;
area4 [0] =   8.0;
area4 [1] =  28.0;
area4 [2] =  85.0;
area4 [3] = 166.0;
area4 [4] = 225.0;
area4 [5] = 295.0;
area4 [6] = 471.0;
area4 [7] = 665.0;
area4 [8] = 857.0;
volume4 [0] =     4 0;
volume4 [1] =   140.0;
volume4 [2] =   690.0;
volume4 [3] =  1910.0;
volume4 [4] =  2900.0;
volume4 [5] =  4220.0;
volume4 [6] =  8000.0;
volume4 [7] = 13550.0;
```

```
   volume4 [8] = 17170.0;
   return (1);
}
/*  Functions  */
double L_fun1 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
   for (i = 0; i < DATA_NUMBER; i++) {
      a = x [0] * volume1 [i] * volume1 [i] + x [4] * volume1 [i]
      + x [8];
      a = a - level1 [i];
      sum = sum + a * a;
   }
   return (sum);
}
double L_fun2 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
        a = x [1] * volume2 [i] * volume2 [i] + x [5] * volume2
        [i] + x [9];
        a = a - level2 [i];
        sum = sum + a * a;
     }
     return (sum);
}
double L_fun3 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
         a = x [2] * volume3 [i] * volume3 [i] + x [6] * volume3
         [i] + x [10];
         a = a - level3 [i];
         sum = sum + a * a;
    }
    return (sum);
}
double L_fun4 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
        a = x [3] * volume4 [i] * volume4 [i] + x [7] * volume4
        [i] + x [11];
        a = a - level4 [i];
        sum = sum + a * a;
    }
```

