Interactive Compromise Stability of Multi-objective Nonlinear Programming problems

Kassem, M.[(1)*], El-Benna, A.[(1)], and El-Badry, N.[(2)]

[(1)] Mathematics department, Faculty of Science, Tanta University

[(2)] Mathematics department, Faculty of Science, Damietta Branch, Mansoura University

Abstract: This paper presents a solution method for multi-objective nonlinear programming (MONLP) problems and stability of this solution. The method, offers a practical solution to MONLP problems by deriving the compromise weights and combining judgment with an automatic optimization technique in fuzzy decision making. This is achieved by using the method and algorithm of compromise programming and the method of compromise weights and we obtain the stability for the solution in each step of the algorithm. A numerical example illustrates various aspects of the results developed in this paper. A maple procedure for this algorithm is introduced.

[Kassem, M., El-Benna, A., and El-Badry, N., Interactive Compromise Stability of Multi-objective Nonlinear Programming problems. Journal of American Science 2011;7(1):222-229]. (ISSN: 1545-1003). http://www.jofamericanscience.org.

Keywords: MONLP; Stability; Interactive decision making; Compromise weights; Membership functions**.**

1. Introduction

Most decision problems have multiple objectives conflicting among themselves. The solution for such problems can only be obtained by trying to get compromises based on information provided by the decision maker (**DM**). Several methods have been developed to solve multiobjective decision making (**MODM**) problems, see [10]. In [5,8] some of these methods are based on prior information required from the **DM**. This information may be in the from the desired achievement levels of the objective functions and the ranking of the levels indicating their importance, such as in goal programming . It may also be in the form of weights showing the importance of the objectives. The disadvantages with these method is that the **DM** cannot easily provided this prior information since he has no idea about the solution process of the problem. Other methods, called interactive methods, have been developed in order to overcome this disadvantage. There are two categories of interactive methods. Interactive methods of the first type require the **DM** to provide some trade-offs among the attained values of the objective functions in order to determine the new solution [4]. The interactive methods of the second type require the **DM** to provide some preference information by comparing the various efficient solutions in the space of the objective functions or the decision variables . The quantity and complexity of the information required from the **DM** in such methods are important factors affecting the chances of reaching the best compromise solution. In [3, 7] an interactive linear multiple objective method, called interactive compromise programming (**ICP**) were introduced. The notions of the solvability set, stability set of the first kind and stability set of the second kind, and analysed these concepts for parametric convex nonlinear programming problems were introduced in [6, 9].

This paper presents an interactive stability compromise programming method for solving **MONLP** problems by using the compromise weights from the pay-off table and fuzzy membership function for each objective function. An illustrative example is given to clarify the obtained results.

2. Problem Formulation

Let us consider the **MONLP** problem:

$$(\text{MONLP}): \quad \max\left(f_1(x), f_2(x), ..., f_m(x)\right)$$

subject to

$$x \in X = \left\{x \in R^n \mid g_j(x) \le 0, j = 1,2,...,k\right\}$$

where $f_i(x)$, $i = 1,...,m$, and $g_j(x)$, $j = 1,...,k$, are convex real valued functions which belong to class $C^{(1)}$.

The corresponding scalarization problem is

$$(\mathbf{MONLP})_\lambda \qquad \max \sum_{i=1}^{m} \lambda_i f_i(x)$$

$$\text{subject to } x \in X,$$

where $\lambda = (\lambda_1,...,\lambda_m) \neq 0, \lambda_i \ge 0, i = 1,2,...,m$, and $\sum_{i=1}^{m} \lambda_i = 1.$

Let $f_i(x)$ be the *i*th objective function and $f_i^U(x)$ be the maximum possible values of $f_i(x)$ and $f_i^L(x)$ are the minimum possible values of $f_i(x)$ found under the constraints, respectively . To obtain the compromise solution of the MONLP problem , find the solution which has a minimum distance with respect to the ideal solution $f_i^U(x)$. This requires normalization of the objective functions and appropriate choice for the distance measure. The solution found in this way is a reduced set of all efficient solution . The set of compromise solution may be large, and also the choice of weights by the DM may be difficult.

these difficulties could be reduced by combining the basic ideas for the methods of compromise programming and compromise weights .

3. Compromise Weights

Here, we introduce a method based on the following two main ideas:

First, the DM could state his preference among some alternative solutions more easily if the values of objective functions were measured on the some scale varying between zero and one . This could be done by employing "the membership function for the objective functions" concept in the compromise programming. In order to elicit a membership function $\mu_{f_i}(x)$ from the DM for each of the objective functions $f_i(x)$ in MONLP problems, we first calculate the individual minimum f_i^L and maximum f_i^U of each objective function $f_i(x)$ under the given constraints. By taking account of the calculated individual minimum and maximum of each objective function together with the rate of increase of membership of satisfaction**,** **DM** must determine his subjective membership function $\mu_{f_i}(x)$ which is a strictly monotone increasing function with respect to $f_i(x)$.

Here, it is assumed that

$$\mu_{f_i}(x)=0 \text{ or } \to 0 \text{ if } f_i(x)\le f_i^L \text{ and}$$

$$\mu_{f_i}(x)=1 \text{ or } \to 1 \text{ if } f_i(x)\ge f_i^U,$$

where f_i^a represents the value of $f_i(x)$ such that the value of membership function $\mu_{f_i}(x)$ is a.

In this method , the following definition of the membership functions is used for scaling:

$$\mu_{f_i}(x)=\frac{f_i(x)-f_i^L}{f_i^U-f_i^L}, \qquad (1)$$

where $f_i(x)$ are the objective functions, f_i^U are the maximum possible values of $f_i(x), i=1,2,...,m$ and f_i^L are the minimum possible values of $f_i(x)$ satisfying the constraints $x\in X$ .The $\mu_{f_i}(x)$ are defined as the membership functions of $f_i(x)$ to the possible value $f_i(x)$.

The corresponding scalarization problem is:

$$\max \mu_{f^{m+1}}(x)=\sum_{i=1}^{m}\lambda_i\mu_{f_i}(x) \qquad (2)$$

subject to $x\in X$.

The second main idea , one of the main drawbacks of the interactive methods is the difficulty of getting the weights of the objective functions from the **DM** even if values of objective functions are presented to him on the same scale.

In this method , the compromise weights of objective functions can be obtained by constructing the pay-off table displaying values of objective functions at $x^1,...,x^m$, where x^i solves $\max f_i(x)$, $i=1,...,m$, subject to $x\in X$. A pay-off table is

	f_1	f_2	$\cdots$	f_i	$\cdots$	f_m
f_1	f_1^*	f_1^2	$\cdots$	f_1^j	$\cdots$	f_1^m
f_2	f_2^1	f_2^*	$\cdots$	f_2^j	$\cdots$	f_2^m
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
f_i	f_i^1	f_i^2	$\cdots$	f_i^*	$\cdots$	f_i^m
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
f_m	f_m^1	f_m^2	$\cdots$	f_m^j	$\cdots$	f_m^*

where $f_i^*=f_i(x^i)$ and $f_i^j=f_i(x^j)$ for each $i=1,...,m$, $j=1,...,k$ and $i\neq j$ the compromise weights λ_i, $i=1,2,...,m$ can be obtained from the pay-off matrix by the formula,

$$\lambda_i=\frac{e^{\alpha a_i}}{\sum_{i=1}^{m}e^{\alpha a_i}},\ i=1,2,...,m,$$

$$\alpha=\frac{1}{a_m-a_{m-1}}\ln\left|\sum_{i=1}^{m}\frac{a_i}{a_m}\right|, a_i=\hat{f_i}-f_i^*, i=1,2,...,m. \qquad (3)$$

where $\hat{f_i}=\max f_i(x^i)$ is the maximum entry in row i.

4. Stability set of the first kind [7]

Definition 1. The solvability set of problem $(\mathbf{MONLP})_\lambda$ is defined by

$$B=\left\{\lambda\in R_+^m \mid \max_{x\in X}\sum_{i=1}^{m}\lambda_i f_i(x) \text{exists}\right\},$$

where R_+^m is the nonnegative orthant of the vector parameter λ.