```
    return (sum);
}
double A_fun1 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
        a = x [12] * volume1 [i] * volume1 [i] + x [16] * volume1
        [i] + x [20];
        a = a - area1 [i];
        sum = sum + a * a;
     }
     return (sum);
}
double A_fun2 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
        a = x [13] * volume2 [i] * volume2 [i] + x [17] * volume2
        [i] + x [21];
        a = a - area2 [i];
        sum = sum + a * a;
     }
     return (sum);
}
double A_fun3 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
         a = x [14] * volume3 [i] * volume3 [i] + x [18] *
         volume3 [i] + x [22];
         a = a - area3 [i];
         sum = sum + a * a;
     }
     return (sum);
}
double A_fun4 (x)
double *x;
{
   int i;
   double a;
   double sum = 0.0;
     for (i = 0; i < DATA_NUMBER; i++) {
         a = x [15] * volume4 [i] * volume4 [i] + x [19] *
         volume4 [i] + x [23];
         a = a - area4 [i];
         sum = sum + a * a;
     }
     return (sum);
}
```

2. Program for the optimization task.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mem.h>
#include <alloc.h>
#include "ucm_mon.def"
#include "ucm7.def"
#include "conv_.def"
#include "fm.def"
#define NO_VALUE      1.0
#define MONTH_NUMBER  12
#define EPS           1.11e-100
#define RESER_NUMBER  4
#define pv1         param [ 0]
#define pv2         param [ 1]
#define pv3         param [ 2]
#define pv4         param [ 3]
#define pe1         param [ 4]
#define pe2         param [ 5]
#define pe3         param [ 6]
#define ini1        param [105]
#define ini2        param [106]
#define ini3        param [109]
#define ini4        param [110]
#define ref1        param [111]
#define ref2        param [112]
#define ref3        param [113]
#define ref4        param [114]
#define demand1     param [115]
#define demand2     param [116]
#define demand3     param [117]
#define  min_lev1   param [118]
#define  min_lev2   param [119]
#define  min_lev3   param [120]
#define  min_lev4   param [121]
#define  t_lev1     param [122]
#define  t_lev2     param [123]
#define  t_lev3     param [124]
#define perm_dev1   param [220]
#define perm_dev2   param [221]
#define perm_dev3   param [222]
/*   for   search/replace
#define out1 (i)     x [i]
#define out2 (i)     x [i+12]
#define out3 (i)     x [i+24]
#define out4 (i)     x [i+36]
*/
void calc_manth_data (/* double *x; */);
double Inf1 (/* double *x;    int i; */);
double Inf3 (/* double *x; int i; *);
double Area (/* double vol; int j; */);
double Level (/* double vol; int j;     */);
double Volume1 (/* double *x; int i; */);
```

```
double Volume2 (/* double *x; int i; */);
double Volume3 (/* double *x; int i; */);
double Volume4 (/* double *x; int i; */);
double Energy1 (/* double *x; */);
double Energy2 (/* double *x */);
double Energy3 (/* double *x */);
double Dvol_1_11 (/* double *x */);
double Dvol_2_11 (/* double *x */);
double Dvol_3_11 (/* double *x */);
double Dvol_4_11 (/* double *x */);
double Dlev1 (/* double *x */);
double Dlev2 (/* double *x */);
double Dlev3 (/* double *x */);
double Dlev4 (/* double *x */);
double Rlev1 (/* double *x */);
double Rlev2 (/* double *x */);
double Rlev3 (/* double *x */);
double Rlev4 (/* double *x */);
static double *param;
static double *nu1;
static double *nu2;
static double *h1;
static double *h2;
static double *h3;
static double *n1;
static double *n2;
static double *n3;
static double *inf3;
static double *inf1;
static double *inf2;
static double *inf4;
static double *vol1;
static double *vol2;
static double *vol3;
static double *vol4;
static double *lev1;
static double *lev2;
static double *lev3;
static double *lev4;
static double *dev_ener1;
static double *dev_ener2;
static double *dev_ener3;
void main ()
{
int dim_x = 48;
int dim_y = 15;
   double * low  = (double*) calloc (dim_x, sizeof (double));
   double * up   = (double*) calloc (dim_x, sizeof (double));
   double * x    = (double*) calloc (dim_x, sizeof (double));
   p_fun funs    = (p_fun)   calloc (dim_y, sizeof (d_fun));
   int * obj_fun = (int*)    calloc (dim_y, sizeof (int));
   double * point  = (double*)  calloc (dim_y, sizeof  (double));
   double * region  = (double*)  calloc (dim_y, sizeof (double));
   int i;
   char **points;
   param = (double*) calloc (N_USER_PAR, sizeof(double));
   h1  = (double*) calloc (RESER_NUMBER, sizeof(double));
   h2  = (double*) calloc (RESER_NUMBER, sizeof(double));
```

```
h3  = (double*) calloc (RESER_NUMBER, sizeof(double));
n1  = (double*) calloc (RESER_NUMBER, sizeof(double));
n2  = (double*) calloc (RESER_NUMBER, sizeof(double));
n3  = (double*) calloc (RESER_NUMBER, sizeof(double));
nu1  = (double*) calloc (MONTH_NUMBER, sizeof(double));
nu2  = (double*) calloc (MONTH_NUMBER, sizeof(double));
inf1 = (double*) calloc (MONTH_NUMBER, sizeof(double));
inf3 = (double*) calloc (MONTH_NUMBER, sizeof(double));
inf2 = (double*) calloc (MONTH_NUMBER, sizeof(double));
inf4 = (double*) calloc (MONTH_NUMBER, sizeof(double));
vol1 = (double*) calloc (MONTH_NUMBER, sizeof(double));
vol2 = (double*) calloc (MONTH_NUMBER, sizeof(double));
vol3 = (double*) calloc (MONTH_NUMBER, sizeof(double));
vol4 = (double*) calloc (MONTH_NUMBER, sizeof(double));