Definition 2. Suppose that $B\neq\varnothing$ with a corresponding optimal point $\bar{x}$, then the stability set of the first kind of problem $(\text{MONLP})_\lambda$ corresponding to $\bar{x}$ is defined by

$$S(\bar{x})=\left\{\lambda\in B \mid \sum_{i=1}^{m}\lambda_i f_i(\bar{x})=\max_{x\in X}\sum_{i=1}^{m}\lambda_i f_i(x)\right\}.$$

It is clear that the stability set of the first kind is the set of all parameters corresponding to an optimal solution of the scalarizing problem.

Let $\bar{\lambda}\in S(\bar{x})$ then there exist $\bar{u}\in R^k$ such that $(\bar{x},\bar{u})$ solves the following Kuhn-Tucker problem:

$$\sum_{i=1}^{m}\bar{\lambda}_i\frac{\partial f_i(\bar{x})}{\partial x_\alpha}+\sum_{j\notin J}\bar{u}_j\frac{\partial g_j(\bar{x})}{\partial x_\alpha}=0,\ \alpha=1,2,...,n,$$

$g_j(\bar{x}) \le 0, \bar{u}_j g_j(\bar{x}) = 0,\ j = 1,2,...,k,$

$\bar{u}_j = 0,\ j \in J \subset \{1,2,...,k\}, \bar{u}_j \ge 0,$

$j \in \{1,2,...,k\} - J,$

that means, we order the function $g_j(x)$, j=1,2,…,k, in such a way that

$j \in \{1,2,...,s\}$ if $g_j(\bar{x}) = 0,$

$j \in \{s+1,...,k\}$ if $g_j(\bar{x}) < 0.$

Consider the system of equations

$$\sum_{i=1}^{m} \lambda_i \frac{\partial f_i(\bar{x})}{\partial x_\alpha} + \sum_{j=1}^{s} u_j \frac{\partial g_j(\bar{x})}{\partial x_\alpha} = 0, \quad \text{(I)}$$

$\alpha = 1,2,...,n.$

It represent n linear homogenous equations in $m+s$ unknowns λ_i, $i = 1,2,...,m$, and u_j, $j = 1,2,...,s$, which can be solved explicitly.

Suppose that $\lambda_i^* \ge 0,\ i = 1,2,...,m,$ and $u_j^* \ge 0,$ j=1,2,…,s, solve the above system of equations, then it is clear that $(\bar{x}, \bar{u})$ solves the Kuhn-Tucker problem , where $\bar{u}_j = u_j^*,\ j = 1,2,...,s,\ \bar{u}_j = 0,$ j=s+1,…,k, and hence $\lambda^* \in S(\bar{x})$.

Let us define the set

$$P(\lambda,u) = \left\{(\lambda,u) \in R_+^{m+s} \,\middle|\, (\lambda,u) \text{solves the system (I)}\right\},$$

where R_+^m and R_+^s are the nonnegative orthants of the R^m vector $\lambda-$space, and R^s vector $u-$space, respectively . Then

$$S(\bar{x}) = \left\{\lambda \in R_+^m \,\middle|\, (\lambda,u) \in P(\lambda,u)\right\}. \quad \text{(II)}$$

If $g_j(\bar{x}) < 0, j = 1,2,...,k$, then it is easy to see that $S(\bar{x})$ can be written in the following form:

$$S(\bar{x}) = \left\{\lambda \in R_+^m \mid \sum_{i=1}^{m} \lambda_i \frac{\partial f_i(\bar{x})}{\partial x_\alpha} = 0, \alpha = 1,2,...,n\right\}.$$

5. Interactive compromise algorithm

In this method, the solution process by solving 2m simple nonlinear programming problems to find the maximum and minimum possible values of m objective under the given constraints.

The compromise weights of the objective functions are determine from the Eq.(3) and employed in the problem (2) we have

$$\max \mu_{f^{m+1}}(x) = \sum_{i=1}^{m} \lambda_i \mu_{f_i}(x)$$

subject to $x \in X$

where $\mu_{f^{m+1}}(x)$ is the composite function of $\mu_{f_i}(x)$ and it determines the $(m+1)th$ solution .

The steps of the algorithm can be summarized as follows:

Step 1. Determine f_i^U, f_i^L for all i=1,…,m, as follows:

(i) $\max f_i(x)$

subject to $x \in X$,

The solutions of this problem are x^{iU} and f_i^U which are known as the "ideal solution".