lev1 = (double*) calloc (MONTH_NUMBER, sizeof(double));
lev2 = (double*) calloc (MONTH_NUMBER, sizeof(double));
lev3 = (double*) calloc (MONTH_NUMBER, sizeof(double));
lev4 = (double*) calloc (MONTH_NUMBER, sizeof(double));
dev_ener1 = (double*) calloc (MONTH_NUMBER, sizeof(double));
dev_ener2 = (double*) calloc (MONTH_NUMBER, sizeof(double));
dev_ener3 = (double*) calloc (MONTH_NUMBER, sizeof(double));
 for (i = 0; i < N_USER_PAR; i++) {
   param [i] = NO_VALUE;
}
for (i = 0; i < RESER_NUMBER; i++) {
   h1 [i] = NO_VALUE;
   h2 [i] = NO_VALUE;
   h3 [i] = NO_VALUE;
   n1 [i] = NO_VALUE;
   n2 [i] = NO_VALUE;
   n3 [i] = NO_VALUE;
}
for (i = 0; i < MONTH_NUMBER; i++) {
   nu1  [i] = NO_VALUE;
   nu2  [i] = NO_VALUE;
   inf1 [i] = NO_VALUE;
   inf2 [i] = NO_VALUE;
   inf3 [i] = NO_VALUE;
   inf4 [i] = NO_VALUE;
   vol1 [i] = NO_VALUE;
   vol2 [i] = NO_VALUE;
   vol3 [i] = NO_VALUE;
   vol4 [i] = NO_VALUE;
   lev1 [i] = NO_VALUE;
   lev2 [i] = NO_VALUE;
   lev3 [i] = NO_VALUE;
   lev4 [i] = NO_VALUE;
   dev_ener1 [i] = NO_VALUE;
   dev_ener2 [i] = NO_VALUE;
   dev_ener3 [i] = NO_VALUE;
}
  low [ 0] = 0.00001; x [ 0] = 1.0E-05; up [ 0] = 100000000.0;
  low [ 1] = 0.00001; x [ 1] = 4.4E+01; up [ 1] = 100000000.0;
  low [ 2] = 0.00001; x [ 2] = 7.8E+02; up [ 2] = 100000000.0;
  low [ 3] = 0.00001; x [ 3] = 5.3E+03; up [ 3] = 100000000.0;
  low [ 4] = 0.00001; x [ 4] = 4.9E+03; up [ 4] = 100000000.0;
  low [ 5] = 0.00001; x [ 5] = 2.4E+03; up [ 5] = 100000000.0;
  low [ 6] = 0.00001; x [ 6] = 1.7E+03; up [ 6] = 100000000.0;
```

```
   low [ 7] = 0.00001; x [ 7] = 2.7E+03; up [ 7] = 100000000.0;
   low [ 8] = 0.00001; x [ 8] = 3.6E+03; up [ 8] = 100000000.0;
   low [ 9] = 0.00001; x [ 9] = 3.3E+03; up [ 9] = 100000000.0;
   low [10] = 0.00001; x [10] = 3.2E+03; up [10] = 100000000.0;
   low [11] = 0.00001; x [11] = 3.8E+03; up [11] = 100000000.0;
   low [12] = 0.00001; x [12] = 3.7E+04; up [12] = 100000000.0;
   low [13] = 0.00001; x [13] = 3.6E+03; up [13] = 100000000.0;
   low [14] = 0.00001; x [14] = 5.1E+02; up [14] = 100000000.0;
   low [15] = 0.00001; x [15] = 3.6E+03; up [15] = 100000000.0;
   low [16] = 0.00001; x [16] = 1.5E+03; up [16] = 100000000.0;
   low [17] = 0.00001; x [17] = 6.8E+02; up [17] = 100000000.0;
   low [18] = 0.00001; x [18] = 8.3E+02; up [18] = 100000000.0;
   low [19] = 0.00001; x [19] = 1.2E+03; up [19] = 100000000.0;
   low [20] = 0.00001; x [20] = 1.5E+03; up [20] = 100000000.0;
   low [21] = 0.00001; x [21] = 1.0E-05; up [21] = 100000000.0;
   low [22] = 0.00001; x [22] = 1.0E-05; up [22] = 100000000.0;
   low [23] = 0.00001; x [23] = 1.0E-05; up [23] = 100000000.0;
   low [24] = 0.00001; x [24] = 3.8E+03; up [24] = 100000000.0;
   low [25] = 0.00001; x [25] = 9.7E+03; up [25] = 100000000.0;
   low [26] = 0.00001; x [26] = 1.8E+03; up [26] = 100000000.0;
   low [27] = 0.00001; x [27] = 1.2E+03; up [27] = 100000000.0;
   low [28] = 0.00001; x [28] = 2.3E+03; up [28] = 100000000.0;
   low [29] = 0.00001; x [29] = 1.8E+03; up [29] = 100000000.0;
   low [30] = 0.00001; x [30] = 1.1E+03; up [30] = 100000000.0;
   low [31] = 0.00001; x [31] = 1.7E+03; up [31] = 100000000.0;
   low [32] = 0.00001; x [32] = 6.5E+02; up [32] = 100000000.0;
   low [33] = 0.00001; x [33] = 1.0E-05; up [33] = 100000000.0;
   low [34] = 0.00001; x [34] = 1.0E-05; up [34] = 100000000.0;
   low [35] = 0.00001; x [35] = 1.0E-05; up [35] = 100000000.0;
   low [36] = 0.00001; x [36] = 4.4E-01; up [36] = 100000000.0;
   low [37] = 0.00001; x [37] = 4.4E-01; up [37] = 100000000.0;
   low [38] = 0.00001; x [38] = 4.4E-01; up [38] = 100000000.0;
   low [39] = 0.00001; x [39] = 1.0E+01; up [39] = 100000000.0;
   low [40] = 0.00001; x [40] = 1.2E+01; up [40] = 100000000.0;
   low [41] = 0.00001; x [41] = 1.4E+03; up [41] = 100000000.0;
   low [42] = 0.00001; x [42] = 1.5E+03; up [42] = 100000000.0;
   low [43] = 0.00001; x [43] = 2.3E+03; up [43] = 100000000.0;
   low [44] = 0.00001; x [44] = 1.1E+03; up [44] = 100000000.0;
   low [45] = 0.00001; x [45] = 3.6E+02; up [45] = 100000000.0;
   low [46] = 0.00001; x [46] = 5.3E+02; up [46] = 100000000.0;
   low [47] = 0.00001; x [47] = 1.8E+03; up [47] = 100000000.0;
/*  Energy j  */
obj_fun [  0] =  1;  point [  0] = 0.0;  region [  0] = 10.0;
obj_fun [  1] =  1;  point [  1] = 0.0;  region [  1] = 10.0;
obj_fun [  2] =  1;  point [  2] = 0.0;  region [  2] = 10.0;
/*  Dlev j  */
obj_fun [  3] =  1;  point [  3] = 0.0;  region [  3] = 10.0;
obj_fun [  4] =  1;  point [  4] = 0.0;  region [  4] = 10.0;
obj_fun [  5] =  1;  point [  5] = 0.0;  region [  5] = 10.0;
obj_fun [  6] =  1;  point [  6] = 0.0;  region [  6] = 10.0;
/*  Rlev j  */
obj_fun [  7] = -1;  point [  7] = 0.0;  region [  7] = 10.0;
obj_fun [  8] = -1;  point [  8] = 0.0;  region [  8] = 10.0;
obj_fun [  9] = -1;  point [  9] = 0.0;  region [  9] = 10.0;
obj_fun [ 10] = -1;  point [ 10] = 0.0;  region [ 10] = 10.0;
/*  Vol j 11  */
obj_fun [ 11] = -1;  point [ 11] = 0.0;  region [ 11] = 10.0;
obj_fun [ 12] = -1;  point [ 12] = 0.0;  region [ 12] = 10.0;
```

```
   obj_fun [ 13] = -1;  point [ 13] = 0.0;  region [ 13] = 10.0;
   obj_fun [ 14] = -1;  point [ 14] = 0.0;  region [ 14] = 10.0;
   funs [ 0] =  Energy1;
   funs [ 1] =  Energy2;
   funs [ 2] =  Energy3;
   funs [ 3] =  Dlev1;
   funs [ 4] =  Dlev2;
   funs [ 5] =  Dlev3;
   funs [ 6] =  Dlev4;
   funs [ 7] =  Rlev1;
   funs [ 8] =  Rlev2;
   funs [ 9] =  Rlev3;
   funs [10] =  Rlev4;
   funs [11] = Dvol_1_11;
   funs [12] = Dvol_2_11;
   funs [13] = Dvol_3_11;
   funs [14] = Dvol_4_11;
   conv_init (dim_x, dim_y, low, up, x, funs);
   set_aim (obj_fun, point, region);
   set_par_free (param);
   coef_set ();
   for (i = 0; i < dim_y; i++) {
      if (c_aim -> obj_fun [i] == 1) {
         c_fold -> v [i] = 0.0;
         c_fold -> u_r [i] = 1.0;
      }else {
         c_fold -> u_r [i] = 100000.0;
      }
   }
   for (i = 0; i < 3; i++) {
      c_fold -> u_r [i] = 1.0;
   }
   for (i = 3; i < 7; i++) {
      c_fold -> u_r [i] = 0.1;
   }
   c_fold -> meth_kind = 5;
     UCM7_restart   = 48;
   UCM_accuracy   = 2.0e-7;
   UCM_it_max     = 40;
   unicon ();
   file_write ();
     conv_term (x, points);
} /*  end of main  */
      coef_set ()
{
   h1 [0] =  param [10] = 8.1E-07  ;
   h1 [1] =  param [11] = -2.8E-07 ;
   h1 [2] =  param [12] = 7.9E-07  ;
   h1 [3] =  param [13] = 5.0E-07  ;
   h2 [0] =  param [14] = -4.9E-02 ;
   h2 [1] =  param [15] = 1.3E-03  ;
   h2 [2] =  param [16] = -5.2E-02 ;
   h2 [3] =  param [17] = -5.1E-03 ;
   h3 [0] =  param [18] = 6.5E+02  ;
   h3 [1] =  param [19] = 9.8E+02  ;
   h3 [2] =  param [20] = 9.0E+02  ;
   h3 [3] =  param [21] = 1.0E+03  ;
   n1 [0] =  param [22] = 7.9E-07  ;
```

```
n1 [1] =  param [23] = -2.9E-04 ;
n1 [2] =  param [24] = 8.2E-07  ;
n1 [3] =  param [25] = 2.1E-07  ;
n2 [0] =  param [26] = -1.5E-02 ;
n2 [1] =  param [27] = 1.5E+00  ;
n2 [2] =  param [28] = -3.6E-02 ;
n2 [3] =  param [29] = 4.4E-02  ;
n3 [0] =  param [30] = 1.3E+03  ;
n3 [1] =  param [31] = 1.5E+02  ;
n3 [2] =  param [32] = 4.8E+03  ;
n3 [3] =  param [33] = 6.1E+01  ;
coef_prep ();
nu1 [ 0] =  param [40];
nu1 [ 1] =  param [41];
nu1 [ 2] =  param [42];
nu1 [ 3] =  param [43];
nu1 [ 4] =  param [44];
nu1 [ 5] =  param [45];
nu1 [ 6] =  param [46];
nu1 [ 7] =  param [47];
nu1 [ 8] =  param [48];
nu1 [ 9] =  param [49];
nu1 [10] =  param [50];
nu1 [11] =  param [51];
nu2 [ 0] =  param [53];
nu2 [ 1] =  param [54];
nu2 [ 2] =  param [55];
nu2 [ 3] =  param [56];
nu2 [ 4] =  param [57];
nu2 [ 5] =  param [58];
nu2 [ 6] =  param [59];
nu2 [ 7] =  param [60];
nu2 [ 8] =  param [61];
nu2 [ 9] =  param [62];
nu2 [10] =  param [63];
nu2 [11] =  param [64];
h1 [0] =  param [10];
h1 [1] =  param [11];
h1 [2] =  param [12];
h1 [3] =  param [13];
h2 [0] =  param [14];
h2 [1] =  param [15];
h2 [2] =  param [16];
h2 [3] =  param [17];
h3 [0] =  param [18];
h3 [1] =  param [19];
h3 [2] =  param [20];
h3 [3] =  param [21];
n1 [0] =  param [22];
n1 [1] =  param [23];
n1 [2] =  param [24];
n1 [3] =  param [25];
n2 [0] =  param [26];
n2 [1] =  param [27];
n2 [2] =  param [28];
n2 [3] =  param [29];
n3 [0] =  param [30];
n3 [1] =  param [31];
```

```
    n3 [2] =  param [32];
    n3 [3] =  param [33];
    inf2 [ 0] =  param [70];
    inf2 [ 1] =  param [71];
    inf2 [ 2] =  param [72];
    inf2 [ 3] =  param [73];
    inf2 [ 4] =  param [74];
    inf2 [ 5] =  param [75];
    inf2 [ 6] =  param [76];
    inf2 [ 7] =  param [77];
    inf2 [ 8] =  param [78];
    inf2 [ 9] =  param [79];
    inf2 [10] =  param [80];
    inf2 [11] =  param [81];
    inf4 [ 0] =  param [90];
    inf4 [ 1] =  param [91];
    inf4 [ 2] =  param [92];
    inf4 [ 3] =  param [93];
    inf4 [ 4] =  param [94];
    inf4 [ 5] =  param [95];
    inf4 [ 6] =  param [96];
    inf4 [ 7] =  param [97];
    inf4 [ 8] =  param [98];
    inf4 [ 9] =  param [99];
    inf4 [10] =  param [100];
    inf4 [11] =  param [101];
      free ((char*) vol1);
      free ((char*) vol2);
      free ((char*) vol3);
      free ((char*) vol4);
      vol1 = param + 40;
      vol2 = param + 40 + MONTH_NUMBER;
      vol3 = param + 40 + MONTH_NUMBER + MONTH_NUMBER;
      vol4 = param + 40 + MONTH_NUMBER + MONTH_NUMBER +
      MONTH_NUMBER;
      free ((char*) lev1);
      free ((char*) lev2);
      free ((char*) lev3);
      free ((char*) lev4);
      lev1 = param + 130;
      lev2 = param + 130 + MONTH_NUMBER;
      lev3 = param + 130 + MONTH_NUMBER + MONTH_NUMBER;
      lev4 = param + 130 + MONTH_NUMBER + MONTH_NUMBER +
      MONTH_NUMBER;
      free ((char*) dev_ener1);
      free ((char*) dev_ener2);
      free ((char*) dev_ener3);
      dev_ener1 = param + 180;
      dev_ener2 = param + 180 + MONTH_NUMBER;
      dev_ener3 = param + 180 + MONTH_NUMBER + MONTH_NUMBER;
}
/*          FUNCTIONS                */
/* general scheme of calculations */
void calc_manth_data (x)
double *x;
{
   int i;
   for (i = 0; i < MONTH_NUMBER; i++) {
```

```
      inf1 [i] = Inf1 (x, i);
      inf3 [i] = Inf3 (x, i);
      vol1 [i] = Volume1 (x, i);
      vol2 [i] = Volume2 (x, i);
      vol3 [i] = Volume3 (x, i);
      vol4 [i] = Volume4 (x, i);
      lev1 [i] = Level (vol1 [i], 0);
      lev2 [i] = Level (vol2 [i], 1);
      lev3 [i] = Level (vol3 [i], 2);
      lev4 [i] = Level (vol4 [i], 3);
      dev_ener1 [i] = demand1 - (lev1 [i] - t_lev1) * x [i];
      dev_ener2 [i] = demand2 - (lev2 [i] - t_lev2) * x [i+12];
      dev_ener3 [i] = demand3 - (lev3 [i] - t_lev3) * x [i+24];
   }
   demo ("user_par", NORMAL);