(ii) $\min f_i(x)$

subject to $x \in X$,

The solution are x^{iL} and f_i^L which are known as the "anti -ideal solution".

Step 2. Determine the membership functions corresponding the solution x^{iU}, $i = 1,2,...,m$ as in the relation (1).

Construct the pay-off table

	f_1	f_2	⋯	f_i	⋯	f_m
f_1	f_1^*	f_1^2	⋯	f_1^j	⋯	f_1^m
f_2	f_2^1	f_2^*	⋯	f_2^j	⋯	f_2^m
⋮	⋮	⋮		⋮		⋮
f_i	f_i^1	f_i^2	⋯	f_i^*	⋯	f_i^m
⋮	⋮	⋮		⋮		⋮
f_m	f_m^1	f_m^2	⋯	f_m^j	⋯	f_m^*

where x^i solves

$\min f_i(x), i = 1,2,...,m$, $f_i^* = f_i(x^i)$

subject to $x \in X$,

$f_i^j = f_i(x^j)$ for each i=1,…,m, j=1,2,…,k, $i \ne j$ and construct fuzzy matrix.

$\mu_{f_i^j}$	x^1	x^2	...	x^m	f^u
f_1	$\mu_{f_1^1}$	$\mu_{f_1^2}$	…	$\mu_{f_1^m}$	f_1^U
f_2	$\mu_{f_2^1}$	$\mu_{f_2^2}$	…	$\mu_{f_2^m}$	f_2^U
⋮	⋮	⋮		⋮	⋮
f_m	$\mu_{f_m^1}$	$\mu_{f_m^2}$	…	$\mu_{f_m^m}$	f_m^U

Step 3: The compromise weights $\lambda_i, i = 1,2,...,m$ can be found from

$$\lambda_i = \frac{e^{\alpha a_i}}{\sum_{i=1}^{m} e^{\alpha a_i}}, i = 1,2,...,m ,$$

$$\alpha = \frac{1}{a_m - a_{m-1}} \ln \left| \sum_{i=1}^{m} \frac{a_i}{a_m} \right|, \ a_i = \hat{f}_{ii} - f_i^*, \ i = 1,2,...,m.$$

$\hat{f}_i = \max f_i(x^i)$ is the maximum entry in row i .

Step 4: By using this weights , we establish the new compromise solution x^{m+1}, from the problem (2).

Step 5. Determine the stability set of the first kind corresponding to this solution as in relations (I) and (II).

Step 6: Determine the membership objective

functions of the new solution of the problem in step 4, $\mu_{f^{m+1}}$ .Add this column to table of fuzzy in step 2.

Step 7: Ask the DM whether he prefers one solution strictly over all the other *m*-solutions if he does go to step 8 , otherwise ask him his least preferred solution among all the others. Then replace this preferred solution by the new found in step 6 and go to step 3.

Step 8: Stop.

6. Numerical example

Let us consider the following problem

$$\min f_1(x) = x_1 + x_2^2,$$

$$\min f_2(x) = (x_1 - 5)^2 + x_2,$$

$$\text{subject to } x_1^2 + x_2^2 \le 25,$$

$$x_1 \ge 0, \quad x_2 \ge 0.$$

The solution of this example will be obtained using a Maple program:

Step1.

(I) $\max f_1(x) = x_1 + x_2^2$,

subject to $x_1^2 + x_2^2 \le 25,$

$x_1 \ge 0, \quad x_2 \ge 0.$

solution $x^{1U} = (0.5, 4.97), \; f_1^U = 25.25$.

(II) $\max f_2(x) = (x_1 - 5)^2 + x_2^2,$

subject to $x_1^2 + x_2^2 \le 25,$

$x_1 \ge 0, \quad x_2 \ge 0.$

solution $x^{2U} = (0,5), \quad f_2^U = 30$.

(III) $\min f_1(x) = x_1 + x_2^2$

subject to $x_1^2 + x_2^2 \le 25,$

$x_1 \ge 0, \quad x_2 \ge 0.$

solution $x^{1L} = (0,0), \quad f_1^L = 0$.