}
/* addition functions */
double Inf1 (x, i)
double *x;
int i;
{
     if (i == 0) {
     return (x [23] + x [35]);
   } else {
      return (x [i+12] + x [i+24]);
   }
}
double Inf3 (x, i)
double *x;
int i;
{
     if (i == 0) {
      return (x [47]);
   } else {
      return (x [i+36]);
   }
}
double Area (vol, j)
double vol;
int j;
{
   double a;
   a = n1 [j] * vol * vol + n2 [j] * vol + n3 [j];
   if (a < EPS) {
      a = EPS;
   }
   return (a);
}
double Level (vol, j)
double vol;
int j;
{
   double a;
   a = h1 [j] * vol * vol + h2 [j] * vol + h3 [j];
   if (a < EPS) {
      a = EPS;
   }
   return (a);
```

```
}
double Volume1 (x, i)
double *x;
int  i;
{
   double v;
   if (i == 0) {
      v = ini1        - nu1 [i] * Area (ini1,       0) + inf1 [i]
      - x [i];
   } else {
      v = vol1 [i-1] - nu1 [i] * Area (vol1 [i-1], 0) + inf1 [i]
      - x [i];
   }
   if (v < EPS) {
      v = EPS;
   }
   return (v);
}
double Volume2 (x, i)
double *x;
int  i;
{
   double v;
   if (i == 0) {
      v = ini2        - nu2 [i] * Area (ini2,       1) + inf2 [i]
      - x [i+12];
   } else {
      v = vol2 [i-1] - nu2 [i] * Area (vol2 [i-1], 1) + inf2 [i]
      - x [i+12];
   }
   if (v < EPS) {
      v = EPS;
   }
   return (v);
}
double Volume3 (x, i)
double *x;
int  i;
{
   double v;
   if (i == 0) {
      v = ini3        - nu2 [i] * Area (ini3,       2) + inf3 [i]
      - x [i+24];
   } else {
      v = vol3 [i-1] - nu2 [i] * Area (vol3 [i-1], 2) + inf3 [i]
      - x [i+24];
   }
   if (v < EPS) {
      v = EPS;
   }
   return (v);
}
double Volume4 (x, i)
double *x;
int  i;
{
   double v;
   if (i == 0) {
```

```
      v = ini4        - nu2 [i] * Area (ini4,       3) + inf4 [i]
      - x [i+36];
   } else {
      v = vol4 [i-1] - nu2 [i] * Area (vol4 [i-1], 3) + inf4 [i]
     - x [i+36];
   }
   if (v < EPS) {
      v = EPS;
   }
   return (v);
}
/* task definition functions */
double Energy1 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   calc_manth_data (x);
   sum = 0.0;
   p = (int) pe1;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = dev_ener1 [i] - perm_dev1;
      if (a > 0) {
         a = 0.01 * a;
         sum += i_pow_with_0 (a, p);
      }
   }
   return (sum);
}
double Energy2 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
   p = (int) pe2;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = dev_ener2 [i] - perm_dev2;
      if (a > 0) {
         a = 0.01 * a;
         sum += i_pow_with_0 (a, p);
      }
   }
   return (sum);
}
double Energy3 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
```

```
   p = (int) pe3;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = dev_ener3 [i] - perm_dev3;
      if (a > 0) {
         a = 0.01 * a;
         sum += i_pow_with_0 (a, p);
      }
   }
   return (sum);
}
double Dvol_1_11 (x)
double *x;
{
   double a;
   a = vol1 [11] - ini1;
   return (a * a);
}
double Dvol_2_11 (x)
double *x;
{
   double a;
   a = vol2 [11] - ini2;
   return (a * a);
}
double Dvol_3_11 (x)
double *x;
{
   double a;
   a = vol3 [11] - ini3;
   return (a * a);
}
double Dvol_4_11 (x)
double *x;
{
   double a;
   a = vol4 [11] - ini4;
   return (a * a);
}
double Dlev1 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
   p = (int) pv1;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = lev1 [i] - ref1;
      if (a < 0) {
         a = -a;
      }
      sum += i_pow_with_0 (a, p);
   }
   return (sum);
```

```
}
double Dlev2 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
   p = (int) pv2;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = lev2 [i] - ref2;
      if (a < 0) {
         a = -a;
      }
      sum += i_pow_with_0 (a, p);
   }
   return (sum);
}
double Dlev3 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
   p = (int) pv3;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = lev3 [i] - ref3;
      if (a < 0) {
         a = -a;
      }
      sum += i_pow_with_0 (a, p);
   }
   return (sum);
}
double Dlev4 (x)
double *x;
{
   int i, p;
   double a;
   double sum;
   sum = 0.0;
   p = (int) pv4;
   if (p < 2)  p = 2;
   if (p > 6)  p = 6;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = lev4 [i] - ref4;
      if (a < 0) {
         a = -a;
      }
      sum += i_pow_with_0 (a, p);
   }
   return (sum);
}
```

```
double Rlev1 (x)
double *x;
{
   int i;
   double a;
   double sum;
   sum = 0.0;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = min_lev1 - lev1 [i];
      if (a > 0)  sum += a * a;
   }
   return (sum);
}
double Rlev2 (x)
double *x;
{
   int i;
   double a;
   double sum;
   sum = 0.0;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = min_lev2 - lev2 [i];
      if (a > 0)  sum += a * a;
   }
   return (sum);
}
double Rlev3 (x)
double *x;
{
   int i;
   double a;
   double sum;
   sum = 0.0;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = min_lev3 - lev3 [i];
      if (a > 0)  sum += a * a;
   }
   return (sum);
}
double Rlev4 (x)
double *x;
{
   int i;
   double a;
   double sum;
   sum = 0.0;
   for (i = 0; i < MONTH_NUMBER; i++) {
      a = min_lev4 - lev4 [i];
      if (a > 0)  sum += a * a;
   }
   return (sum);
}
coef_prep ()
{
   param [10] = h1 [0];
   param [11] = h1 [1];
   param [12] = h1 [2];
   param [13] = h1 [3];
```