(IV) $\min f_2(x) = (x_1 - 5)^2 + x_2^2,$

subject to $x_1^2 + x_2^2 \le 25,$

$x_1 \ge 0, \quad x_2 \ge 0.$

solution $x^{2L} = (5,0), \quad f_2^L = 0$.

Step 2.the corresponding pay- off table is

	f_1	f_2
f_1	0	5
f_2	25	0

where $f_1(x^2) = \left[x_1 + x_2^2\right]_{(5,0)} = 5,$

$$f_2(x^1) = \left[(x_1 - 5)^2 + x_2^2\right]_{(0,0)} = 25.$$

the corresponding fuzzy matrix is

$\mu_{f_i^j}$	x^1	x^2	f^u
f_1	0	0.1980	25.25
f_2	0.8333	0	30

Step 3. Substitute of pay- off table in relation (3) to obtain the corresponding compromise weights $\lambda_1 = 0.4545$ and $\lambda_2 = 0.54545$

Step 4. The new composite membership function is

$$\min \mu_{f^3}(x) = \min\left\{\frac{0.4545}{25.25}\left[x_1 + x_2^2\right] + \frac{0.54545}{30}\left[(x_1 - 5)^2 + x_2^2\right]\right\}$$

Subject to $x_1^2 + x_2^2 \le 25,$

$$x_1 \ge 0, \quad x_2 \ge 0.$$

The solution is

$x^3 = (4.504950495, 0), \; f_1(x^3) = 4.504950495,$

$f_2(x^3) = .2450740124$

Step 5. The set of all parameters which corresponds to this solution is defined by the stability set of the first kind in the following form:

$$S(x^3) = \{\lambda_1 - 0.990099010\,\lambda_2 + 9.009900990\,u_1 = 0,$$

$$\lambda_2 = 0, \; u_1 > 0\}$$

Step 6.

$$\mu_{f_1^3} = \frac{f_1(x^3) - f_1^I}{f_1^U - f_1^I} = \frac{4.504950495 - 0}{25.25 - 0} = 0.1784138810,$$

$$\mu_{f_2^3} = \frac{f_2(x^3) - f_2^I}{f_2^U - f_2^I} = \frac{0.2450740124 - 0}{30 - 0} = 0.00816913347.$$

Therefore, the new fuzzy matrix is

$\mu_{f_i^j}$	x^1	x^2	x^3	f^u
f_1	0	0.1980	0.1784138810	25.25
f_2	0.8333	0	0.00816913347	30

Step 7. Present the three solution to **DM** if he is certain that one of them is the best solution of the problem (not only preferred regarding the other two), stop. Else, ask DM whether he prefers one solution over the two solutions. Suppose that he would not, and his least preferred solution would be solution 2. This solution is then replaced by solution 3 return to Step 3.

The new pay-off table is

	f_1	f_2
f_1	4.504950495	5
f_2	0.2450740124	0

By using relation (3) we obtain the compromise weights

$\lambda_1 = 0.2487562190$ and $\lambda_2 = 0.7512437810$.

We note that these weights are out of the range of parameters which were defined in the above set $S(x^3)$ so we must have the next solution.

The new composite membership function is

$$\min \mu_{f^3}(x) = \min\left\{\frac{0.2487562190}{25.25}\left[x_1 + x_2^2\right] + \frac{0.7512437810}{30}\left[(x_1 - 5)^2 + x_2\right]\right\}$$

subject to $x_1^2 + x_2^2 \le 25,$

$$x_1 \geq 0, \quad x_2 \geq 0.$$

which solution is

$x^3 = (4.80329159, 0),\ f_1(x^3) = 4.803292,\ f_2(x^3) = 0.0386942.$

, the corresponding stability set of the first kind is

$$S(x^3) = \{\lambda_1\text{-}0.39356829\lambda_2 + 9.6064318\, u_1 = 0, \lambda_2 = 0, u_1 > 0\}$$

We have the corresponding membership function in the form

$\mu_{f_1^3} = 0.1902293698, \mu_{f_2^3} = 0.00128906658$.

Therefore the fuzzy matrix is

$\mu_{f_i^j}$	x^1	x^2	x^3	f^u
f_1	0.17841388	0.1980	01902293698	25.25
f_2	0.0081691334	0	0.001289806658	30

Suppose the DM would prefer the new solution over these solutions. Go to Step 8.