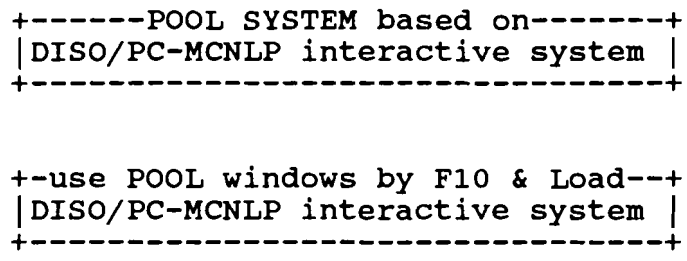
```
param [14] = h2 [0];
param [15] = h2 [1];
param [16] = h2 [2];
param [17] = h2 [3];
param [18] = h3 [0];
param [19] = h3 [1];
param [20] = h3 [2];
param [21] = h3 [3];
param [22] = n1 [0];
param [23] = n1 [1];
param [24] = n1 [2];
param [25] = n1 [3];
param [26] = n2 [0];
param [27] = n2 [1];
param [28] = n2 [2];
param [29] = n2 [3];
param [30] = n3 [0];
param [31] = n3 [1];
param [32] = n3 [2];
param [33] = n3 [3];
/*  October is the 1-st month [0]  */
/*  nu(i) is evaporation in m/month !!!  */
/*  nu1 - Cabora Bassa ,   nu2 - Itezhitezhi, Kafu, Kariba  */
param [40] = nu1 [0 ] =   0.17109830;
param [41] = nu1 [1 ] =   0.12423000;
param [42] = nu1 [2 ] =  -0.00436535;
param [43] = nu1 [3 ] =  -0.00949065;
param [44] = nu1 [4 ] =  -0.03052280;
param [45] = nu1 [5 ] =   0.04037027;
param [46] = nu1 [6 ] =   0.09891526;
param [47] = nu1 [7 ] =   0.11193680;
param [48] = nu1 [8 ] =   0.10114110;
param [49] = nu1 [9 ] =   0.08661708;
param [50] = nu1 [10] =   0.11285150;
param [51] = nu1 [11] =   0.14711930;
param [53] = nu2 [0 ] =  0.1672;
param [54] = nu2 [1 ] =  0.1158;
param [55] = nu2 [2 ] = -0.0395;
param [56] = nu2 [3 ] = -0.0368;
param [57] = nu2 [4 ] = -0.0669;
param [58] = nu2 [5 ] =  0.0074;
param [59] = nu2 [6 ] =  0.0910;
param [60] = nu2 [7 ] =  0.1113;
param [61] = nu2 [8 ] =  0.0994;
param [62] = nu2 [9 ] =  0.0865;
param [63] = nu2 [10] =  0.1113;
param [64] = nu2 [11] =  0.1464;
/*  inf2[ ] is for Kariba  in 10**6  m**3/month  */
param [70] = inf2 [0 ] =  873.5334;
param [71] = inf2 [1 ] =  929.8478;
param [72] = inf2 [2 ] =  2406.826;
param [73] = inf2 [3 ] =  4050.347;
param [74] = inf2 [4 ] =  5852.717;
param [75] = inf2 [5 ] =  8536.369;
param [76] = inf2 [6 ] =  9760.369;
param [77] = inf2 [7 ] =  7733.260;
param [78] = inf2 [8 ] =  4759.413;
param [79] = inf2 [9 ] =  2700.913;
```

```
    param [80] = inf2 [10] =  1650.500;
    param [81] = inf2 [11] =  1133.347;
    /*  inf4[ ] is for Itezhitezhi   in 10**6  m**3/month  */
    param [90] = inf4 [0 ] =  101.6857;
    param [91] = inf4 [1 ] =   99.2286;
    param [92] = inf4 [2 ] =  316.3714;
    param [93] = inf4 [3 ] =  916.9142;
    param [94] = inf4 [4 ] =  1765.114;
    param [95] = inf4 [5 ] =  2483.200;
    param [96] = inf4 [6 ] =  1668.171;
    param [97] = inf4 [7 ] =  935.0285;
    param [98] = inf4 [8 ] =  482.0285;
    param [99] = inf4 [9 ] =  300.1714;
    param [100] = inf4 [10] =  215.9142;
    param [101] = inf4 [11] =  148.7714;
    param [ 0]  = 2;   /* pv1 */
    param [ 1]  = 2;   /* pv2 */
    param [ 2]  = 2;   /* pv3 */
    param [ 3]  = 2;   /* pv4 */
    param [ 4]  = 2;   /* pe1 */
    param [ 5]  = 2;   /* pe2 */
    param [ 6]  = 2;   /* pe3 */
    /* 1 - Cabora Bassa, 2 - Kariba, 3 - Kasaka, 4 -Itezhitezhi */
    /*  ini[i] = Volume[i] (6) ;  ref[i] = Level[i] (6)  */
    param [105] = 39320.0;   /* ini1 */
    param [106] = 45780.0;   /* ini2 */
    param [109] = 1125.0;    /* ini3 */
    param [110] = 8000.0;   /* ini4 */
    /*  ref: 1 - Cabora Basa , 2 - Kariba, 3 - Kafue-Kasaka, 4 -
    Itezhi  */
    param [111] = 321.0;   /* ref1 */
    param [112] = 485.5;   /* ref2 */
    param [113] = 976.8;   /* ref3 */
    param [114] = 1035.0;  /* ref4 */
    /*  Demands: 1 - for Cabora Bassa, 2 - for Kariba, 3 - for
   Kafue-Kasaka  */
    /* Demands in 10**6 (m**3/month * m)
    param [115] =  400000.0;   /* demand1 */
    param [116] =  300000.0;   /* demand2 */
    param [117] =  200000.0;   /* demand3 */
    /*  MinLevel(j) in  meters above see level  */
    /*  MinLevel is Level when  Volume = 0  */
    /*  MinLevel1 - Cabora Bassa ,  MinLevel2 - Kariba  */
    /*  MinLevel3 - Kasaka-Kafue , MunLevel4 - Itezhitezhi  */
    param [118] = 295.0;   /* MinLevel1 */
    param [119] = 475.5;   /* MinLevel2 */
    param [120] = 900.0;   /* MinLevel3 */
    param [121] = 990.0;   /* MinLevel4 */
    /*  t_lev is Tailrace Level in  m.a.s.l.  */
    /*  1 - Cabora Bassa ,  2 - Kariba ,  3 - Kafue-Kasaka  */
    param [122] = 205.00;   /*  t_lev1  */
    param [123] = 377.95;   /*  t_lev2  */
    param [124] = 580.96;   /*  t_lev3  */
    param [220] = 0.0;      /* perm_dev1 */
    param [221] = 0.0;      /* perm_dev2 */
    param [222] = 0.0;      /* perm_dev3 */
}
```

Appendix 3

DISO/PC - MCNLP and POOL systems windows.

File Field Object
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help



VECTOR [64] OF STRING names [14]

Window (file) BEGIN.FM .

File Field Object one step run stop
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

RUN by point	IMPROVING CRITERIA	RUN in feasible region

increments

4	-4	4	-4	4	-4

priorities

1	1	1	0.1	0.1	0.1

reference point

0	0	0	0	0	0

functions: goals & restrict.

2.99366e+12	1.8e+07	2.68507e+07	2.84527e+08	2754720	623410

x

1220.18	4551.65	3959.42	2700.76	8133.55	1e-05

problem parameters				method	criter
2	2	2	2	5	113962

Window METHODS.FW .

File Field Object RUN
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

```
+--Deviations of Energy (1,2,3-d res.)--+
|2.99366e+12  1.8e+07      2.68507e+07  |
+---------------------------------------+
+---------------PENALTIES---------------+
|1            1            1           |
+---------------------------------------+

+----------------Deviations of levels----------------+
|2.84527e+08  2754720      623410       16084.7      |
+---------------------------------------------------+
+---------------------PENALTIES---------------------+
|0.1          0.1          0.1          0.1         |
+---------------------------------------------------+

+--------------Breaks of minimal levels--------------+
|8.14457      8.07073e+09  2.09581e+09  5053        |
+-------------------------------------------------+
+--------Breaks of init. value in last month---------+
|8.07073e+09  2.09581e+09  5053         6.4e+07     |
+-------------------------------------------------+
SCALAR OF STRING run_point
```

Window CRITERIA.FM .

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

```
+---------------Dif. energies: 1-st res.: Oct - Mar.---------------+  *Pen_E_
|6.91569e+0 6.53033e+0 6.23155e+0 6.33649e+0 4.36864e+0 6.94066e+0 |  *2
+-------------------------...: Apr - Sep.--------------------------+*M*------
|3.98401e+0 3.99362e+0 2088770    4.30163e+0 2.35542e+0 2.5954e+07 |*0
+-----------------------------------------------------------------+*--------
+---------------Dif. energies: 2-nd res.: Oct - Mar.---------------+  *Pen_E_
|-2.23373e+ -8834030   -1919510   -1549050   -1930190   300000     |  *2
+-------------------------...: Apr - Sep.--------------------------+*Max_dif_
|-1998350   -7638470   -23049.3   -5102940   300000     -7100150   |*0
+-----------------------------------------------------------------+*--------
+---------------Dif. energies: 3-d res.: Oct - Mar.----------------+  *Pen_E_
|200000     478024     -1115220   -551106    -4163460   -994144    |  *2
+-------------------------...: Apr - Sep.--------------------------+*Max_difv
|-1321450   -5002300   -654339    -447636    -10791.7   -1138920   |*0
+-----------------------------------------------------------------+*--------
```

SCALAR OF STRING run_one

Window DIF_ENER.FW .