Step 8. Stop. The best compromise solution of this problem would be

$$\bar{x} = (4.80329158725595917, 0),$$
$$f = (4.803291587, 0.03869419974),$$
$$\mu_f = (0.1902293698, 0.001289806658).$$

7. Conclusion

An interactive stability compromise programming method , using a fuzzy approach and a pay-off table. In this method, no prior information is required from the **DM** and the compromise weights of the objective functions are determined from the pay-off table and fuzzy matrix. The method does not require significantly more data than pure nonlinear programming and the scale of multi-objective problem by using substituting the objective functions by the membership function and to obtain compromise weights by the grades of membership of the current vectors in each iteration in the "close ideal" fuzzy set.

The proposed algorithm programmed by using Maple program.

8. Reference

[1] Chankong, V. and Haimes, Y. Y., "Multiobjective Decision Making Theory and Methodology". Elsevier Science, New York, 1983.

[2] El-Sayed, H. M. "A Unified Interactive Approach For Solving Multiple-Objective Nonlinear Programming and Computer Code ", Proceedings of the first international conference on operations research and its applications, Cairo 1994 .

[3] Evren, R. "Interactive compromise programming", Journal of the Operational Research Society 38 (2) (1987) 163-172.

[4] Geoffrion, A. , Dyer, J. and Finbred, A. "An interactive approach for multi-criteria optimization with an application to the operation of an academic department", Management Science 19 (1972) 357-368.

[5] Ignizeo, J. "Goal Programming and Extensions", Heath, Lexington, MA, 1976.

[6] Kassem, M. "Interactive Stability of Multiobjective Nonlinear Programming Problems with Fuzzy Parameters in the Constraints", Fuzzy Sets and Systems 73 (1995) 235-243.

[7] Kassem, M. "Interactive Stability of Vector Optimization Problems", European Journal of Operational Research 134 (2001) 616-622.

[8] Lee, S. "Goal Programming for decision Analysis", Auerbach, Philadelphia, PA, 1972.

[9] Osman, M. and El-Benna, A. "stability of multiobjective nonlinear programming problems with fuzzy parameters", Mathematics and Computer Simulation 35 (1993) 321-326.

[10] Zeleny, M. "Multiple Criteria Decision Making", McGraw-Hill, New York, 1982.

9. Appendix

A maple program for solving multi-objective nonlinear programming (MONLP) problems and stability of this solution**.**

```
> restart:
with(Optimization):
with(Maplets[Elements]):
with(Groebner):
> maplo:=Maplet(["Enter The Type of The
Problem",
[Button("Minimize",Shutdown("Minimize")),
Button("Maximize",Shutdown("Maximize"))
]]):
d:=Maplets[Display](maplo):dm:=parse(d);
ma := Maplet([["Enter No of Vector Spaces",
TextField['TF']()],
[Button("ok",Shutdown(['TF']))]]):
n1:=Maplets[Display](ma):n:=parse(n1[1]);
q1:="a":i:=0:
while(q1="a") do
i:=i+1:
maplet := Maplet([["Enter an Objective
function ", TextField[TF1]()],
[Button['b']("ok",Shutdown( [TF1]))]]):
t[i]:=Maplets[Display](maplet);
maplet2 := Maplet([[Label("Enter Another
objective function?:")],
[Button['B1']("Ok",Shutdown("a"))],
[Button['B2']("No",Shutdown())]]):
q1:= Maplets[Display](maplet2):
end do:
> m:=i;
> for i from 1 to m do
```

```
f[i]:=parse(t[i][1]);
end do;
> q2:="a":i:=0:
while(q2="a")do
i:=i+1:
mapl1 := Maplet([["Enter Your Constraints",
TextField[TF1]()],
[Button['b']("ok",Shutdown( [TF1]))]]):
t1[i]:=Maplets[Display](mapl1);
mapl2 := Maplet([
[Label("Enter Another Constraint?: ")],
[Button['B1']("Ok",
Shutdown("a"))],[Button['B2']("No",
Shutdown())]]):
q2:= Maplets[Display](mapl2):
end do:
> k:=i;
> for i from 1 to k do
g[i]:=parse(t1[i][1]);
end do;
> for i from 1 to m do
for j from 1 to k do
> Q[i]:=Maximize(f[i], {g[j]},
assume=nonnegative); P[i]:=Minimize(f[i],
{g[j]}, assume=nonnegative);
> end do;Q[i];P[i];end do;
> for i from 1 to m do
for j from 2 to m+1 do
> R1[i,j]:=rhs(Q[i][2][j-1]);
> # Minimization.
> S1[i,j]:=rhs(P[i][2][j-1]);end do;
end do;
> nnn:=proc(R1,S1,Q,P,f,g,n,m,k,MU1)
local R,i,j,K,MUf,A,alpha,FN,MuF,mx,f1,
f2,f3,f4,f5,f6, maplet3:
global S,Mu,Lamda,Z,eq,ss,eq1,rr,su1,su2,lam,
alph,kk,UU,U:
for i from 1 to m do
> # Maximization.
R[i,1]:=Q[i][1];
> # Minimization.
> S[i,1]:=P[i][1];
for j from 2 to m+1 do
R[i,j]:=R1[i,j];
S[i,j]:=S1[i,j];
end do: end do:
> Z:=Matrix(1..m,1..m+1):
> for i from 1 to m do
for j from 1 to m do
f5[i]:=f[i]:f4[i]:=f[i]:
if i=j then
kk:=1:
while(kk<=m) do
f1[i]:=subs(x[kk]=S[i,kk+1],f5[i]):
f5[i]:=f1[i]:kk:=kk+1:
end do:
Z[i,i]:=f5[i]:
else
kk:=1:
while(kk<=m) do
f2[i]:=subs(x[kk]=S[j,kk+1],f4[i]):
f4[i]:=f2[i]:kk:=kk+1:
end do:
Z[i,j]:=f4[i]:
end if:
end do;
end do;
Z;
for i from 1 to m do
> MUf[i]:=(f[i]-S[i,1])/(R[i,1]-S[i,1]):
> end do;
Mu:=Matrix(1..m,1..m+2):
> for i from 1 to m do
> for j from 1 to m do
> Mu[i,j]:=(Z[i,j]-S[i,1])/(R[i,1]-S[i,1]);
 end do;
> Mu[i,m+1]:=R[i,1];
> end do: Mu;
> mx:=Array(1..m):
for i from 1 to m do
mx[i]:=Z[i,1];
for j from 1 to m do
if (mx[i]<Z[i,j]) then mx[i]:=Z[i,j]; end if;
end do;
end do;
mx;
Lamda:=Array(1..m):
A:=Array(1..m):for i from 1 to m do
> A[i]:=mx[i]-Z[i,i];
> end do;
> alpha:=ln(abs(add((A[i]/A[m]),i=1..m)))/
(A[m]-A[m-1]);
> for i from 1 to m do
> Lamda[i]:=exp(alpha*A[i])/
  add(exp(alpha*A[k1]),k1=1..m);
> end do;
> add(Lamda[i],i=1..m);
> FN:=add(Lamda[i]*MUf[i],i=1..m);
> for i from 1 to k do
MuF:=dm(FN, {g[k]}, assume=nonnegative);
end do:
> for j from 2 to n+1 do