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

```
+---------------Levels of the 1-st res.: Oct - Mar.----------------+ *Pen_L_1·
|108.911    6790.91    10790.3    19966.4    13909.3    17609      | *2
+-------------------------...: Apr - Sep.--------------------------++*-------·
|17349.4    16556.4    11057.9    6503.19    4398.78    1640.1     ||321
+-----------------------------------------------------------------++--------·
+---------------Levels of the 2-dn res.: Oct - Mar.----------------+ *Pen_L_2·
|980        980        980        980        977.997    941.285    | *4
+-------------------------...: Apr - Sep.--------------------------++-Ref_L_2·
|849.868    827.704    759.714    715.299    697.14     676.554    ||485.5
+-----------------------------------------------------------------++--------·
+---------------Levels of the 3-d res.: Oct - Mar.----------------+ *Pen_L_3·
|900        900        696.307    900        576.567    689.575    | *4
+-------------------------...: Apr - Sep.--------------------------++-Ref_L_3·
|900        900        900        900        900        900        ||976.8
+-----------------------------------------------------------------++--------·
+---------------Levels of the 4-th. res.: Oct - Mar.---------------+ *Pen_L-4·
|987.241    989.312    1000       996.018    1000       1000       | *6
+-------------------------...: Apr - Sep.--------------------------++-Ref_L_4·
|1000       996.375    994.836    1000       1000       1000       ||1035
+-----------------------------------------------------------------++--------·
```

VECTOR [256] OF DOUBLE user_par [0]

Window LEVELS.WW .

Appendix 4

The set of Pareto optimal solutions (values of outflows from all reservoirs are presented in million cubic meters per month).

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

Volumes of the 1-st res.: Oct - Mar.					
45958.8	122422	146151	187607	161717	178071
...: Apr - Sep.					
176983	173608	147568	120475	104698	76477
Volumes of the 2-nd res.: Oct - Mar.					
1.11e-100	1.11e-100	1.11e-100	1.11e-100	5862.75	14307.1
...: Apr - Sep.					
24004.3	25758.6	30466.2	33155.7	34190	35323.3
Volumes of the 3-d res.: Oct - Mar.					
1.11e-100	1.11e-100	4183.01	1.11e-100	6954.69	4331.7
...: Apr - Sep.					
1.11e-100	1.11e-100	1.11e-100	1.11e-100	1.11e-100	1.11e-100
Volumes of the 4-th res.: Oct - Mar.					
5801.81	2947.42	1.11e-100	851.926	1.11e-100	1.11e-100
...: Apr - Sep.					
1.11e-100	768.768	1139.92	1.11e-100	1.11e-100	1.11e-100

VECTOR [256] OF DOUBLE user_par [87]

Window VOLUMES.FM .

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

Outflows from the 1-st res.: Oct - Mar.					
304.954	11736.5	35109.9	8112.79	30130.3	16208.5
...: Apr - Sep.					
18718	18915.6	43778.9	42225.1	19741.4	34261.3
Outflows from the 2-nd res.: Oct - Mar.					
61823.8	49925.6	47341.8	28204.9	1e-05	91.9775
...: Apr - Sep.					
63.2234	5978.9	51.8983	11.393	616.203	1e-05
Outflows from the 3-d res.: Oct - Mar.					
30192.4	38557.2	11447.5	21212.8	3437.57	33258.7
...: Apr - Sep.					
19909.1	12179.9	19963.1	16531.7	4585.58	7274.27
Outflows from the 4-th res.: Oct - Mar.					
2228.58	2916.17	15441	67.2334	10071.1	30669.7
...: Apr - Sep.					
3252.75	159.471	101.439	6251.05	904.62	281.572

VECTOR [48] OF DOUBLE x [47]

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

Outflows from the 1-st res.: Oct - Mar.					
359.403	10412.1	552.923	6852.86	14736.9	1e-05
Apr - Sep.					
1e-05	2073.56	1073.3	364.519	11684.1	9921.78
Outflows from the 2-nd res.: Oct - Mar.					
39229.1	587.694	16109.4	3105.52	1608.96	13506.1
Apr - Sep.					
162.543	3589.28	204.561	2.69037	47.7439	12545.5
Outflows from the 3-d res.: Oct - Mar.					
346.321	7457.52	12821	10288	12159.9	6630.63
Apr - Sep.					
144.296	701.556	1940.23	1361.23	1956.59	19599.4
Outflows from the 4-th res.: Oct - Mar.					
1e-05	3744.98	4600.78	2132.91	4435.77	1e-05
Apr - Sep.					
1e-05	1e-05	165.693	1e-05	63.577	1e-05

VECTOR [48] OF DOUBLE x [47]

File Field Object GET POINT
F1 - help, F10 - menu, Arrows - position move, 'q' - exit, others - see Help

Outflows from the 1-st res.: Oct - Mar.					
1596.68	39102.9	33642.1	11372.3	21279.1	22709.6
...: Apr - Sep.					
17567.3	24195.6	50493.9	22710.1	27278	35587.3
Outflows from the 2-nd res.: Oct - Mar.					
34722.2	64897.8	29374.2	35262.1	4213.94	19468.8
...: Apr - Sep.					
6584.16	50889.5	14093.2	1e-05	6.94867	1e-05
Outflows from the 3-d res.: Oct - Mar.					
1350.55	14839	3512.37	16646.5	15342.8	17358.7
...: Apr - Sep.					
25999.9	1e-05	12774.3	7923.26	7659.4	2489.78
Outflows from the 4-th res.: Oct - Mar.					
17558.6	18369.6	24325.7	2.95074	7382.18	51859.5
...: Apr - Sep.					
1105.91	901.689	15723.8	2792.56	1e-05	372.933

VECTOR [48] OF DOUBLE x [47]