S[m+1,j]:= rhs(MuF[2][j-1]); end do;
lam:=Vector(m,symbol=la):
```

```
UU:=Vector(k,symbol=U):
> for alph from 1 to n do
su1:=add(diff(f[i],x[alph])*lam[i],i=1..m);
su2:=add(diff(lhs(g[j]),x[alph])*UU[j],j=1..k);
> eq[alph]:=su1+su2;
end do:
> for i from 1 to m do
> j:=1:
f6[i]:=f[i]:
while(j<=m) do
f3[i]:=subs(x[j]=S[m+1,j+1],f6[i]);
f6[i]:=f3[i]:j:=j+1:
end do:
Z[i,m+1]:=f6[i];
> end do;Z;
> for i from 1 to m do
> Mu[i,m+2]:=Mu[i,m+1];
Mu[i,m+1]:=(Z[i,m+1]-S[i,1])/(R[i,1]-S[i,1]);
> end do:MU1:=print("Mu=",Mu);
end proc:
> nnn(R1,S1,Q,P,f,g,n,m,k,MU1);
RS:=Array(1..n):RS1:=Array(1..n):
for alph from 1 to n do
rr1:=eq[alph]:rr11:=eq[alph]: j:=1:
while(j<=m) do
rr:=subs([la[j]=Lamda[j],x[j]=S[m+1,j+1]],rr1):
r:=subs([x[j]=S[m+1,j+1]],rr11):
rr1:=rr: rr11:=r: j:=j+1:
end do:RS[alph]:=rr1;
RS1[alph]:=rr11;
end do:RS1;#RS;
syst:= [seq(RS[alph],alph=1..n)];
> var:=[seq(U[i],i=1..k)];
bs:=0;
printlevel :=4:
if IsProper(syst)=true then
B:=solve(syst,var);
for i from 1 to k do
if rhs(B[1][i])<0 then bs:=bs+1 end if:
end do:
if bs>=1 then "not stable" else "stable" end if;
else
"System is not stable" ;
 end if;
> maplet3 := Maplet([["agree these values?"],
[Button['q']("&OK", Shutdown("yes")),
Button['q1']("&No",Shutdown("No"))]]):
ss:=Maplets[Display](maplet3):
for i from 2 to n+1 do
print("x",m+1,"=",S[m+1,i]);end do;print("Pay-
Table=",Z);print("Lamda=", Lamda);
> while ss="No" do
mapl1:=Maplet([["Enter the no of x you want
to replace
with",TextField['TF']()],[Button("ok",Shutdown
(['TF']))]]):
d1:=Maplets[Display](mapl1):
d2:=parse(d1[1]):
unassign('ss');unassign('Mu');
unassign('Lamda');
unassign('Z');unassign('eq');unassign('MU1');
> for i from 1 to m do
> R1[d2,i+1]:=S[m+1,i+1];
> S1[d2,i+1]:=S[m+1,i+1];
for j from 1 to d2-1 do
S1[j,i+1]:=S[j,i+1];
end do;
for j from d2+1 to m do
S1[j,i+1]:=S[j,i+1];
end do;
end do;
> nnn(R1,S1,Q,P,f,g,n,m,k,MU1);
> maplet3 := Maplet([["agree these values?"],
[Button['q']("&OK", Shutdown("yes")),
Button['q1']("&No",Shutdown("No"))]]):
ss:=Maplets[Display](maplet3):
> end do:
RS:=Array(1..n):RS1:=Array(1..n):
for alph from 1 to n do
rr1:=eq[alph]:rr11:=eq[alph]:
j:=1:
while(j<=m) do
rr:=subs([la[j]=Lamda[j],x[j]=S[m+1,j+1]],rr1):
r:=subs([x[j]=S[m+1,j+1]],rr11):
rr1:=rr: rr11:=r: j:=j+1:
end do:
RS[alph]:=rr1;
RS1[alph]:=rr11;
end do:RS1;#RS;
syst:= [seq(RS[i],i=1..n)]:
> var:=[seq(U[i],i=1..k)]:
bs:=0;
> if IsProper(syst)=true then
B:=solve(syst,var);
for i from 1 to k do
if rhs(B[1][i])<0 then bs:=bs+1 end if:
end do:
if bs>=1 then "not stable" else "stable" end if;
else
"System is not stable" ;
 end if;
> if ss="yes" then
mapl4:=Maplet([["Which one you prefere,
x(",TextField['TF'](),")"],
```

```
[Button("ok",Shutdown(['TF']))]]):
dd:=Maplets[Display](mapl4):
d3:=parse(dd[1]):
> end if:
> for i from 1 to m do
> zz[i]:=subs({x[1]=S[d3,2],x[2]=S[d3,3]},f[i]);
> l:=d3;
> end do:
print("Mu=",Mu);
> for i from 1 to  m do
> print("x=",S[l,i+1]);end do:
for i from 1 to  m do
print("fmin=",zz[i]);
> end do;
for i from 1 to  m do
> print("Mu=",Mu[i,l]);
> end do;print("Pay-
Table=",Z);print("Lamda=",Lamda);
```

12/12/2010